# Oracle Migration Workbench for MS Access Reference Guide

Release 1.2.5.0.0 for Windows NT and Windows 95/98

December 1999

Part No. Z26073-02

This reference guide describes how to migrate from MS Access to Oracle8 or Oracle8*i*.

Oracle Migration Workbench for MS Access Reference, Release 1.2.5.0.0

Part No. Z26073-02

Release 1.2.5.0.0

# Contents

## 3 Data Types, Reserved Words, and Functions

## 4 Troubleshooting

## 5 Performance Tuning

## A MS Access Error Messages

## B Code and Query Samples

## Index

# Send Us Your Comments

### Oracle Migration Workbench for MS Access Reference Guide, Release 1.2.5.0.0 for Windows NT and Windows 95/98

**Part No. Z26073-02**

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Email - mwbinfo@ie.oracle.com
- FAX - +353-1-803-1899
- Postal service:  
  Documentation Manager  
  Migration Technology Group  
  Oracle Corporation  
  Eastpoint Business Park  
  Dublin 3  
  Ireland

If you would like a reply, please give your name, address, and telephone number below.

# Preface

The *Oracle Migration Workbench for MS Access Reference Guide* provides detailed information about migrating a database from MS Access 2.0, MS Access 95, or MS Access 97 to Oracle8 or Oracle8*i*. This reference guide describes several differences between MS Access and Oracle, and outlines how those differences are dealt with by Oracle Migration Workbench (Migration Workbench) during the conversion process.

## Audience

This reference guide is intended for anyone who is involved in converting an MS Access database to Oracle using the Migration Workbench.

## What You Should Already Know

You should be familiar with relational database concepts and with the operating system environments under which you are running MS Access and Oracle.

## How the Oracle Migration Workbench for MS Access Reference Guide is Organized

This reference guide is organized as follows:

Chapter 1, "Overview"

Introduces the Migration Workbench and outlines features of this tool.

Chapter 2, "The Migration Process"

Outlines the architecture of MS Access and Oracle, explains how to prepare your MS Access database for migration, and describes how to migrate from MS Access to Oracle using the Migration Workbench.

Chapter 3, "Data Types, Reserved Words, and Functions"

Illustrates the migration of data types and built-in functions from MS Access to Oracle and provides a list of Oracle8*i* reserved words.

Chapter 4, "Troubleshooting"

Provides tips on how to migrate hyperlinks and how to perform a manual migration if necessary

Chapter 5, "Performance Tuning"

Outlines some measures that can be taken to tune your database.

Appendix B, "MS Access Error Messages"

Provides a list of Microsoft Jet database engine errors.

Appendix A, "Code and Query Samples"

Uses code examples to illustrate the migration process.

## How to Use This Reference Guide

Every reader of this reference guide should read Chapter 1, "Overview". That chapter provides an introduction to the concept and terminology of the Migration Workbench.

## Conventions Used in This Reference Guide

The following typographic conventions are used in this reference guide:

*Table 0–1 Typographic Conventions*

| Convention | Description |
|---|---|
| UPPERCASE | Uppercase text indicates case-insensitive filenames or directory names, commands, command keywords, initializing parameters, data types, table names, or object names. Enter text exactly as spelled; it need not be in uppercase. |
| [UPPERCASE] | Key names are represented by uppercase letters enclosed in brackets, as square in [RETURN]. |
| Italicized Characters | Italic type in text indicates the complete names of documents, emphasizes a single word or short phrase, indicates variables, or indicates the first instance of an important word or phrase. |
| Code Examples | Monospace text distinguishes examples of commands and statements from the rest of the text. Monospace text must be entered exactly as shown. Example statements may include punctuation, such as commas or quotation marks. All punctuation in example statements is required. All example statements terminate with a semicolon (;). Depending on the application, a semicolon or other terminator may or may not be required to end a statement. |
| UPPERCASE in Code Examples | Uppercase words in example statements indicate the keywords within Oracle SQL. When you issue statements, however, keywords are not case sensitive. |
| lowercase in Code Examples | Lowercase words in example statements indicate words supplied only for the context of the example. For example, lowercase words may indicate the name of a table, column, or file. |
| Bold | Boldface type in text indicates emphasis with stress, a term defined in the text or the glossary or in both locations, or case-sensitive filenames or directory names. |
| > | Right-facing angle brackets appear in navigation paths to indicate movement from one Web page to another. |
| { } | Curly braces indicate that one of the enclosed arguments is required. Do not enter the braces themselves. |
| [ ] | Square brackets indicate that the enclosed arguments are optional. Do not enter the brackets themselves. |

