Using the Oracle ODBC Drivers with Third Party Products

Rick Schultz,
Associate Technical Analyst
Microsoft Languages
Oracle Worldwide Customer Support
Revision 20:10032000
[CR File ID: 111334]

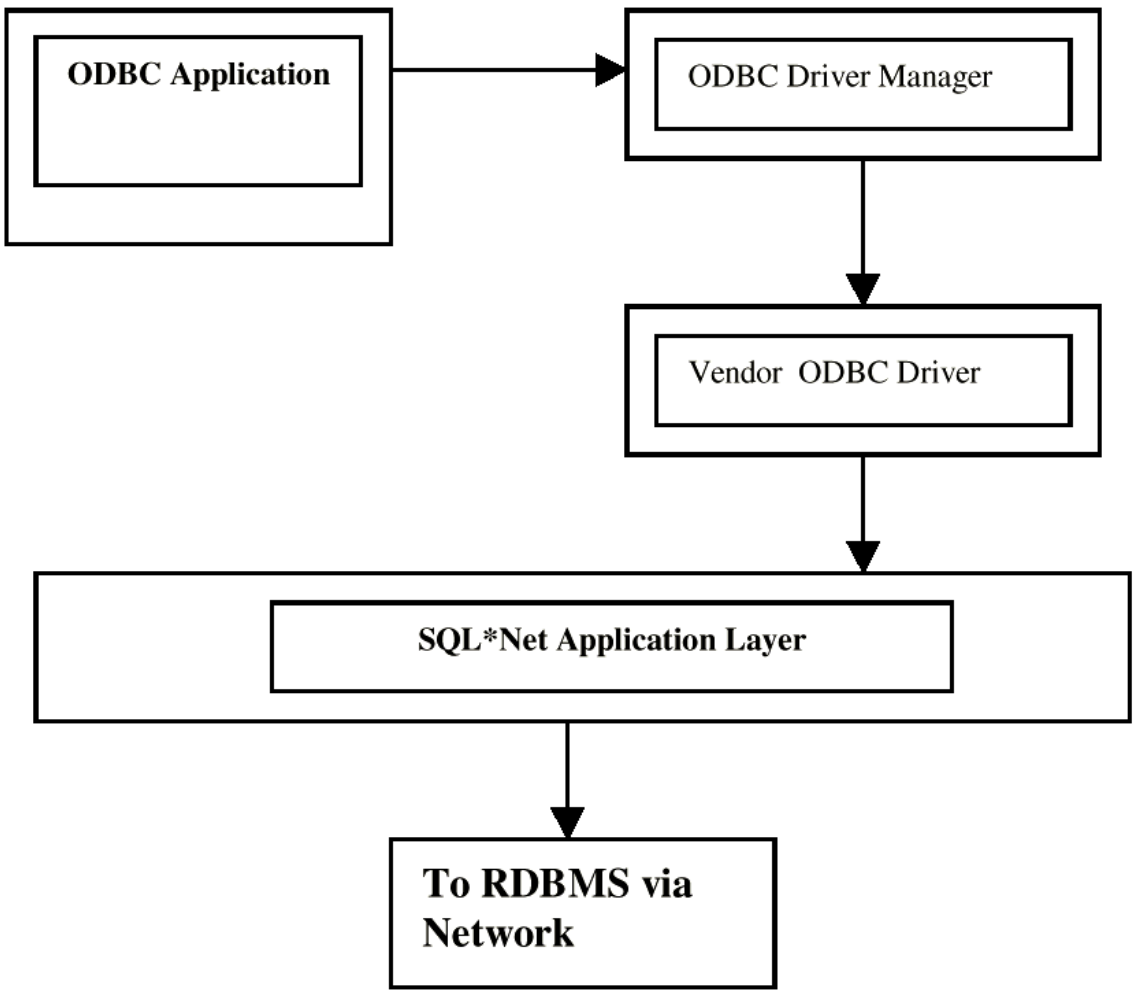
TABLE OF CONTENTS

I. Introduction

This white paper discusses the most common ODBC issues encountered by customers using various third party products, even if the specific product you are using is not mentioned, the same principles may apply. This document is kept current with the newer releases of the ODBC drivers and other 3rd party applications that we test, so always be sure that you have the latest copy. The latest copy is available from MetaLink [http://metalink.oracle.com/].

First some ODBC basics:

A visual representation of the parts of a typical ODBC connection:



As you can see, there are many layers to the connection, and often multiple vendors' components involved[1]. The ODBC application can be from any vendor, the driver

[1] With new products Like ADO, there are additional layers involved (such as the OLEDB-ODBC bridge).

manager can be from Microsoft, Intersolv, or some other vendor, the ODBC driver itself from an even greater number of companies (including Oracle) and the SQL*Net layer is provided by Oracle. With the advent of newer technology (such as Active Data Objects or ADO) there are even more layers that must be used for the connection to occur. This paper will discuss issues with using the Oracle provided ODBC Drivers for Oracle 7.x, or 8.x RDBMS versions and assumes that you are using the default driver manager provided by Microsoft. There are often inter-dependencies between these parts that are version specific, there is a specific version number associated with the Driver Manager layer as well as the ODBC Driver and SQL*Net, often problems with ODBC are caused by mixing together parts of differing versions that are not compatible. Oracle maintains a matrix of ODBC driver versions and SQL*Net versions on the WWW site where the drivers may be downloaded. Additionally, patches to the current downloadable driver may be obtained from Oracle Support via their external ftp site. When at all possible you should obtain and use the very latest version of the driver, as Oracle does not provide backports of bug fixes and enhancements of the driver to previous versions. The latest drivers are seldom bundled on shipping media (such as CDROM) but are available electronically via MetaLink (http://metalink.oracle.com/).

II. Conformance & Compliance: Driver Capability

The first *version* of the ODBC specification defined 2 *levels* of conformance:

- CORE
- LEVEL 1

The second major version introduced the next compliance level – LEVEL 2. The 2.5 version of the ODBC API Specification didn't add any additional conformance levels, but it did introduce some new level 2 calls. The 3.0 version of the ODBC API *specification* again did not add any new conformance levels, but it did the following:

- Added additional level 2 API calls.
- Deprecated all of the level 1 calls.

This can lead to a LOT of confusion when discussing the capabilities of a given driver (and I didn't even mention SQL Grammar conformance levels). Basically the Oracle drivers break down as follows:

- Driver versions through 1.16.x are all LEVEL 1 compliant to the version 2.0 ODBC API specification.
- Driver versions from 2.x to 2.5 are all LEVEL 2 compliant to the version 2.5 ODBC API specification.
- The version 8.0.3.x driver versions are also LEVEL 2 compliant to the version 2.5 ODBC API specification.
- Driver versions later than 8.0.4.x are LEVEL 2 compliant to the version 3.0 ODBC API specification.

- Driver versions 8.0.4.x and later are LEVEL 2 compliant to the version 3.0 ODBC API specification.
- Driver versions 8.0.5.8 and later are LEVEL 2 compliant to the version 3.51 ODBC API specification.

The SQL grammar conformance has been steadily increasing as the drivers have progressed. This not remarkable as it is mostly a function of the parser built into the driver, which has continued to improve with each release.

III. Different Methods of Using ODBC

Originally there were only a couple of ways to utilize ODBC technology for Data Access. There was the ODBC API directly from a program or using the Jet Database engine provided by Microsoft. There have been several additions to this as the ODBC API has matured you now have the following methods at your disposal (actual there are other methods, but these are the most common ones):

- ODBC API
- Microsoft Access/Jet engine
- RDO[2] (Remote Data Objects)
- ADO (ActiveX Data Objects AKA OLEDB) using the OLEDB/ODBC Bridge

Each of the methods has different requirements for the ODBC driver being used. For example:

- RDO will expect at a minimum that the driver used is fully compliant with the ODBC API Level 2 specification. There are some other factors regarding cursor support in the driver that may affect the level of functionality provided by RDO.
- ADO will expect that the ODBC driver being used is fully compliant with the ODBC API version 3 specification[3].

There are different versions of the ODBC Core components that are provided by Microsoft for redistribution. These include things such as the ODBC Administrator and the ODBC Driver Manager. These components are now referred to as the Microsoft Data Access Components (or MDAC for short). The 2.5 release was the version Originally shipped with NT 4.0 and Windows 95. There are three ways that these components get upgraded:

- You install an OS Patch or upgrade (Service Pack 6 for NT 4.0 is a good example).
- You install a product from Microsoft (or any other vendor) that is bundled with the newer versions.
- You can download the update from www.microsoft.com/data and install it.

[2] As of this writing, Microsoft has deprecated RDO in favor of ADO
[3] The version of ADO used will determine which 3.x specification it is expecting the driver to adhere to.

These upgrades are usually done without any operator notification and can lead to problems with different ODBC drivers. [Some of these problems will be discussed in detail in later sections of this paper dealing with specific problems.]

IV. A Bit of History

1. The original ODBC 1.x drivers that Oracle provided were completely re-written between the 1.11.x and 1.15.x (the next official version) releases and does not share a common code base with the earlier drivers. This is also true of the version 8.x drivers they have no common code base with the 2.x drivers. This sometimes resulted in bugs that were fixed in a much earlier version 're-appearing' in a later one.

2. Support for the ODBC drivers has fluctuated over the years, but with the 2.5 and later releases (particularly the 8.x series) oracle has committed to delivering drivers that not only work, but also are feature rich.

3. Contrary to popular opinion, Oracle writes and supports its own ODBC drivers. There are many 3rd party vendors who also write drivers for Oracle databases, some are free (as is the Microsoft ODBC for Oracle) and others are not (such as the Merant (Intersolv) Oracle ODBC driver). This paper is intended to address only the Oracle provided ODBC drivers.

V. Supported Driver Versions

Unlike some of the third party drivers, the Oracle provided driver has strict configuration compliance for the client installation[4]. Before upgrading the client components, check the following table for compliance:

Full Release	Platform Supported Operating System	SQL*Net Requirements	RDBMS	Required Support Files
8.1.6.1.0	Windows 95 or 98 or NT 4.0 or 2000	8.1.6	7.3.4 or higher	8.1.6
8.1.5.5.0	Windows 95 or 98 or NT 4.0	8.1.5	7.3.4 or higher	8.1.5
8.0.6.1	Windows 95 or 98 or NT 4.0	8.0.6	7.3.4 or higher	8.0.6
8.0.5.8.0	Windows 95 or 98 or NT 4.0	8.0.5	7.3.4 or higher	8.0.5
2.5.3.1.7	Windows 95 or 98 or NT(3,51 or 4.0)	2.3.x	7.3.2.1.1 or higher	7.3
2.5.3.1.6	Windows 3.1 or Windows for Workgroups 3.11	2.3.x	7.3.2.1.1 or higher	7.3

[4] Note here the 'client' referred to is the ODBC application that will be using the driver. This is true even if the application is executed on the database server machine.