*Table 0–1 Typographic Conventions*

| Convention | Description |
|---|---|
| \| | A vertical bar separates alternative items that may be optional or required. Do not type the vertical bar. |
| ... | Ellipses indicate that the preceding item can be repeated. You can enter an arbitrary number of similaritems. In code fragments, an ellipsis means that code not relevant to the discussion has been omitted. Do not type the ellipsis. |
| SQL*Plus Prompts | The SQL*Plus prompt, SQL>, appears in SQL statement and SQL*Plus command examples. Enter your response at the prompt. Do not enter the text of the prompt, SQL>, in your response. |
| MS-DOS Prompts | The MS-DOS prompt, >, appears in MS-DOS command examples. Enter your response at the prompt. Do not enter the prompt in your response. |
| Storage Measurements | Storage measurements use these abbreviations:<br>K, for kilobyte which equals 1024 bytes<br>M, for megabyte which equals 1 048 576 bytes<br>G, for gigabyte which equals 1 073 741 824 bytes |

# 1 Overview

This chapter introduces the Oracle Migration Workbench (Migration Workbench) under the following headings:

- Introduction
- Product Description
- Features
- Terminology

## Introduction

Products such as Microsoft Access (MS Access) allow developers and advanced users to build complete business systems. However, because MS Access is based on file sharing technology, it lacks the speed, reliability, and robustness provided by an independent relational database management system (RDBMS) server.

Using an RDBMS server with MS Access in a client/server architecture provides the strengths of both technologies. MS Access provides excellent forms and reports systems, as well as a complete programming language (MS Access Basic). An independent RDBMS provides reliable, robust, and secure high speed data management.

The Oracle RDBMS is a modern, scaleable, high performance database server that can run on a wide range of computers from PCs to mainframes. Oracle operates in a networked, client/server environment and can support tens, hundreds, or thousands of users depending on the server.

This reference guide explains how to migrate from MS Access to Oracle using the Migration Workbench. This reference guide also provides guidelines on how to modify your MS Access applications to work with the new Oracle database. If you have an investment in MS Access applications, you can retain this investment while adding the advanced features of Oracle to your application architecture.

## Product Description

The Migration Workbench is a tool that simplifies the process of migrating your data and applications from an MS Access 2.0, MS Access 95, or MS Access 97 environment to Oracle8 or Oracle8*i*. The Migration Workbench allows you to quickly and easily migrate an entire application system, that is the database schema including validation rules, default values, indexes and relations, in an integrated, visual environment.

The Migration Workbench employs an intuitive and informative User Interface and a series of wizards to simplify the migration process. To ensure portability, all components of the Migration Workbench are written in Java.

The Migration Workbench uses a repository to store migration information. This allows you to query the initial state of the application before migration. By initially loading the migratable components of the application system into a repository, you can work independently of the production application.

Furthermore, the Migration Workbench saves useful dependency information about the components being converted. For example, the Migration Workbench keeps a record of all the tables accessed by a stored procedure. You can then use this information to understand the impact of modifying a given table.

## Features

The Migration Workbench allows you to:

- Migrate a complete MS Access 2.0, MS Access 95, or MS Access 97 database to Oracle8 or Oracle8*i*.
- Migrate tables (including validation rules and default values), indexes, relations, and primary keys.
- Use unmodified MS Access forms and reports with the destination Oracle database via an ODBC connection.
- Migrate MS Access databases with attached MS Access tables.
- Display a representation of the source database and its Oracle equivalent.
- Generate and view a summary report of the migration.
- Customize users, tables, indexes, and tablespaces.

- Customize the default data type mapping rules.
- Create ANSI-compliant names.
- Automatically resolve conflicts such as Oracle reserved words.
- Remove and rename objects in the Oracle Model.
- Migrate individual table data.

# Terminology

The following terms are used to describe the Migration Workbench:

*Application System* is the database schema and application files that have been developed for a database environment other than Oracle. For example, MS Access, MS SQL Server, or Sybase.

*AUTONUMBER* is another name for the MS Access COUNTER data type.

*Capture Wizard* is an intuitive wizard that takes a snapshot of the data dictionary of the source database, loads it into the Source Model, and creates the Oracle Model.

*Migration Wizard* is an intuitive wizard that helps you to migrate the source database to Oracle.

*Migration Component* is part of an application system that can be migrated to an Oracle database. Examples of migration components are tables and stored procedures.

*Migration Entity* is an instance of a migration component. The table EMP would be a migration entity belonging to the table MIGRATION COMPONENT.

*Dependency* is used to define a relationship between two migration entities. For example, a database view is dependent upon the table it references.

*Migration Workbench* is the graphical tool that allows migration of an application system to an Oracle database environment.

*Workbench Repository* is the area in an Oracle database used to store the persistent information necessary for the Migration Workbench to migrate an application system.

*Software Development Kit (SDK)* is a set of well-defined application programming interfaces (APIs) that provide services that a software developer can use.

*Source Database* is the database containing the data dictionary of the application system being migrated by the Migration Workbench. The source database is a database other than Oracle, such as MS Access.

*Destination Database* is the Oracle database to which the Migration Workbench migrates the data dictionary of the source database.

*Source Model* is a series of tables in the Workbench Repository that contain a mirror image of the data dictionary of the source database.

*Oracle Model* is a series of Oracle tables that is created from the information in the Source Model. It is a visual representation of how the source database will look when generated in an Oracle environment.

*Navigator Pane* is the part of the Migration Workbench User Interface that contains the tree views representing the Source Model and the Oracle Model.

*Properties Pane* is the part of the Migration Workbench User Interface that displays the properties of a migration entity that has been selected in one of the tree views in the Navigator Pane.

*Progress Window* is the part of the Migration Workbench User Interface that contains informational, error, or warning messages describing the progress of the migration process.

# 2

# The Migration Process

This chapter introduces the migration process by outlining the architecture of both MS Access and Oracle. It includes the following sections:

- MS Access Architecture
- Oracle Architecture
- Jet/ODBC/Oracle Architecture
- Preparing for Migration
- Changes Made to the MS Access Database
- Extending your Application

## MS Access Architecture

MS Access is based on a file server DBMS technology named "Jet." Forms, reports, and Basic code in MS Access rely on Jet to manage data stored in the native "mdb" file format. This is illustrated in the following diagram:

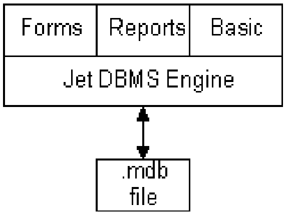

***Figure 2–1 MS Access Architecture***

In a single-user MS Access application, the mdb file and MS Access are located on the same machine. In a multi-user MS Access application, the mdb file is placed on a file server and shared. Each client runs a copy of MS Access and the Jet engine. In this configuration, Jet must move a large amount of data across the network, including whole tables, to complete its query processing.

The Data Access Objects (DAOs) DLL provides a hierarchy of classes to MS Access Basic and Visual Basic. The MS Access 2.0 DAO DLL is DA02016.DLL and the MS Access 95 DAO DLL is DAO350.DLL. DAOs define and expose databases, workspaces, query definitions, parameters, recordsets, tables, fields, indexes, relationships, users, and groups from Jet.

Jet manages links to external data sources. Links to dBase, Paradox, and Btrieve are made through an internal Jet ISAM driver interface. These DLLs are included with MS Access. Jet also supports a link to ODBC, which allows access to a wide range of DBMS servers.

## Jet Recordsets

When Jet processes a query, it returns a recordset (a cursor) for the resulting set of records. Jet supports two types of recordsets; snapshots and dynasets.

A snapshot is a picture of data as it existed at the time the query was run. When returning a snapshot, Jet runs the query to completion, extracts all of the resulting rows and columns into a virtual table, and presents this virtual table to the user. The user of a snapshot can perform a full range of operations on a snapshot such as query the snapshot and base forms and reports on the snapshot. You cannot make changes to snapshot data and it does not reflect changes made by other users after it has been opened.