In general, the ODBC driver shipped with the given client software is valid for that version of the client. It is quite easy to install an older driver from an earlier release of the client software CD, so be sure that the version you are installing also matches the Oracle client software you are currently using. Additionally, the driver is frequently updated aside from the Oracle client and a later version may be available for the version of the Oracle client installation you are using. As of this writing, all of the version 1.x and 2.0.x Oracle provided drivers are obsolete and have been de-supported and replaced with the 2.5.x or later versions. In addition, 8.0.3 and 8.0.4 have also been de-supported and removed from availability.

VI. Multi-Byte character sets and Internationalization

The Oracle provided ODBC driver[5] (below 8.0.x) is single-byte, that is to say it does not support Unicode or other multi-byte characters either at the system character set level or at the Oracle NLS translation level. There are other issues with NLS usage in ODBC:

- The ODBC API has specifications of its own for certain things such as dates & times.
- The ODBC Driver may use regional settings from the control panel for some translations. Or they may have an optional translator DLL that can be specified. If so this will be controlled by the Datasource (DSN) configuration settings for the driver.

Remember that the Oracle NLS translation happens at the Oracle Call Interface[6] (OCI) layer and it is then passed up to the ODBC driver for manipulation as needed before getting displayed on the client. This means that any ODBC API conformance for formatting of data would be done AFTER it had already been formatted via NLS.

VII. Dynaset Open Time and Scrolling Speed

Many of the 'high level' ODBC access methods discussed in section II use the concept of a 'Dynaset'. This can be thought of as a 'smart' snapshot that tries to keep its contents synchronized with the RDBMS. Obviously this adds more overhead than a simple static snapshot[7]. The main factors that adversely affect Dynaset open time and scrolling speed, are the number of columns you select and the number of the query tables that are output. Select only the columns you need; outputting all columns using SELECT TABLE.

[5] Oracle does provide a Kanji specific Japanese ODBC driver, which supports that specific multi-byte NLS character set.

[6] This is the 'low level' API for programs connecting to Oracle.

[7] The term Snapshot in ODBC does not mean the same as with the Oracle RDBMS. An ODBC Snapshot can be updated, but its contents will not reflect the change until it is refreshed.

SELECT * is more convenient but slower. Also consider the effects of multiple users if this is a concurrent application.

A Dynaset will try updating itself to reflect changes that you make to the RDBMS using the Dynaset itself. It does not and cannot know about changes made by other users (or even the RDBMS itself via stored procedures, functions, triggers, etc.) until a refresh of the Dynaset is done or an update is attempted. Some drivers support a Dynaset that is live with respect to data changes made on the RDBMS as well; Oracle drivers do not support this type of Dynaset, the Microsoft ® definition of a Database side Dynaset implies that the RDBMS supports bi-directional cursors. Oracle RDBMS cursors are forward scrolling only. Additionally, some RDBMS platforms support the concept of 'dirty-reads', which allows one user to see data that has been modified by another user, but not yet committed, again Oracle does not permit this kind of data access.

Snapshots are often faster to open and scroll than a Dynaset. If you do not need the features of a Dynaset, use a snapshot. This means that careful consideration is required on the part of the user when designing ODBC applications for use in a client/server multi-user environment. This is especially true in cases where shared tables will be accessed to prevent users from locking each other out.

VIII. Using the ODBC Administrator

The ODBC Administrator is the Microsoft component of ODBC that allows you to configure and use an ODBC Datasource (usually referred to as a DSN – or Data Source Name).

A. Determining the version of the ODBC Administrator

For 16-bit the last version of the Administrator released was 2.5 The current 32-bit version is 3.52 and is part of the MDAC 2.5 upgrade. The ODBC Administrator is normally run as a Control Panel applet (Oracle redistributes the Administrator with the ODBC driver, but you should run the one in the Control Panel by preference as it may be a newer version). If the Administrator dialog has tabs across the top then this is the 3.x version, otherwise it is the 2.x version. The latest versions of the ODBC common components are available on the Microsoft Web site (http://www.microsoft.com/data).

1. A history of ODBC Administrator versions

Microsoft Product	ODBC Administrator version
Windows NT	2.50.3321
Visual Studio 97/ Office 97 PE	3.00.2301.
Windows NT sp3	3.00.28.22.
Mdac 1.5	3.50.32.14.
Visual Studio 6 EE	3.510.3002.13 (Mdac 2.0)
Mdac 2.1	3.510.(3510.0 to 4202.0 depending on version)
Mdac 2.5	3.520.4403.2.
Mdac 2.5 sp1	3.520.5303.2.
Mdac 2.6	3.520.6526.0

B. Configuring an Oracle DSN

The Oracle ODBC driver configuration has changed slightly with each newer version of the ODBC driver, but there are only 2 pieces of information that are absolutely required to create a working Oracle DSN:

1. A DSN Name (whatever you like)
2. A valid SQL*Net connect string (TNS Alias or Oracle 8 service Name)

In Oracle 8, this *connect string* is sometimes referred to as the Service name, and in some versions of the 7.3 driver (and in some 3rd party drivers) this is sometimes called SERVER NAME. This does not mean to use the network name of the server or even the SID name, but rather the actual TNS Alias that you configured when you configured the SQL*Net layer. You can look in the Oracle_Home\network\Admin (or Net80\admin) directory for the TNSNAMES.ORA file (a text file) to see what TNS Aliases is configured. You could also use the *SQL*Net Easy Config* (or *Net8 Easy Config* depending on your version of SQL*Net and the driver you are configuring) program to view this information. In the case of a local RDBMS you would normally use the **Beq-Local** alias, normally created for you when the RDBMS is installed. If you are unsure (or if you are using Names Server) you may need to consult with your Oracle DBA or whoever configured your SQL*Net connection.

C. DSN Options (version 8.x driver)

Here is a table of the DSN configurable options supported by the 8.x drivers. This is provided to support manual DSN configuration or use of a DSN-less connection (programmatically). These are the optional keywords supported in the CONNECT parameter of the SQLDriverConnect() ODBC API call:

Keyword	Meaning
DSN	ODBC data source name
UID	User ID or user name
PWD	Password (specify PWD=; for an empty password)
DBQ	Service Name
DBA	Database attribute (W=write access, R=read-only Access)
TLO	Translation option
TLL	Translation library name
PFC	Prefetch count (specify a value zero or greater)
APA	Applications Attributes
FEN	Failover Enabled (T=Failover Enabled)
FRC	Failover Retry Count (Number Value)
FDL	Failover Delay (Number Value)
LOB	LOBs Writes Enabled (T=LOBs Enabled)
RST	Results Sets (T=Result Sets Enabled)
FRL	Forced Retrieval of Oracle Long Col (T=Forced Long Reads Enabled)
MTS	Microsoft Transaction Server Support (T=Disabled, F=Enabled) (only in 8i)
QTO	Query Timeout Option (T= Query Timeout Enabled)
CSR	Close Cursor Enabled (T= Close Cursor Enabled, F=(Default) Close Cursor Disabled)

D. Tips on improving performance

It is possible to improve performance of the Oracle 8.x ODBC driver by implementing one or more of the following options:

Unchecking the following options in the ODBC Configuration Window:
- Enable LOBs
- Enable Result Sets
- Enable Closing Cursors

And/or checking:
- Disabling MTS Support (In 8i only)

You may want to test these options out with your application since other options in the Configuration Window may also provide additional performance benefits.

Also make sure that you are not unchecking any options that you need to use.

E. DSN Configuration Dialog fails to open (ODBC Driver Setup)

This can be caused by NOT having the ORACLE_HOME\BIN directory included in the DOS/System search path. You need to verify that this is actually the case, open an MSDOS prompt and type PATH and press return. Examine the path to insure that it actually includes the ORACLE_HOME\BIN directory. If the path is NOT correct in your environment:

- Look in your AUTOEXEC.BAT and insure the syntax is correct (no spaces, etc) for the path statement (Windows will ignore the entire

statement if it is not) and that it includes the appropriate ORACLE_HOME\BIN directory.

- If the path is correct in the AUTOEXEC.BAT and still is not correct in your MSDOS environment after startup, you may need to check with your network administrator to insure that the path is not overridden by network startup batch files or by a Mandatory profile. This can happen if your pc is on a Novell network and the pc has Windows 95 installed.

F. Tracing ODBC Calls

SQL generated by the ODBC layer, specifically the Microsoft ® Jet engine, may vary drastically from what you think is being sent to the RDBMS. In fact, this is almost always the case unless you are directly using the ODBC API itself. You may see varying results from a query or the results are different than those generated from a SQL*Plus query. Understanding this can help in resolving SQL issues. The ODBC trace will reveal what the ODBC application is sending to the ODBC driver. The SQL*Net trace will reveal what the ODBC driver is, in turn, sending to the RDBMS.

1. Enabling/Disabling Tracing

A simple ODBC trace can be turned on via the ODBC administrator[8]. Enabling a client SQL*Net trace can be helpful in determining what SQL is actually being applied. To enable an ODBC trace with the 2.5 ODBC administrator, under OPTIONS, select **Turn on Tracing**, this will also let you optionally specify the trace output filename and location. The default is SQL.LOG. [With the 3.x administrator there is a **TRACING** tab that contains this information]

2. No trace is generated…

If you have enabled an ODBC trace using ODBC Administrator, yet no trace file is generated, you should consider re-installing the ODBC Common components (from one of the MDAC kits that Microsoft has available for download). This indicates a problem with the underlying core ODBC components, as they are what actually generate the ODBC trace. The alternate possibility is that the application is simply never getting to the point of issuing any ODBC API calls.

[8] If you are on a platform such as Windows 95 or NT, be certain that you are enabling the correct trace. 16 bit drivers should have the trace enabled in the 16 bit administrator, and 32 bit drivers in the 32 bit administrator.