A dynaset is a live view of the data. When returning a dynaset, Jet extracts the key values from the data and stores them in memory. When a user requests rows of data from the dynaset, Jet fetches the rows of interest by looking them up in the base tables via the internally stored key values. Once you open a dynaset, the set of key values cannot change. Thus, while the data pointed to by the key value may change and is reflected to the user, new rows added after the query is started will not be a part of the set of key values and will not be made visible to the user. Rows that you delete after you run the keyset query are still part of the set of key values, however, they are marked #DELETED# when presented to the user.

The dynaset model is a powerful and flexible model that gives the user of PC-based information the opportunity to browse large quantities of data and update the data at will. When used with local data, dynasets are fast and effective. However, the dynaset model presents one of the key performance challenges when MS Access works with an RDBMS server such as Oracle.

## Jet Multi-User Updates

Jet handles updates by multiple users by using optimistic and pessimistic locks.

Using pessimistic locking, Jet places a hard lock on the data page that contains the row being edited. Other users cannot start editing the locked row until the lock is abandoned or the changes are written to disk.

Jet employs an optimistic locking scheme when working with Oracle. An optimistic locking scheme does not place a hard lock on the source table(s). Instead, when a change is to be committed, Jet checks to make sure that the data has not been altered by another user before allowing the changes to be posted.

## Jet Enforced Referential Integrity

Jet supports declarative referential integrity. This includes primary key/foreign key relationships with one-to-one and one-to-many cardinality with cascading UPDATEs and DELETES.

## Jet Query Processor

The Jet query processor does not support a full implementation of SQL. It optimizes queries, especially when the query references both local and remote tables. Jet can connect to a wide range of data sources and process queries against all of them. Transaction support is limited to native file format database tables. Jet relies on the transaction support of any RDBMS attached via ODBC.

## Jet Transactions

Jet supports an explicit transaction model. Transactions are not started until a `BeginTrans` statement is executed. Transactions are committed with `CommitTrans` and aborted with `Rollback`. In addition to using transactions to group related units of work, developers use transactions to improve performance. If a program makes numerous references to a table, grouping the work in a single transaction will force Jet to perform the operation in memory and then commit all work to disk when the transaction is committed. Be aware that this type of transaction may not map directly to Oracle.

# Oracle Architecture

Oracle8 and Oracle8*i* are powerful, flexible, and scaleable relational database management system (RDBMS) servers, which run on a range of computer systems, from personal computers to the largest mainframes. Oracle has been designed to run effectively in a client/server environment and supports hundreds to thousands of users.

The Oracle architecture supports advanced server features such as record locking with versioning (not page locking as provided by MS Access), advanced query optimization, the PL/SQL programming language, data replication, distributed database management, and other important features.

The architectural features discussed here are only a few of the features found in Oracle and are focused on the elements that pertain to working with MS Access. Refer to the following Oracle Server manuals for a complete description of the Oracle architecture. These manuals can also be found in online format on CD-ROM:

- *Getting to Know Oracle8i, Release 2 (8.1.6)* (Part Number: A76962-01)
- *Oracle8i Concepts, Release 2 (8.1.6)* (Part Number: A76965-01)
- *Oracle8i Administrator's Guide, Release 2 (8.1.6)* (Part Number: A76956-01)
- *PL/SQL User's Guide and Reference, Release 2 (8.1.6)* (Part Number: A77069-01)
- *Oracle8i Error Messages, Release 2 (8.1.6)* (Part Number: A76999-01)

## Triggers and Stored Procedures

Oracle allows you to write and store code in the DBMS along with data. You can associate trigger code with an UPDATE, INSERT, or DELETE event for each row or for a table as a whole. You can also set a trigger to run before or after the event. For example, you can set a trigger to run after any row is updated.

A stored procedure is a general routine (either function or subroutine) that is stored in pre-compiled form on the server. A trigger may call stored procedures, but triggers are only activated by specific database activity such as the insertion of a row in a table.