*G. Tracing the SQL*Net connection*

To enable SQL*Net client tracing make the following alterations to the SQLNET.ORA file, located under the client ORACLE_HOME\Network\Admin directory (NET80\Admin directory for SQL*Net 8.0). **Please remember to backup the existing copy of your SQLNET.ORA first!**

Entry	Value
Automatic_ipc	OFF
Trace_unique_client	ON
Trace_level_client	16
Trace_directory_client	[path]*
Trace_file_client (this is optional)	[filename]

*** Where [path] is a valid directory for the trace files to be written to, if this is a ROOT directory, you must include the trailing backslash (c:\).**

H. Other problems configuring the ODBC DSN

If the path issue noted above is not at fault, then you should insure that correct versions of SQL*Net and the Required Support Files (RSF) are installed for the ODBC driver you are trying to configure. It may also be advisable to download and install the latest version of the ODBC Common components from the Microsoft web site to correct any problems that may exist in the underlying ODBC installation itself.

IX. Required View and Table Accesses

In general the ODBC engine will require access to specific DBA views that exist in the Oracle Data Dictionary, the primary ones are:

- ALL_CATALOG
- ALL_CONSTRAINTS
- ALL_OBJECTS

There are other tables (each with the ALL_ prefix) that may also be needed depending upon the operation you attempt. For an ODBC connection to work, the logon username/password that does the logon must have READ access to these views for ODBC to function correctly. For an application that uses Microsoft ® Jet (Microsoft ® Access, Microsoft ® Visual Basic, etc.) to be able to update a table you must have a primary key constraint[9] on the table which is verified by Microsoft ® Jet through these catalog views. This restriction is not applicable to Oracle specific tools such as SQL*Plus

[9] A combination of the UNIQUE and NOT NULL constraints on a particular column (or columns) is considered a PRIMARY_KEY

as they are ‘Oracle Aware’ and will use the ROWID to uniquely identify a row to be modified.

X. Database Security and ODBC

For the Database Administrator, use of ODBC applications can greatly complicate user security. Suppose a user was given a fixed username/password and the appropriate update privileges granted for use with existing database applications (such as Oracle Forms or other 3GL applications), which allowed the user (through these applications) to modify the RDBMS tables. Use of ODBC with such applications as Microsoft Access now allows the user to easily browse the underlying tables and make edits directly to them, possibly bypassing business logic that was coded into the applications normally used. The bottom line is that ODBC will inherit whatever RDBMS privileges a given user is given. ODBC defers all security for a user to the RDBMS level. Most ODBC drivers for Oracle will also not utilize a **Product User Profile** (such as SQL*Plus uses). In order to provide for only approved applications to modify the data, you may need to implement non-default roles or profiles. These would then be set by the application after it has connected. This means that a given username/password has a very minimal set of privileges (such as select only) and then is granted the needed updates privileges through an ALTER USER… command. The Database Administrator will then control these roles and privileges.

XI. Causes for READ Only Dynasets

Due to the way in which Oracle stores information in the data dictionary, Microsoft ® Jet will not be able to verify a primary key constraint for Synonyms; this causes them to be read-only via ODBC.

Another cause of a read-only Dynaset can be the use of ‘SQL Passthrough’. This causes Microsoft ® Jet to be left out of the picture, such that the SQL is passed on to the RDBMS unmodified. The drawback is that all results are read-only. See the Jet/DAO section for some specific issues with that product.

While Oracle is quite happy having certain objects (tables, synonyms, etc.) share the same name, ODBC may not be (actually ODBC itself doesn’t care, but the Microsoft Jet engine does). You will receive errors attempting to access these objects, which makes these objects inaccessible via ODBC/Jet. Other applications may also impose these same restrictions.

XII. Thread Exceptions & Multithreading

All currently supported Oracle ODBC drivers support multi-threading (Version 2.0 and above). Using Oracle's 32-bit Version 1.x drivers with applications that are multi-threaded may cause problems with thread exceptions. Multi-threaded and thread-safe Oracle ODBC drivers (32-bit Version 2.x) only work correctly with Oracle RDBMS Version 7.3.x or later client libraries and a 7.3 or later RDBMS.

XIII. Joins and the Oracle ODBC Driver (ORA-0933 & 0936)

In the version 1.x ODBC drivers that Oracle distributed, there was (limited) support for JOIN statements using either the ODBC Join Escape syntax or the Oracle syntax. Full support for JOIN operations is included in the version 2.x driver, but the ONLY syntax supported with this version of the driver is the ODBC JOIN Escape syntax, not Oracle syntax (although the Oracle syntax will still work in most PassThrough queries). Here is an example using the standard EMP & Dept sample tables in the SCOTT schema:

```
Example Oracle Join SQL:
============================
select dept.deptno, sum(sal)
from dept, emp
where emp.deptno(+) = dept.deptno
group by dept.deptno

Example ODBC Join SQL:
============================
select dept.deptno, sum(sal)
from {oj dept left outer join emp
on dept.deptno = emp.deptno}
group by dept.deptno
```

Here is a complete example using ODBC syntax with a three table join:

```
--SQL to create tables and populate
create table table_one (one_id integer, one_desc char(20));
create table table_two (two_id integer, two_desc char(20));
create table table_three (three_id integer, three_desc char(20));
insert into table_one (one_id, one_desc) values(1, 'TABLE1-1')
insert into table_one (one_id, one_desc) values(2, TABLE1-2‘')
insert into table_one (one_id, one_desc) values(3, 'TABLE1-3')
insert into table_two (two_id, two_desc) values(1, 'TABLE2-1')
insert into table_two (two_id, two_desc) values(3, 'TABLE2-3')
insert into table_two (two_id, two_desc) values(4, ‘TABLE2-4’)
insert into table_three(three_id, three_desc) values(1, 'TABLE3-1')
insert into table_three(three_id, three_desc) values(2, 'TABLE3-2')
insert into table_three(three_id, three_desc) values(5, 'TABLE3-5')

--the select statement (can be done in ODBCTEST)
```

```
select one_id, one_desc, two_id, two_desc, three_id, three_desc from
{oj table_one left outer join table_two on one_id=two_id},
{oj table_one left outer join table_three on one_id=three_id}
```

[37000: [Microsoft][ODBC driver for Oracle][Oracle]ORA-00923: FROM keyword not found where expected.

This error msg comes from attempting the following sql:

"SELECT xxxx.yy FROM xxxx, TRY WHERE ((xxxx.yy = www.zz(+)))"

This is unsupported at this time.]

XIV. Calling a Stored Procedure Via ODBC

[See also the Stored Procedures from Microsoft ® Visual Basic section of this paper.]

The following is an example of the Visual Basic syntax for calling a stored procedure via ODBC:

```
db.ExecuteSQL("{CALL procedurename(param1,param2,param3)}")
```

NOTE: This assumes input parameters only and that you have assembled this such that each of the parameters is embedded into the string as a literal. Also note that this syntax DOES NOT work with packaged procedures, for those you must use the alternative ***begin …end;*** *syntax.*

In the above example dB is assumed to be a valid database object. If you are using a tool such as MSQuery just use the {CALL ...} (ODBC Procedure Call Escape) syntax without the double quotes. You must include the () even when you don't have any parameters. Out parameters are supported at the ODBC Level 2 conformance (Oracle7 ODBC Version 2.x). The Oracle Level 1 drivers (Version 1.x) will not support this, you must be using a Level 2 or better driver. The 7.3 Oracle driver does not support returning dynasets. This functionality is first implemented in the 8.0.5.x version of the driver. An alternative to the call syntax is shown below:

```
db.ExecuteSQL("BEGIN procedurename(param1,param2,param3); END;",
SQLPASSTHROUGH)
```

This alternative does require the use of the SQLPASSTHROUGH parameter, but will also allow for calling packaged procedures (i.e. packagename.procedurename()).

To return a result set with a stored procedure, refer to the following Microsoft knowledge base articles:

- Q147938 (RDO)
- Q126992 (DAO)

The Microsoft provided Oracle ODBC supports this functionality through the use of PL/SQL table types. The Oracle provided drivers do not support this

functionality prior to version 8.0.5.x (where it is implemented in PL/SQL by returning a REF CURSOR).

For simple output parameters from a stored procedure you could use the following SQL:

```
{call procname(?,?)}
```

The above would be passed to SQLExecute() and then have called SQLBindCol() or SQLBindParameter() for the output bind variables (the variables referred to by the '?') you defined in your program. [Note: the Begin; … End; syntax would also work just as well here.] If you are using the Oracle 8.0.x ODBC driver and receiving an ORA-6502 and/or ORA-6512 errors, you must upgrade the driver to version 8.0.3.0.1 or later and update your MDAC to the latest version.

XV. Long & Long Raw Data via ODBC

There is a 32 KB limitation on the size (per chunk) of data blocks read to/from a long raw column in the RDBMS using a combination such as SQLParamData() and SQLGetData()/SQLPutData() to retrieve the LONG/LONG RAW data. To insert character data into a LONG column, you must use a BIND variable. Inserting more than 2000 characters inside single quotes ('') as a string literal, is not allowed.

A. Long data is truncated when using MSQuery

MSQuery appears to have a fixed maximum buffer of 30000 bytes that it allocates and it does not attempt to do a piecewise retrieval of the data. This is not a problem with the driver itself, but rather MSQuery.

B. ORA-3127 using Oracle 8.0 driver on a LONG column.

This is a known issue (BUG # 666935). It is dependent upon the column position of the LONG column in the table, so one possible workaround is to change the position of the LONG column to be other than the LAST. This is not a good design for tables with LONG data, as it will encourage row chaining and migration. If your DSN config window shows the option, 'Force Retrieval of Long Columns', enable it. If it doesn't, you need to update your ODBC driver to 8.0.4.4 or better.

XVI. Oracle 8 LOB (Long or Large Object) Columns

The Oracle 8.0.4 ODBC driver was the first version supplied by Oracle that conformed to the ODBC version 3 API specification. The drivers released prior to this ODBC API

only supported level 2 compliance of the version 2.5 specification. The first versions of this driver allowed READ accesses to the new LOB column data types in Oracle 8. Starting with version 8.0.5.1.x, write access was also allowed[10]. In order to access the LOB data types you simply bind them as the appropriate ODBC data type (LONG VARCHAR, LONG VARBINARY) as you would LONG or LONG RAW columns, or use the new ODBC API data types that Oracle registered with Microsoft[11] (BLOB, CLOB). [Note: A SQLDescribeCol may report the new data types] These new data type declarations should be included with the MDAC 2.0 and later releases from Microsoft, but for compatibility, you should simply use the appropriate LONG VARxxxx type.