When using MS Access with Oracle, triggers and stored procedures play a role in mapping the functionality of MS Access to Oracle, such as in the support for the MS Access AUTONUMBER (COUNTER) data type in Oracle.

## PL/SQL Programming Language

The PL/SQL Programming Language is an ALGOL-based language like Pascal. PL/SQL is a modern, full-featured programming language with exception handling. You can use PL/SQL to write stored programs and triggers in Oracle. It is also the programming language used in many of Oracle's client-side tools such as Forms from the Oracle Developer suite of products.

## Sequences

A sequence is a unique number generator that is implemented in shared memory on a server. It is designed to provide a set of unique values for PL/SQL programs for use as primary keys. Sequences are designed for high performance applications that might otherwise 'single-thread' on table-based unique number generators. You use sequences, along with supporting code in a trigger, to emulate the 'COUNTER' field type in MS Access.

## Transactions

Unlike MS Access, Oracle supports an implicit transaction model. Each SQL statement is part of a logical transaction. A logical transaction begins with the first SQL statement and ends with a `Commit` or `Rollback` statement. Immediately after either of these statements, a new transaction takes effect with the next SQL statement.

MS Access developers use transactions to improve the performance of Jet. Grouping database statements in a transaction forces Jet to attempt to complete all database work in memory; Jet defers writing to disk until the transaction is committed. When this use of transactions is mapped to Oracle via Open Database Connectivity (ODBC), Jet sends only the outer most pair of `Begin/Commit` transaction requests. Oracle keeps an open transaction during the entire processing period. You must decide if you want this outcome when you move from MS Access to Oracle.

## Other Oracle Features

A database administrator has great flexibility when configuring Oracle. Data can be written on multiple disks for increased performance, rollback and recovery options can be tuned, and computer resources can be allocated to optimize the configuration for each server. Oracle also supports distributed processing, so data can be distributed across multiple machines. Oracle offers a version of the server called Trusted Oracle Server for applications that require a higher level of user and use authentication.

# Jet/ODBC/Oracle Architecture

Using Oracle with MS Access can increase the robustness and reliability of a multi-user system. This reduces network traffic because only query requests and the resulting data are sent over the network (instead of complete tables). Jet technology is focused on single-user performance with adequate multi-user capabilities; Oracle on the other hand is a mature central server technology focused on multi-user performance, rollback and recovery, and centralized query processing.

Obtaining adequate performance from the combination of MS Access and Oracle requires an understanding of how Jet works with centralized servers.

The following diagram illustrates that MS Access requires ODBC to make its connection to Oracle.

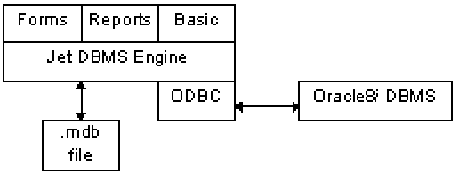

*Figure 2–2 MS Access/ODBC/Oracle Architecture*

ODBC is an API that allows client applications to connect to different RDBMS servers. Jet has been designed to make efficient use of ODBC while requiring a level 1 ODBC driver. When MS Access uses ODBC to connect to remote RDBMS servers, Jet continues to function as the DBMS engine for MS Access. MS Access forms, reports, and Basic code continue to work with Jet as if they were working with local or shared data in the mdb file format. Jet presents remote Oracle tables as attached tables. These attached tables are created at design time and appear to be local tables.

Jet requires a primary key on tables in Oracle in order to support dynasets against those tables. If a remote table does not have a primary key, Jet only opens a non-updateable snapshot on the table.

It is possible to define updateable tables in MS Access that do not have a primary key. When these tables are migrated to Oracle, they do not have a primary key and MS Access is unable to update them. If you need the migrated tables to be updateable, you should ensure that either all updateable MS Access tables have primary keys before you migrate or that you define a primary key once the tables are migrated to Oracle. Any updateable tables that do not have primary keys are flagged with a warning in the Log Window.

## Preparing for Migration

Follow the steps outlined below to prepare your MS Access database for migration. It is imperative that you complete these steps before you run the Capture Source Wizard:

1. Turn off security.
2. If your application contains linked tables, refresh these links by opening the application in the MS Access IDE and choosing Tools -> Add Ins -> Linked Table Manager.
3. Compact your MS Access database files.
4. Back up your MS Access database files.