A. Data incorrectly inserted into a BLOB field (data corrupted)

This is bug# 764772 and is corrected in the 8.0.5.2.0 and later versions of the Oracle ODBC driver. The corruption was actually occurring on the insert (the data was getting truncated).

B. ORA-932 when inserting a BLOB field

While the driver did not support writing LOB data until 8.0.5.1.0, this was a bug in that release (bug# 760239) and is resolved in version 8.0.5.2.0 and later of the ODBC driver.

XVII. Maximum Length of a SQL Statement

For the versions of the driver before 1.13.5.x had a limit of 4k characters for a SQL statement, after this version the limit was 30,000 characters. The 2.5.3.1.5 and later 7.3 drivers had a limit of 64,000 characters. There is effectively no limit in the 8.x drivers (the actual limit is the maximum size of an unsigned integer value).

The Oracle ODBC test utility has a limitation of 512 bytes.

(However, there have been some known issues with 3rd party products truncating

the sql string. This happens before the sql string is passed to the Oracle

ODBC driver. The 3rd party vendor would need to be contacted in this case.)

[10] Except for the BFILE type which is read-only at the RDBMS level.
[11] Originally in the early 8.0.x releases you needed to use the new data types (BLOB, CLOB) explicitly when accessing these types of columns. This was changed for ease of use beginning with the 8.0.5.x drivers.

XVIII. "Specified Driver Could Not be Loaded" Error

This may or may not also be accompanied by an ORA-3121 or an ORA-12154 error (see the native error code).

A. ORA-3121:

This simply means that when the ODBC Driver Manager attempted to load the Oracle ODBC driver, the driver itself failed to load one of the components that it depends on itself (i.e. SQL*Net or the Required Support Files (RSF)). These files are located and loaded via the System/DOS search path (this is true even on Windows 95 and NT). You can verify that the PATH is NOT an issue with this error by opening a Command Prompt/MSDOS window and issuing the PATH command (with no arguments). The resulting path displayed MUST contain the Oracle_Home\BIN directory. If using a 16-bit driver on a 32-bit version of Windows, the 16 bit SQL*Net and RSF must be installed as well. These can co-exist with the 32 bit versions as they get their own Oracle_home and \BIN directories which would also have to be in the PATH). In the following example F:\ORANT\BIN is the Oracle_Home\BIN directory in question:

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

c:\users\admin>path
PATH=C:\WINNT40\system32;C:\WINNT40;F:\ORANT\BIN;C:\DOS

c:\users\admin>
```

[Note: refer to section VIII.C for further details on correctly setting the path environment]

B. ORA-12154:

This is error indicates a failure to lookup the SQL*Net/NET8 Alias, used for the DSN configuration, by the underlying SQL*Net layer. This alias is typically defined in the TNSNAMES.ORA file. This is normally the ORACLE_HOME\NETWORK\ADMIN directory (or NET80\ADMIN).
The SERVER name or SQL*NET Alias is the same information stored in this configuration file, so insure that you provided the correct information when configuring the ODBC DSN.
Insure that this is the ONLY place on your system where a TNSNAMES.ORA file is located, due to a quirk in the network layer, it will load a TNSNAMES.ORA found in a current working directory in preference to one specified in this file.

[Use of the Oracle SQL*NET Names server will change how this is done, contact your system administrator for details on that configuration.]

XIX. “Driver Not Capable” Error

This error is returned when the calling program asked the driver to make a data conversion (data types) that the driver cannot perform. Try binding all of the variables to string or character variables; the Oracle driver can convert almost any data type to string. The actual issue in this case is usually the SQL conformance of the driver. For additional information on API compliance and SQL Conformance, please refer to the ODBC Programmer’s Reference, which is available from Microsoft ®.

XX. “Conformance Error” or “Function Not Supported by Driver” Errors

These errors are most often generated when the calling application has attempted to use an ODBC API call which is not supported by the driver (i.e. making an ODBC Level 2 call into a driver that is only Level 1 compliant). The calling program exceeding the SQL conformance of the driver can also cause this. There are some specific conformance errors documented later in this paper, see those as well. Function not supported by Driver is pretty straightforward, the calling application asked the ODBC driver to do something that it simply does not have the support built in to do. Usually this is caused by the application failing to check what ODBC API functions the driver is capable of supporting. See section XXXII later in this paper for known issues with specific conformance error messages.

XXI. Microsoft ® Jet -specific Issues (DAO)

For Microsoft ® Visual Basic 3.0 users, you should have installed the Microsoft ® Jet 2.0 engine upgrade when installing Microsoft ® Visual Basic 3. There are known problems with earlier versions. See Microsoft's Knowledge Base Article #Q113594. The Oracle ODBC driver will assume that you are using at least Jet 2.x.

For Microsoft ® Visual Basic 4.0 users, often Microsoft ® Visual Basic 4 will not default to displaying ODBC data sources for you to select from in a property sheet dialog. You can still manually enter the DSN (example: ODBC;DSN=orlnt1;UID=scott;PWD=tiger;)

A. If using Visual Basic 4.x (VB4), 5.x (VB5) or 6.x (VB6)

Under the TOOLS/References menu (VB4) or Project/References (VB5) selection you may have to use the Microsoft DAO 2.5/3.0 Compatibility Library depending upon the version of the Oracle ODBC driver you are using, if you receive errors try using the other library. If using VB 6 insure that you have installed Service Pack 3.

B. Microsoft ® Jet and Joins

It is possible with Microsoft ® Jet to get the appearance of updateable JOIN SQL. When using Microsoft ® Jet to do the actual JOIN, what really happens behind the scenes is that Microsoft ® Jet will make local copies of the JOIN tables and apply the updates separately. Obviously this can be memory intensive if the SQL is returning a lot of data.

C. Microsoft ® Jet and Multiple Connections

Microsoft ® Jet will open multiple connections to the RDBMS, this is due to the way the ODBC driver manages WRITE connections and locking. See the Microsoft ® Developers Network Article on Optimizing ODBC for a further explanation or the Microsoft's article titled "ODBC: Architecture, Performance and Tuning". This can lead to issues on the RDBMS with the MAX_OPEN_CURSORS parameter.

D. Public Synonyms with Jet ("Table or View does not exist")

SHOW SYSTEM Objects option must be set within the Jet application, and Jet will become confused (and fail) if the public synonym has the same name as the table on which it is based. As an example, if you create a public synonym on the EMP table and name it EMP. Jet will then not be able to access the synonym correctly. This limitation on unique names extends to any two objects that are visible to the Jet engine for this connection. Attempting to specify the schema name onto the object will not help as Jet will then issue an error: "can not open xxx.mdb" where xxx is the name of the object you are attempting to access. The workaround is to insure that within your entire schema (tables, views, etc…) are unique.

E. #Deleted Rows in Microsoft ® Jet

If a database trigger were to update a table such that Microsoft ® Jet can no longer uniquely identify the row, these rows will display as #DELETED in the data

sheet views. Having a floating-point column as the bookmark or a Primary-key/index column can also cause this same symptom. Because machines vary in how precisely they handle floating-point data, occasionally precision loss can occur. The actual loss may be small enough to be insignificant, but if the data forms part of a table's bookmark Microsoft ® Jet will think the row has been deleted. This is because Microsoft ® Jet asked the server for the row by its key values, but no exact match was found and it cannot distinguish this situation from that of a real record deletion by another user.

F. Troubleshooting Connection Timeouts (ORA-1013)

Version 2.x of the Oracle ODBC driver (client 7.x) does not support the QueryTimeout option. Oracle ODBC driver 8.0.5 added this feature and had it enabled by default (with the only way to turn it off being to make ODBC API calls). Since Jet® could then timeout the query it would automatically set this up and use it. It was now possible to have queries that worked fine with the older driver, fail due to excessive execution time with the new driver. Versions 8.0.5.3 and later of the ODBC driver provide a feature to disable query timeout.

See the following sections on Configuring Microsoft ® Jet, and refer to the ConnectionTimeout entry and set it to the number of seconds you want the aging timeout to be; the default value is 600. Once you've established the ConnectionTimeout, you may want to force idle connections closed by using Microsoft ® Visual Basic 's FreeLocks statement in your code. This INI file section can also contain QueryTimeout and LoginTimeout entries that specify, in seconds, how long Microsoft ® Jet will wait for the server to perform queries and logins before it aborts. The respective default settings of 60 and 20 seconds for these entries may be too low. Try 120 for each or experiment in your environment and see what works best.

Note: Simply editing the INI file is not enough, for 16 bit Access the queries must be recreated to pick up on this change. For 32 bit Access you can create a new "ODBC" key in the registry path:

```
\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\3.x\Engines\ODBC
```

In the above line, replace the 3.x with the actual version number of the engine you are using (3.0, 3.5, etc.). Create a new value named QueryTimeout in this key then set this as appropriate and it will override the default setting. You can set ConnectionTimeout values here as well.

Alternatively, with version 2.5 and later of the ODBC driver, a checkbox was included in the DSN configuration dialog to allow forcibly disabling asynchronous behavior, reverting back to the behavior of the earlier drivers.
While the Oracle 8.0 drivers do not implement asynchronous support, they do implement the QUERY_TIMEOUT option of the SQLSetConnect call (which was not supported in the 7.x series drivers). With this option, jet can still implement the timeout. As mentioned above, version 8.0.5.3.0 and later drivers support a DSN configurable option to disable the SQL_QUERY_TIMEOUT support in the SQLSetConnectOption call[12].

1. Microsoft Access

Some versions of Microsoft Access do NOT allow overriding the default timeout parameters with ANY INI OR REGISTRY settings (contrary to the product documentation). In this case it MUST be done on a query-by-query basis, using the query properties page. When performing a join query in Microsoft Access 97 setting QueryTimeOut to 0 and increasing Logintimeout and Connectiontimeout in the registry for the Jet Engine along with the Refresh Interval in Microsoft Access does not fix the problem. The Refresh Interval can be found under the menu options: Tools->Options->Advanced. In this case, you need to go one step further. Set the ODBC timeout to 0. This property belongs to the query itself in MS Access 97. To set the ODBC timeout, go into the design view for the new query you are trying to add, go to view/properties and set ODBC timeout to 0 instead of 60.

2. Stored Procedures and ADO

Executing stored procedures using ADO takes a long time, even though it runs fine at sql plus level.

ADO gives an interface to execute the stored procedure called command object. ADO has a property called CommandTimeout which is by default set to 30 seconds. If the stored procedure is taking longer than this, the execution of the stored procedure is canceled and user gets the above error.

Example:

```
Public Sub ConnectionStringX()

   Dim cnn2 As ADODB.Connection
```

[12] Since asynchronous support does not exist currently with the 8.0 drivers, the application can't cause a QUERY TIMEOUT to occur. This was the behavior in the 2.0 and earlier 7.x drivers.