**Note:** Ensure that the database is not a replica database. The Migration Workbench cannot migrate a replica MS Access database.

# Changes Made to the MS Access Database

When you migrate an MS Access database to Oracle, some changes are required to enable your MS Access forms and reports to continue accessing the data. You must select to have your MS Access database modified in order for these changes to be made.

Figure 2-3 shows an MS Access application before and after migration to Oracle. Both before and after migration, app.mdb contains the forms, reports, macros, and Basic modules that make up the application. Before migration, app.mdb contains an attached table from data.mdb. This diagram uses <*tablename*> to refer to the name of this table.

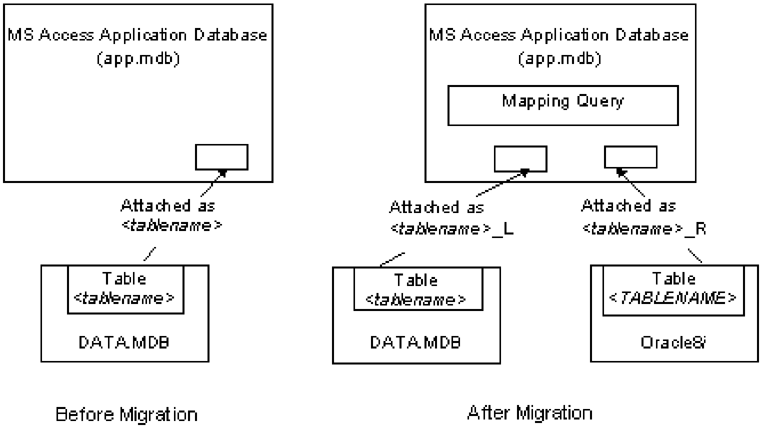

***Figure 2–3 MS Access Application before and after Migration to Oracle***

After migration, app.mdb has two attached tables for each original table and a mapping query. The original MS Access tables are renamed to <*tablename*>_L. The original table is exported to Oracle where it is called <*TABLENAME*>. A table attachment is created to the Oracle table with the name <*tablename*>_R.

Because the forms, reports, and modules in app.mdb are expecting a table with the original table name, a mapping query called <*tablename*> is created. This mapping query takes the place of the original table in the application. The mapping query can refer to either <*tablename*>_L or <*tablename*>_R; you can switch between the local and remote table as you move your application to Oracle. The mapping query also helps resolve reserved word conflicts between MS Access and Oracle. For example, a column named “Sequence” must be renamed as this is an Oracle reserved word. The mapping query can remap this Oracle column back to Sequence for use by the MS Access application. Refer to Chapter 3 for a complete list of Oracle Reserved Words.

All requests for data from the MS Access application are directed to the mapping query, which retrieves the data from the new Oracle table.

The original table that has been exported to Oracle is retained in case the data is needed locally and, more importantly, so that any new MS Access forms created can be based on this table.  This enables a complete set of table properties to be inherited by the MS Access form.  After the MS Access form is defined, the data source can be switched to the mapping query so that the data can be retrieved from Oracle.

If the application opens a table directly (not using a dynaset or snapshot) it will not work with linked tables. This restriction also applies after you move the data to Oracle. If this happens, you might want to leave some tables in the app.mdb file so that each client has an independent copy. This could be appropriate for tables with lookup values such as a ‘State’ table. If you must move a table that is opened directly to the data.mdb file, you must change the application to use dynasets or snapshots.

# Extending your Application

After you move the data management portion of your MS Access application to Oracle, you can rely on Oracle to protect your data and maintain all referential integrity and business rules that you have encoded in PL/SQL.

With this foundation, you can extend your application with MS Access or a wide range of other tools. Oracle offers several high productivity tools such as Oracle Web DB, Oracle Developer, and Oracle Objects for OLE. Oracle Objects for OLE (OO4O) is a high performance connectivity solution for Visual Basic, Delphi, and other client tools that can control OLE Automation Servers.

In addition, if your application grows, you can move your Oracle server to larger computers without changing your application.