```
' Open a connection using a DSN and ODBC tags.
   Set cnn2 = New ADODB.Connection
   cnn2.ConnectionString = "DSN=Pubs;UID=sa;PWD=pwd;"
   cnn2.ConnectionTimeout = 30
   cnn2.Open

' Display the state of the connections.
   MsgBox "cnn1 state: " & GetState(cnn2.State) & vbCr

   cnn2.Close

End Sub
```

Set the CommandTimeout property to 0 programmatically. This could be achieved for the particular query/stored procedure in question by setting the property for the command object or for all the queries/ stored procedures at the connection object level.

G. Jet/DAO and Dates

Jet expects to use only a VBA style date format, thus to get a date datatype to work correctly with the Oracle ODBC driver you should use the following syntax:

```
Dim odb As Database
Dim ody As Recordset

Set odb = OpenDatabase(ODBC, False, False, _
"ODBC;DSN=ORA32;UID=SCOTT;PWD=TIGER")
Set ody = odb.OpenRecordset("SCOTT.EMP", dbOpenDynaset)
ody.FindFirst "Hiredate = #03-DEC-81#"

Do Until ody.NoMatch
 Combo1.AddItem ody.Fields("ENAME")
ody.FindNext "Hiredate = #03-DEC-81#"
Loop
```

H. AutoCommit

You may need the non-core function SQLSetConnectOption(), [replaced by **SQLSetConnectAttr()**], to turn off auto-commit mode (SQL_ATTR_AUTOCOMMIT is the attribute being set) which is the default in most drivers that support it. In this mode, the driver automatically commits each individual SQL statement upon execution. See the "ODBC Programmer's Reference" for more information on these API calls. If you are not using the ODBC API directly, then you may not be able to disable this behavior as it would be controlled by the ODBC application.

I. Data Type Mapping

ODBC Datatype	Microsoft ® Jet Datatype
SQL_BIT	Yes/No
SQL_TINYINT	Number Size: Integer
SQL_SMALLINT	Number Size: Integer
SQL_INTEGER	Number Size: Long Integer
SQL_REAL	Number Size: Single
SQL_FLOAT	Number Size: Single
SQL_DOUBLE	Number Size: Double
SQL_TIMESTAMP	Number Size: Double
SQL_DATE	DateTime
SQL_TIME	Text
SQL_CHAR	Text
SQL_VARCHAR	if length <= 255 then Text (Field Size = length)
	if length > 255 then Memo
SQL_BINARY	
SQL_VARBINARY	if length <= 255 then Binary (Field Size = length)
	if length > 255 then OLE Object
SQL_LONGVARBINARY	OLE Object
SQL_LONGVARCHAR	Memo
SQL_DECIMAL SQL_NUMERIC	if wScale = 0 then if Precision <= 4 then Number Size: Integer
	if Precision <= 9 then Number Size: Long Integer
	if Precision <= 15 then Number Size: Double
	if wScale > 0 then if Precision <= 15 then Number Size: Double
	Any other field types mapped to Text (Field Size = 255).

J. Configuring Microsoft ® Jet

Microsoft ® Jet applications use an INI file to configure how Microsoft ® Jet will manage certain functions. For Microsoft ® Access it is msaccess.ini (or msacc20.ini) and for Microsoft ® Visual Basic, it is vb.ini. For an application created with Microsoft ® Visual Basic, it is determined by the application. The following is a list of entries that can be made in the application INI file to control Microsoft ® Jet. All of them go under a section called [ODBC] with the exception of RmtTrace which goes in a section called [Debug]. With the 32 bit versions of Jet this information is contained in the system registry under the following Key:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\3.0\Engines\ODBC
```

(or)

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\3.5\Engines\ODBC
```

Which key you update is dependant upon which version of the Jet engine you have installed and are trying to configure.

[If you are using a product such as Microsoft Access ®, (in particular the 16 bit versions), simply changing this value "after-the-fact" may not have any effect on your application. That is because the product (Access, etc…) is storing the parameters that it uses for the query at the time the query is created/modified. For the new parameter to be noticed, you must 'touch' or modify the query so that this gets updated along with the query. The 32 bit versions do not seem to have this issue.]

Entry	Value	Effect
RmtTrace	0	Use asynchronous query execution if possible (default)
	8	Trace ODBC API calls into file "odbcapi.txt"
	16	Force synchronous query execution
	24	Trace ODBC API calls; force asynchronous query execution
TraceSQLMode	1	Trace SQL ® Jet sends to ODBC into file "sqlout.txt"
	0	No Microsoft ® Jet-level SQL tracing (default)
QueryTimeout	S	Cancel queries that don't finish in S seconds (default: 60)
LoginTimeout	S	Cancel log-in attempts that don't finish in S seconds (default: 20) ConnectionTimeout
ConnectionTimeout	S	Close cached connections after S seconds idle time (default: 600)
AsyncRetryInterval	M	Ask server "Is query done?" every M milliseconds (default: 500) AttachCaseSensitive
AttachCaseSensitive	0	Attach to first table matching specified name regardless of case (default)
	1	Attach only to table exactly matching specified name
SnapshotOnly	0	Call SQLStatistics at attach time, to allow dynasets (default)
	1	Don't call SQLStatistics, forces snapshots
AttachableObjects	String	List of server object types to allow attaching to (default: **'TABLE','VIEW','SYSTEM TABLE', 'ALIAS','SYNONYM'**)

K. Reference Materials

Microsoft ® Jet Database Engine 2.0: A User's Overview Microsoft â Jet Database Engine ODBC Connectivity

XXII. Microsoft Access Specific Issues

A. Dealing with tables Linked in an Access (MDB) database

You can link an Oracle table via an ODBC connection into a Microsoft Access ® database, and then subsequently attach to this Access datafile using Jet (from an environment such as Visual basic). This presents some problems as the Jet engine will often become confused when generating the SQL that will ultimately be supplied over this link to the Oracle RDBMS. The ODBC driver itself does not directly control the SQL that will be generated (and in many instances the SQL will be incorrect). The recommended approach would be to attach directly to the Oracle RDBMS via the ODBC driver from the host program (VB, MSVC++, etc…), directly rather than through a link in an Access MDB file. [If you must use this approach then you may need to utilize ODBC traces to analyze and troubleshoot the SQL that Jet is submitting to Oracle and re-organize the queries as needed to cause Jet to generate the SQL correctly.]

B. Microsoft Access Appears to 'Hang' when attaching a table

This can be due to having a large database (or one with an extensive data dictionary), as Jet will default to querying the ENTIRE catalog for ALL objects. If this amounts to millions of rows then Access may appear to hang when trying to display the list of tables. See the following section on the topic of ATTACHABLE OBJECTS to limit the amount of data that Jet is attempting to bring over.

C. Unable to view System tables from inside Access

By default, Access doesn't allow SYSTEM tables to be displayed in the table browser lists. To enable viewing SYSTEM tables, you must enable the SYSTEM OBJECTS checkbox in the Access options menu.

XXIII. Using the Oracle ODBC Driver with MFC

The Oracle driver has several caveats when used with the Microsoft ® Foundation Classes, i.e.::CRecordset() objects.

- The Oracle driver itself supports only a forward scrolling cursor (as does the RDBMS) some versions of MFC define that for a forward scrolling only driver the recordset MUST be also declared as READ_ONLY. This is actually an ASSERT done inside of MFC and is not a restriction of the driver itself. Apparently MFC was written with an assumption of bi-directional cursors. This makes use of the CRecordset() objects a bit cumbersome with some versions of MFC.
- The variables used with DATE/TIME fields may not correctly bind data using Rfx.
- Some versions of the Oracle ODBC driver do not work with certain versions of Visual C++ (VC++) when using the Class Wizard to create a CRecordset() class template for you. This appears to be resolved with the VC++ 4.2 and Oracles ODBC driver 1.15.x or later. Earlier versions would report a syntax error in the table type string, or in some cases with VC++ 1.5x it would result in a GPF. **Note:** The driver would work, but it was necessary to manually create the classes.

XXIV. Dates & Times via ODBC

You have several options when dealing with dates and times using ODBC.

You can use the default ODBC Date formatting:

```
#03-DEC-81#
```

Which is the style that DAO/Jet will use by default.

You can use a TO_DATE() SQL function if you are directly executing a SQL statement to encapsulate whatever date format you want. For example:

```
SELECT TO_CHAR(sysdate,'MM/DD/YYYY') FROM dual
```

This will be brought over then as a character string containing the date format you specified.

You can use an ANSI date format via an ODBC escape:

```
{ d '1981-12-03' }
```

Note that this is enclosed in single quotes and always of the format YYYY-MM-DD.

You can always use the standard Oracle date format in much the same way:

```
{ '03-Dec-1981' }
```

Time may be specified as follows:

```
{ t 'hh:mm:ss' }
```

A timestamp (date including time) can be done with the following format:

```
{ ts '1981-12-03 00:00:00' }
```

XXV. Other ODBC SQL Escapes

In addition to the Date/Time, Outer Join, and Procedure Call escapes mentioned elsewhere in this paper, the following Escapes are also included in the ODBC specification:

A. Interval Escape Sequences

{interval-literal}
(I.E. hours-value [:minutes-value [:seconds-value]])

B. Scalar Function Escapes

{fn** scalar-function**}

XXVI. Stored Procedures with Microsoft ® Visual Basic

A. Program Example

The following is a simple ODBC application that will connect and execute a stored procedure on an Oracle7 Server.
This is the SQL necessary to create the test table in Oracle:

```
create table test_tab (col1 varchar2(100), col1 varchar2(100));
```

This is the code necessary to create the stored procedure in Oracle:

```
create or replace
procedure test_sp(field1 varchar, field2 varchar)
is
begin
```

```
insert into test_tab (col1, col2) values (field1, field2);
end;
```

The following is the Microsoft ® Visual Basic code to execute the stored procedure. Give it any two parameters of any size (up to 100 characters).