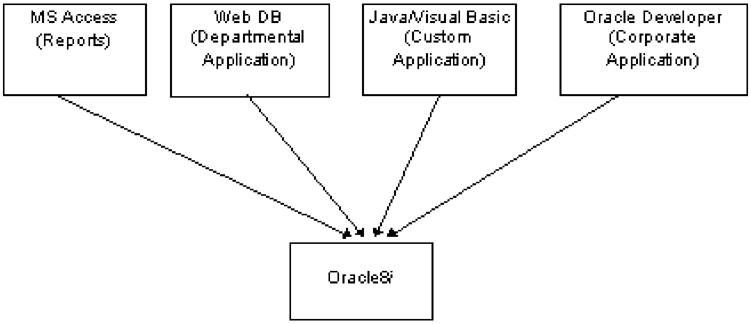
MS Access (Reports)
Web DB (Departmental Application)
Java/Visual Basic (Custom Application)
Oracle Developer (Corporate Application)
Oracle8i

***Figure 2–4   Extending the Application with a Mix of Client Tools***

The *Oracle Objects for OLE and OLEDB/ADO Cookbook* provides additional information about how to extend your application. You can find this document on the Oracle Technology Network (OTN) Web site at the following URL:

http://technet.oracle.com/tech/migration/

# 3

# Data Types, Reserved Words, and Functions

This chapter includes the following sections:

## Oracle Data Types

The following table describes the Oracle data types supported by the Migration Workbench:

*Table 3–1 Oracle Data Types Supported by Oracle Migration Workbench*

| Data Type | Description |
|---|---|
| BLOB | A binary large object. Maximum size is 4 gigabytes. |
| CHAR | Fixed-length character data of length size bytes. Maximum size is 2000 bytes. Default and minimum size is 1 byte. |
| CLOB | A character large object containing single-byte characters. Both fixed-width and variable-width character sets are supported, both using the CHAR database character set. Maximum size is 4 gigabytes. |

*Table 3–1 Oracle Data Types Supported by Oracle Migration Workbench*

| Data Type | Description |
|---|---|
| DATE | The DATE data type stores date and time information. Although date and time information can be represented in both CHAR and NUMBER data types, the DATE data type has special associated properties. For each DATE value, Oracle stores the following information: century, year, month, day, hour, minute, and second. |
| FLOAT | Specifies a floating-point number with decimal precision 38, or binary precision 126. |
| LONG | Character data of variable length up to 2 gigabytes, or 231 -1 bytes. |
| LONG RAW | Raw binary data of variable length up to 2 gigabytes. |
| NCHAR | Fixed-length character data of length size characters or bytes, depending on the choice of national character set. Maximum size is determined by the number of bytes required to store each character, with an upper limit of 2000 bytes. Default and minimum size is 1 character or 1 byte, depending on the character set. |
| NCLOB | A character large object containing multibyte characters. Both fixed-width and variable-width character sets are supported, both using the NCHAR database character set. Maximum size is 4 gigabytes. Stores national character set data. |
| NUMBER | Number having precision p and scale s. The precision p can range from 1 to 38. The scale s can range from -84 to 127. |
| NVARCHAR2 | Variable-length character string having maximum length size characters or bytes, depending on the choice of national character set. Maximum size is determined by the number of bytes required to store each character, with an upper limit of 4000 bytes. You must specify size for NVARCHAR2. |
| RAW | Raw binary data of length size bytes. Maximum size is 2000 bytes. You must specify size for a RAW value. |
| VARCHAR | The VARCHAR data type is currently synonymous with the VARCHAR2 data type. Oracle recommends that you use VARCHAR2 rather than VARCHAR. In the future, VARCHAR might be defined as a separate data type used for variable-length character strings compared with different comparison semantics. |

Refer to *Oracle8i SQL Reference, Release 2 (8.1.6*) (Part Number: A76989-01) for more information about Oracle data types.