```
Sub Command1_Click()
Dim oradb As database
Dim Conn1 As String
Dim sql1, sql2, ret

Conn1$ = "odbc;dsn=alias;uid=scott;pwd=scott;"
Set oradb = OpenDatabase("", False, False, Conn1$)
sql1 = "begin test_sp('one_a','two_a'); end;"
ret = oradb.ExecuteSQL(sql1)
sql1 = "begin test_sp(col2=>'two_b',col1=>'one_b'); end;"
ret = oradb.ExecuteSQL(sql1)
oradb.Close
End Sub
```

This will first insert the two values 'one_a' and 'two_a' into the using default column notation. Second, it will insert 'one_b' and 'two_b' into the database using explicit column notation.

Note: The value 'ret' (in ret = oradb.ExecuteSQL(sql1)) contains an integer representing the number of rows affected by the stored procedure.

There is an alternative method of calling stored procedures that makes use of the CALL specifier. The following is the Microsoft ® Visual Basic syntax for this, oradb is assumed to be a valid database object:

```
oradb.ExecuteSQL("{CALL
procedurename(param1,param2,param3)}")
```

Note: From a tool such as MSQuery just use the {CALL ...} syntax without the double quotes.

If you don't have any parameters, you still must include the (). Also out parameters are supported only at the ODBC Level 2 (Oracle7 ODBC Version 2.x) conformance. The 7.3 Oracle driver does not support returning dynasets, the first version of the Oracle driver to implement this is version 8.0.5.x See the section on stored procedures for alternative methods.

B. Visual Basic error 40041-Object Collection could not find item indicated by text

This usually demonstrates itself on an upgrade of the driver with code that was working previously. This is due to the fact that the stored procedure names are in lower or mixed case. Prior to version 2.5.3.0.1, the ODBC driver did not correctly preserve the case of procedure names (even if they were double quoted), but as of this version it now does. This leads to calls that were previously working now failing. After version 2.5.3.x you MUST specify the procedure name in the appropriate case if the item is double quoted (which is how DAO and RDO will implement the call). This means that {CALL myproc()} does not automatically map to PROCEDURE MYPROC in Oracle, you must use {CALL MYPROC()} instead.

XXVII. RDO with Microsoft ® Visual Basic

Remote Data Objects (RDO) presumes the use of at least an ODBC API Level 2 compliant driver. You must be using version 2.x of the Oracle ODBC driver to insure that RDO functions correctly. Some RDO functions will require, or assume, the capability to utilize bi-directional scrolling cursors on the RDBMS, which the RDBMS, and the ODBC driver, do not natively support. The 8.0.4.0.2 and later versions of the driver do support scrollable cursors by way of the Microsoft Cursor library (the 2.5.3.x versions also have support for the MS Cursor library). If you wish to use RDO you should use that version or later of the Oracle ODBC or a 3rd party driver which supports this functionality. Examples of functions that require scrollable cursors are opening a Dynaset or a Keyset, also updating a table with a LONG or LONG RAW column. There are third party drivers available which emulate this functionality, including the Intersolv version 3.x driver. RDO 1.x has many known compatibility issues, you should use 2.x or later of RDO (patches available from Microsoft web site).

Here is a table of the options available with RDO[13]:

RDO Cursor type	Resultset type	Lock Type
RdUseNone	RdOpenForwardOnly	RcConcurReadOnly
RdUseODBC	RdOpenStatic	RdConcurReadOnly
		RdConcurValues
	RdOpenForwardOnly	RdConcurReadOnly
	RdOpenDynamic	RdConcurReadOnly
		RdConcurValues
	RdOpenKeyset	RdConcurReadOnly
		RdConcurValues
rdUseClientBatch	RdOpenStatic	RdConcurReadOnly
	RdOpenForwardOnly	RdConcurReadOnly
	RdOpenDynamic	RdConcurValues
		RdConcurRowVer
		RdConcurBatch
	RdOpenKeyset	RdConcurValues

[13] 7.x = 2.5.3.1.2 or later versions ONLY, 8.x = 8.0.4.0.3 or later ONLY

		RdConcurRowVer
		RdConcurBatch

It should be noted that users have experienced problems when attempting to do explicit record locking, rather than defaulting to the RDBMS locking behavior (i.e. using SELECT … FOR UPDATE). The FOR UPDATE will be dropped from the SQL by the RDO engine unless server side cursors are used[14].

Here is a snippet of VB code to demonstrate that:

```
Dim Env as rdoEnvironment
Dim Con as rdoConnection
Dim RST as rdoResultset
Dim SQL as String

Set SQL = "SELECT * FROM EMP FOR UPDATE"
Set Env = rdoEngine.rdoEnvironments(0)
Env.CursorDriver = rdUseServer
Set Con = Env.OpenConnection("",rdDriverNoPrompt, False, sConnect)
Set rs = rdoConn.OpenResultset (SQL,rdOpenForwardOnly)
```

This will result in a write lock on the entire EMP table.

To deal with TIMEOUT issues in RDO much as in Jet, you can configure the appropriate timeout values (either login inactivity timeouts or query timeouts). The login timeout can be configured for the RDOEngine object as so:

```
With rdoEngine
      .rdoDefaultLoginTimeout = 120
      .rdoDefaultCursorDriver = rdUseODBC
End with
```

This code snippet demonstrates setting the default LoginTimeout value to 120 seconds. This can also be set at the rdoEnvironment level as well. To adjust the QueryTimeout value you can set this at either the rdoConnection object level or at the resultset level. Here is a Visual Basic code snippet for setting the QueryTimeout at the connection level:

```
Dim ConOBJ as rdoConnection
ConObj.Connect = "DSN=mydsn;UID=scott;PWD=tiger;"
ConObj.QueryTimeout = 120
```

The above code snippet sets the timeout value for this connections queries to 120 seconds.

[14] Same is true when using ADO

XXVIII. IIS, MTS and ADO issues.

ADO is not supported with any of the version 7.x drivers. The first version of the Oracle driver that is supported with ADO is version 8.0.5.1.0. If you require this functionality you may need to use a driver from a 3rd party vendor. The Microsoft ODBC Driver for Oracle driver version 2.7x or later supports ADO as does the Merant (Intersolv) 3.01 or later driver.

MTS is supported by Oracle's 8.1.5.3a.0 ODBC patch kit or later versions of the driver. ORAMTS Patch 8.1.5.0.1 must be installed for the 8.1.5.x ODBC driver to be supported. ODBC 8.1.6 and later comes with the ORAMTS.

Some issues may arise that are specific to IIS. That is to say, the problems only occur in the IIS environment and not when using a tool such as ODBC test. This is usually due to the configuration of the IIS service. Since it runs as a service and not a normal user there are issues that can arise specific to service connections. With IIS 3.x the easiest way to alleviate these issues is to run the WWW publishing service as a user (such as administrator) whose environment is correctly configured. With IIS 4.x, a special user (Internet Guest Account) is configured via the management console (IUSR_xxxxx). This is the user that will be used for Internet connections through the server, so insure that the environment and permissions are correct for this user. For more specifics on configuring IIS, refer to the documentation or the Microsoft Web site.

A. ORA-12641 when connecting via ODBC

This most frequently happens with IIS or services that use ODBC. Edit the SQLNET.ORA file (normally found in the ORACLE_HOME\NETWORK\ADMIN directory) and add or modify the following entry:

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

B. Unable to connect using ASP or IDC (ORA-1017)

Be certain that the directive for the password immediately follows the line specifying the username:

```
Datasource: Oracle7
Username: scott
Password: tiger
```

This is a must for IDC files, due to the way IIS parses them.

C. Numeric columns do not work correctly with IIS/ADO

This is actually an issue with the client RSF (Required Support Files). You must have version 8.0.5 or later of the Oracle client installed. You should also be using version 8.0.5.2.x or later of the ODBC driver.

D. Hang or ORA-12203 when connecting with ADO

Insure that you are using the MS OLEDB provider for ODBC (just omitting the Provider directive from the connection statement will insure this, as it is the default). Microsoft has developed an Oracle OLEDB provider (MSDAORA) which is not supported by Oracle Support. If you are having difficulty with this provider please contact Microsoft Support. Other companies (such as Intersolv) also provide an OLEDB-ODBC bridge product for use with their ODBC drivers. Oracle has provided their own OLEDB Provider starting with release 8.1.6.

E. ODBC Support for Connection Pooling (MTS/IIS)

Support for this feature was added in the 8.0.5.1.0b version of the Oracle ODBC driver and is controlled by the administrator. You should have the version 2.0 or later of the MDAC installed (this will include version 3.510.x of the ODBC Administrator).

XXIX. ODBC API 3.0 specification issues

The first Oracle ODBC driver that uses the new functionality provided in the ODBC 3.0 API specification is the 8.0.4.x driver. There are currently no plans to release an Oracle 2.x driver which implements the functionality added in the 3.0 specification. Although the 2.5.x driver does implement support for a FILE DSN, they are otherwise compliant with the ODBC API 2.5 specification.

A. ODBC version 3.x and Oracle ODBC drivers

The versions 2.0.3 and earlier of the Oracle ODBC driver are certified against the Microsoft Driver Manager 2.5. The new 3.x ODBC Driver Managers from Microsoft will introduce some problems when used with the Oracle ODBC driver 2.0.3 or earlier. The versions 2.5 and later should work with the limitation that they do not implement the extended 3.0 API functionality. The 8.0.4.x. driver is the first to start implementing this extended support.

B. SQLSTATE 08003

Native Error Code: 0
Driver Message:[Oracle][ODBC Oracle Driver]Connection Not open

(See **C.**)

C. SQLSTATE 01000, Native Error code 0

Native Error Code: 0
Driver Message:[Microsoft][ODBC Driver Manager]The Driver returned invalid (or failed to return)SQL_DRIVER_ODBC_VER: %s

Both **B** & **C** are symptoms of attempting to use one of the Oracle version 1.x (level 1 compliant) drivers with the new ODBC version 3.0 driver manager and/or administrator. With the release of the version 3.0 driver manager, Microsoft has removed support for drivers that are not at a minimum compliant with level 2.0 of the ODBC API specification. Since the version 1.x driver report their conformance level as 1, you must install and use a level 2 or better Oracle ODBC driver (i.e. version 2.x or later).

D. File DSN:

Microsoft's answer to having to make registry changes to support a given ODBC DSN (i.e. remember the old ODBC.INI days?). A file DSN simply holds the DSN configuration information in a file (by default a file DSN is created for every existing machine DSN when the 3.0 Admin installs). A file DSN might look like this:

```
[ODBC]
Oracle7
```

Where Oracle7 is the name of the Actual DSN information in the registry. This file will be named:
Oracle7 (not sharable).dsn
By default, and would be placed in the following directory:

```
\Program Files\Common Files\ODBC\Data Sources
```

Shareable FILE DSN support starts with the 2.5.3.1.5 drivers onward.

E. SQLSTATE 08001 using a FILE DSN

Support for FILE DSNs is NOT included in the 2.0.3.x & earlier versions of the Oracle ODBC driver, you must use version 2.5.x or later. Attempting to use a File DSN with an earlier version will generate this error from the ODBC Administrator.

F. MSQuery 97 ONLY uses File DSNs

You appear to have no other option, so manual creation of a file DSN may be needed if you are not using the driver version 2.5.x or later. Create New DSN button will fail with earlier drivers from inside MSQuery, the error you get is:

"Unable to connect to database server. Driver's SQLSetConnectAttr failed."

Looking at the ODBC trace reveals that the actual failure occurs on the call to SQLDriverConnect(). If the File DSN already exists then you may use it to connect from MSQuery and retrieve the data.

G. Manual configuration of a FILE DSN

You cannot use a File DSN with any version of the Oracle ODBC driver prior to **2.5.x** as they did not include the new ODBC API calls[15]. If you must use an older version of the driver, an unsupported option which may work for you is to create the following in the *.dsn file, but is still referring back to a Machine DSN, and is thus not a shareable FILE DSN:

```
[ODBC]
DSN=Oracle7
```

[Where Oracle7 is an existing Machine DSN (System or User)]

H. Excel 97 Hangs when retrieving data

Even after manually configuring a FILE DSN so that MSQuery is able to get to the Oracle ODBC DSN, you are experiencing problems with Excel 'hanging' when it tries to bring the data into a spreadsheet. When you are configuring the query, one of the last dialogs that you see before the query is actually executed is titled "**Returning External Data to Microsoft Excel**". When you are presented with this dialog you must select the **PROPERTIES** button, under the properties **REFRESH CONTROL** you must uncheck (disable) the property called: **Enable Background Refresh.** This is due to a bug with the way Microsoft Excel deals with drivers, which implement Asynchronous Queries (the Oracle 2.0.3 driver does this by default). **Version 2.5.3.x and later of the driver includes a way to forcibly disable this behavior when configuring the DSN, by simply NOT enabling asynchronous support.**

[15] File DSN support was added in the 3.0 ODBC API specification and the support for these new API calls was not present in the driver in this version.

XXX. ODBC 3.5 Specification issues

Oracles 8.0.5.8.0 and above and 8.1.5.5.0 and above drivers are currently written to meet to the 3.51 ODBC API specification. See the online ODBC help for further info.

XXXI. Issues with Crystal Reports (Seagate)

All the information in this section is provided as a service by Oracle. It is obtained by going to:

http://support.seagatesoftware.com/library/kbase.asp

Type 'Oracle ODBC' (without the single quotes) and you will get a list of articles. We do not support this product, but are providing this info as a starting point for troubleshooting. All wording is Seagate's.

Seagate Crystal Reports version	Summary of Text	Seagate Article Number
5.x, 16bit and 32bit	Technical details concerning ODBC driver versions	c2000171
5	Linking Btrieve tables and Oracle tables	c2000485
6	Needs P2sora7.dll (version 7.0.0.45) to get reports from Oracle	c2002184
6 (only?)	Requires an ODBC dsn be a system dsn	c2002954
6	Currently cannot access any complex types in Oracle. Specific reference is made to images, video or sound. Types that will be supported: row, collections, distinct and opaque. Interfacing with datablades, cartridges and extenders are also in development	c2000063
6	Has a limit of 8000 objects that it can report off of	c2001162
6	Can only access Oracle stored procedures using SCR Direct Driver P2sora7.dll. Driver version 6.0.0.20 has a limit of 97 fields. This is fixed in version 7.0.0.46	c2001269
Any	The SCR option to perform asynchronous queries does not work with Oracle because Oracle's ODBC driver does not support asynchronous queries	c2003042
7	Using ODBC or the Oracle Native Driver (OCI?), SCR generates invalid code in a join using UNION and SELECT DISTINCT. It adds the table name in the ORDER BY clause	c2002660
7 (16bit)	Needs Cror704.dll and Cror707.dll to get reports from Oracle7 tables	c2006778
7 (32bit)	Needs Cror709.dll to get reports from Oracle 7.1-7.3 tables	c2006778
7	Support for Oracle Stored Procedures	c2004581
7	Creating a view with a calculated column from an Oracle 7.3 db	c2005140
8	Has problems reporting off an Oracle table with a RAW field using ODBC	c2005140

8	With Oracle ODBC driver 8.1.5.5.0 caused truncation of string values from a table. Re-installing version 8.1.5.0.0 corrected the issue	c2006909

The following table shows the Microsoft or Seagate/Merant dlls involved in ODBC. (Seagate Article number c2001889)

General Dependencies	
For 16-bit ODBC: Odbc.dll Odbccp.dll Odbcint.dll Msvcrt.dll Advapi.dll Kernel.dll Crbas09.dll Crutl09.dll Crflt09.dll	For 32-bit ODBC: Odbc32.dll Odbccp32.dll Odbcint.dll Msvcrt.dll Advapi32.dll Kernel32.dll User32.dll Crbas11.dll Crutl11.dll Crflt11.dll

The following tables give information about the particular Seagate or Merant ODBC drivers: (Seagate Article number c2001889)

Specific Dependencies:	
Seagate Supplied Driver for Oracle (16-bit)	Seagate Supplied Driver for Oracle (32-bit)
Driver: CR Oracle7	Driver: CR Oracle7
Description: INTERSOLV Oracle7 ODBC Driver	Description: INTERSOLV Oracle7 ODBC Driver
Version: 2.50.0000	Version: 3.00.0000
File Name: Cror709.dll	Cror711.dll
Direct Dependent Files: Ociw16.dll	Ociw32.dll
Seagate Supplied Oracle Native (16-bit)	Seagate Supplied Oracle Native (32-bit)
Description: Crystal Reports 32-bit Physical Server DLL for Oracle 7.x	Description: Crystal Reports 32-bit Physical Server DLL for Oracle 7.x
Version: 6.0.0.15	Version: 6.0.0.15
File Name: Pdsora7.dll	P2sora7.dll
Direct Dependent Files: Ociw16.dll	Ociw32.dll
	The following driver is an available download from Seagate for connectivity to Oracle8i. It works with SCR 7 and 8.
	Seagate Supplied Driver for Oracle (32-bit)
	Driver: CR Oracle8
	Description: INTERSOLV Oracle7 ODBC Driver
	Cror815.dll
	Ociw32.dll

A Seagate article detailing connectivity with various Oracle databases:

Oracle Connectivity Support (Seagate Article c2000287)			
	Pre Oracle 7	Oracle 7.x	Oracle 8
Pre 5.0 16-bit Direct	No	No	No
ODBC	Yes	Yes	Yes
32-bit Direct	No	No	No
ODBC	Yes	Yes	Yes
5.0.x.108 16-bit Direct	No	7.2, 7.3	No
ODBC	Yes	Yes	Yes
32-bit Direct	No	Yes	No
ODBC	Yes	Yes	Yes
6.0.x.135 16-bit Direct	No	7.2, 7.3	Yes
ODBC	Yes	Yes	Yes
32-bit Direct	No	Yes	Yes
ODBC	Yes	Yes	Yes
The following is assumed from the last entry in the previous chart:			
32-bit ODBC supported connectivity to Oracle 8.1.5 with Crystal Reports 7 and 8			

As far as can be determined from Seagate's web pages, they only support their ODBC driver for Oracle or an ODBC driver that they supply from Merant (Intersolv). There is no indication from their web pages that they support using Oracle's ODBC drivers for connectivity.

XXXII. Conformance Errors

The following are known conformance errors that have been encountered using the Oracle ODBC drivers, along with the solutions to them.

A. Conformance Error –7751 or -7713 using Access 2.0 or VB 3

Using an old version of the Jet DLL (MSAJT200.DLL) with a newer version of the Oracle ODBC driver causes this error. You must upgrade this DLL from version 2.0 to version 2.5. This upgrade is available from the Microsoft ftp site.

B. Conformance error –7711 when using 16 bit Access (1.0 or 2.0)

This was problem in the original 1.10 version of the driver and has been resolved in any version after 1.11.0.2

C. Conformance error –7711 when using Access 97 (or other 32 bit ODBC application)

This is caused by a bug in the ODBC driver parser (refer to bug# 758536). The problem involves calling a stored procedure with the following syntax:

```
BEGIN
  Myproc();
END;
```

If the syntax of the call is modified to:

```
BEGIN Myproc(); END;
```

It will work correctly.

D. Conformance error –7739 when using tables with Bitmapped Indices

This was a bug in the 8.0.3 ODBC driver, and is fixed in version 8.0.4.x and later of the Oracle driver.

E. Conformance Error -7746 when using tables with Bitmapped Indices.

This was a bug in the 7.x ODBC driver, you must be using version 2.0.3.1.2 or later.

F. Conformance error –7768 when tables contain NULL DATE data using Oracle 8.0 ODBC driver.

Another bug also resolved in the 8.0.3.0.2 version (or later) of the driver

G. Conformance error –7776 using Access 2.0

This was a bug in the way the Oracle ODBC driver was handling invalid date formats (i.e. if the table contained invalid date data). It is resolved in version 2.0.3.1.5 or later.

XXXIII. Miscellaneous Issues

A. A SYSTEM DSN created by Administrator is not visible to other users.

This is an issue with permissions in the NT registry, run the REGEDT32 program as Administrator on the PC in question. Highlight the HKEY_LOCAL_MACHINE\SOFTWARE\ODBC hive in the registry. From the

menu, select SECURITY, then select PERMISSIONS. The user: EVERYONE should be set to SPECIAL ACCESS with the following permissions:
Query Value, Set Value, Create Subkey, Enumerate Subkeys, Notify, Delete, and Read Control. You may alternatively simply give EVERYONE Full Control on this hive.

B. Driver does not show up as available in ODBC Administrator after Installation.

This is usually due to a permission problem in the registry. Open REGEDIT and look in the following location:

HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI

In the ODBC Drivers key (under the ODBCINST.INI key), you should see one or more of the following:

Oracle73 Ver 2.5 = "Installed"
or
Oracle ODBC Driver = "Installed".

In the main ODBCINST.INI key list, you will see a corresponding entry. These entries correspond to the 2.5.3.1.x driver and either the 8.0.5.x driver or the 8.1.x driver. The only way to tell if the 'Oracle ODBC Driver' is referring to the 8.0.5.x or the 8.1.x driver is by looking into the main key and looking at the specified path. The path will indicate the home of the driver.

If you do not have the correct entries, the driver will not show up as installed in the Administrator. Normally the Oracle Installer will create the correct entries for you. Try the following test:

- Create a new Key under the ODBCINST.INI key called TEST
- In the TEST Key create a new string value and assign it a value of TEST
- Close the registry editor
- Restart the registry editor and navigate back to the ODBCINST.INI key. If your changes are still there (and you did not get any errors attempting to create it) then you have the necessary privileges and the ODBC installation failed for another reason. If you do NOT see the changes you made (or you received an error creating it) then you do NOT have the appropriate privileges needed to install the ODBC driver. The solution is to either log on as administrator or in the case of Windows 95/98 insure that your network administrator has granted you the necessary privileges (you may be under a mandatory Profile).

C. SQLSTATE 01000, Native error code 444.

Full error is: Sqlstate 01000 native error 444 driver message invalid or failed to return sql_driver_odbc_ver:03.51.

The Cause of the sqlstate 01000 native error code 444 is that the 8.1.6 ODBC driver conformance level is 3.5.1. The ODBC administrator is at conformance level 3. There is a mismatch of the ODBC Administrator and ODBC driver. Update or re-install your MDAC to 2.5. This install/update can be downloaded from www.microsoft.com/data. Follow the links to mdac 2.5 download.

D. SQLSTATE 01000, Native error code 3121 (i.e. ORA-3121).

This error indicates that the ODBC layer was unable to communicate with the underlying SQL*Net layer. Here are the possible causes for this.

- The ORACLE_HOME\BIN (i.e. ORAWIN\BIN, ORANT\BIN, ORAWIN95\BIN) directory is not in the search path.

  Unlike the Oracle native tools, ODBC is NOT usually running from the ORACLE_HOME\BIN directory, and thus needs to load the underlying required support files via the SYSTEM SEARCH PATH. Failure to find them will result in this error. Use the MSDOS PATH command or the SET command from an MSDOS prompt to verify that the SYSTEM SEARCH PATH includes the ORACLE_HOME\BIN directory appropriate to your platform.

- The version of SQL*Net you have installed is incompatible with the version of the ODBC driver that you have installed.

  See the chart under **V. Supported Driver Versions** which lists the supported ODBC driver and SQL*Net versions.

- When you configured the ODBC DSN you gave it the incorrect information under the SQL*Net Connect String option. The SQL*Net connect string is ***EXACTLY THE SAME CONNECT INFORMATION THAT YOU WOULD GIVE SQL*PLUS TO MAKE THE CONNECTION***. For version 2.x of SQL*Net this is SIMPLY THE SQL*NET ALIAS. Do **NOT** prefix this with a **T: or X:, etc. UNLESS YOU ARE ATTEMPTING TO USE SQL*Net Version 1.x and have that version installed and working.** The help documentation that is distributed with the ODBC driver was unclear on this point in some versions.

E. SQLSTATE IM003, with a system error code of 1157.

See the first bullet under Item **B** above, the system search path not including the ORACLE_HOME\BIN directory also causes this. Required Support Files missing or not installed is yet another reason this might occur. Remember that the driver will be expecting a certain version of the Oracle client software to be installed, if a different version is installed you may also receive this error. (See the chart under V above.)

F. Oracle 8.0 driver does not preserve letter case on Object names

Tables created in 3rd party tools (such as Microsoft Access) which attempted to create table names etc. using mixed letter case would actually be created all Upper Case. This has been resolved in the 8.0.3.0.1 or later versions of the driver.

G. Schema & Table names with Underscores '_' using Oracle 8

This is a known issue in the 8.0.3.x and 8.0.4.x drivers, and is fixed in a later release (8.0.4.0.2 or later). If either the schema name or table names contain the underscore character, you will get errors accessing the object. In the case of the schema name having the underscore, you can actually access the object if logged on as the user, via the ODBC API. You will not be able to access the USER_TABLES (Jet will be unable to access the object as it will prefix the schema name onto the object).

H. Access violation (GPF) connecting using ODBCT32.EXE (or MSQuery) when using Oracle NamesServer (stack related crash or error)

This is a known issue with the SQL*Net Names use – (reference bug# 562551). The problem is that the ODBCT32 application did not have enough application stack space allocated (and this is also likely the issue with MSQuery). This does require that the application (ODBCT32.EXE) be patched to have a larger default stack space, and will be fixed in a later release of the program. The version 8.0.5.1.x and later versions of the driver have been tuned to minimize their stack requirements on the calling application, so this may help with 3rd party applications that experience this problem. The existing bug on the nameserver is requesting that the stack requirements be minimized within this layer itself (this has already been done in the ODBC layer). There will be some minimum stack requirement for the application to implement[16] which will be beyond the control of the ODBC driver and/or the networking layer.

[16] Since the stack size can be very large for Windows 32-bit applications, this should not be a restriction on any WIN32 application developer. For ODBC enabled applications, the application developer should expect that a greater stack space would be needed than normal.

I. OPS$ (OS Authentication) via ODBC

The first step is to insure that OS Authentication is actually setup correctly on the RDBMS and that it does work via some tool such as SQL*Plus. Once this is done you can use OS Authentication via ODBC as follows:
-When connecting programmatically, you simply pass empty strings (“”) in place of the username (UID, and password (PWD) parameters to the SqlDriverConnect (or whichever ODBC connection API call you are making). If you have a default username specified in the DSN configuration, then OPS$ may not work. You should set the default user to be blank (empty). Then when connecting the ODBC driver will automatically authenticate the OS User. If you get presented with a logon dialog, setting the USERNAME and Password fields to blank MAY allow you to connect as an OS user. I say may because with the Oracle 8 driver (as of 8.0.4.0.2) the default user if configured will always override this, although the 7.3 drivers (as of 2.5.x) will allow you to connect this way. You should NOT have a default user configured if you wish to use OS Authentication.

J. No asynchronous support option with Oracle 8 driver

The Oracle 7.x ODBC driver is the last driver to support asynchronous execution of statements. The Oracle 8.0 OCI no longer has support for non-blocking mode (although this is slated to return in a later version of the RDBMS). Since this is the mechanism that was used to implement the asynchronous behavior, this is not available in any driver after 7.x.

K. ODBC Support With FailSafe & Parallel server

Support for these features was added in the 8.0.5.1.0b version of the Oracle ODBC driver.

L. Connecting without a DSN (DSN-less connection)

This functionality was added with the API Level 3 specification, and any Oracle driver version 8.0.4.0.3 and later will be able to use this type of connection. As of 8.0.4.0.3, the SERVER parameter is not yet used, instead the connect information (SQL*Net SERVICE/Alias name) is specified in the DBQ parameter. This may alter with later versions of the driver. Here is a code snippet (VB) demonstrating a connection without using a DSN:

```
Dim en As rdoEnvironment
Dim cn As rdoConnection

Set en = rdoEnvironments(0)
Set cn = en.OpenConnection(dsName:="", Prompt:=rdDriverNoPrompt, _
                           Connect:="uid=scott;pwd=tiger; _
                           driver={Oracle ODBC Driver};" _
```

```
                              & "DBQ=Beq-Local;")

MsgBox cn.Connect

Set RdoRecordset = cn.OpenResultset("select * from dept", _
                                    rdOpenDynamic, rdConcurValues)

While Not RdoRecordset.EOF
    MsgBox RdoRecordset(1)
    RdoRecordset.MoveNext
Wend
```

M. "Type Mismatched" Error when using a Number(11) column type.

This is a bug and is resolved with the 8.0.5.0.0 or later driver.

*N. Column order is incorrect when using a * select*

This is a bug and is resolved in version 8.0.4.4.0 or later of the Oracle ODBC driver.

O. [Oracle][ODBC]Invalid precision value (#0) updating a VARCHAR col. with SPACE or NULL character

Bug# 771026, resolved in version 8.0.5.3.0 or later of the ODBC driver.

P. Pessimistic Locking Support

The problem is that ADO does not support adLockPessimistic in conjunction with client-side cursors. Microsoft Knowledgebase article Q190625 mentions this fact. Make sure to specify adUseServer for the cursor location.

* General Info: ADO, ODBC, & cursors

In ADO, there are two main options for specifying the property "cursor location", "adUseClient" and "adUseServer". Unfortunately, while these names are informative if you're using SQLServer, they're significantly less informative if you're using Oracle. The Oracle ODBC driver does not support server-side cursors, but there are important behavioral differences between specifying adUseServer and adUseClient.

* What adUseClient & adUseServer actually mean:

adUseClient tells the driver to use the cursor library provided by the Microsoft ODBC Driver Manager. adUseServer tells the driver to use its own cursor library.

Because the Microsoft SQLServer drivers use server-side cursors if you instruct them not to use the driver manager's library, Microsoft settled on this naming convention.

* Important distinctions between our cursor library and the driver manager's library

Note: Appendix F of the ODBC Programmer's Reference talks extensively about these and other differences.

1) The driver manager library modifies SQL statements that involve
   - positioned update & delete statements,
   - SELECT FOR UPDATE statements, and
   - batched SQL statements.

The most common problem here, is that the FOR UPDATE clause is deleted by the cursor library. This causes a lot of confusion because the statement succeeds without warnings and should have the effect of locking rows, but the rows are not locked in the database.

2) If you specify the driver manager cursor library, you may cache the data twice on the client. The driver manager cursor library caches data and our driver caches data (potentially). Since our driver isn't aware that Microsoft's cursor library is being used, we can't avoid this double-caching problem.

3) If you specify the driver manager cursor library, you add another layer between the application and the database. Rather than

Client -> Driver Manager -> Our driver -> database

you'll get

Client -> Driver Manager -> MS Cursor Library -> Our driver -> database

which is almost always going to be slower.