# Jet Data Types for MS Access 97

The following table illustrates the Jet data types for MS Access 97:

***Table 3–2 Jet Data Types for MS Access 97***

| Data Type | Description | Min | Max |
|---|---|---|---|
| Text | Stores variable length text | 1 | 255 |
| Memo | Large variable length text | 1 | 64,000 bytes |
| Number Byte | 1 byte storage | 0 | 255 |
| Number Integer | 2 bytes storage | -32,768 | 32,767 |
| Number Long Integer | 4 bytes storage | -2,147,483,648 | 2,147,483,647 |
| Number Single | 4 bytes storage | $-3.4 \times 10^{38}$ | $3.4 \times 10^{38}$ |
| Number Double | 8 bytes storage | $-1.8 \times 10^{308}$ | $1.8 \times 10^{308}$ |
| Currency | 8 bytes storage - monetary values | -922337203685477.5808 | 922337203685477.5808 |
| Counter Yes/No | 4 bytes - AutoIncrement Field 1 bit storage - Boolean Value | 0 | 2,147,483,647 |
| Date/Time | 8 bytes storage | | |
| OLE | OLE, graphics other complex data | 1 | 1.2 gigabytes |
| Binary | Binary Data | 1 | 1.2 gigabytes |

# Default Data Type Mappings

The following table illustrates the default settings used by the Migration Workbench to convert data types from MS Access to Oracle. The Migration Workbench allows you to change the default setting for certain data types by specifying an alternative type. You can do this in the Capture Wizard or in the Data Type Mappings page of the Options dialog box.

Refer to the Oracle Migration Workbench Online Help for more information about changing the default data type mappings.

***Table 3–3 Default Data Type Mappings Used by Oracle Migration Workbench***

| MS Access Data Type | Oracle Data Type |
|---|---|
| Boolean | NUMBER(1, 0) |
| Byte | NUMBER(3, 0) |
| Currency | NUMBER(15, 4) |
| Date | DATE |
| Double | FLOAT |
| Integer | NUMBER(5, 0) |
| Long | NUMBER(11, 0) |
| LongBinary | BLOB |
| Memo | CLOB |
| Single | FLOAT |
| Text | VARCHAR2 |

# Oracle Reserved Words

The following table lists those words that are reserved in Oracle. The Migration Workbench appends an underscore to any object names that conflict with these reserved words.

***Table 3–4 Oracle Reserved Words***

| | | |
|---|---|---|
| ABORT | ACCEPT | ACCESS |
| ADD | ALL | ALTER |
| AND | ANY | ARRAY |
| ARRAYLEN | AS | ASC |
| ASSERT | ASSIGN | AT |
| AUDIT | AUTHORIZATION | AVG |
| BASE_TABLE | BEGIN | BETWEEN |
| BINARY_INTEGER | BODY | BOOLEAN |
| BY | CASE | CHAR |
| CHAR_BASE | CHECK | CLOSE |
| CLUSTER | CLUSTERS | COLAUTH |
| COLUMN | COMMENT | COMMIT |
| COMPRESS | CONNECT | CONSTANT |
| CRASH | CREATE | CURRENT |
| CURRVAL | CURSOR | DATA_BASE |
| DATABASE | DATE | DBA |
| DEBUGOFF | DEBUGON | DECIMAL |
| DECLARE | DEFAULT | DEFINITION |
| DELAY | DELETE | DESC |
| DIGITS | DISPOSE | DISTINCT |
| DO | DROP | ELSE |
| ELSIF | END | ENTRY |
| EXCEPTION | EXCEPTION_INIT | EXCLUSIVE |

*Table 3–4 Oracle Reserved Words*

| | | |
|---|---|---|
| EXISTS | EXIT | FALSE |
| FETCH | FILE | FLOAT |
| FOR | FORM | FROM |
| FUNCTION | GENERIC | GOTO |
| GRANT | GROUP | HAVING |
| IDENTIFIED | IF | IMMEDIATE |
| IN | INCREMENT | INDEX |
| INDEXES | INDICATOR | INITIAL |
| INSERT | INTEGER | INTERFACE |
| INTERSECT | INTO | IS |
| LEVEL | LIKE | LIMITED |
| LOCK | LONG | LOOP |
| MAX | MAXEXTENTS | MIN |
| MINUS | MLSLABEL | MOD |
| MODE | MODIFY | NATURAL |
| NATURALN | NETWORK | NEW |
| NEXTVAL | NOAUDIT | NOCOMPRESS |
| NOT | NOWAIT | NULL |
| NUMBER | NUMBER_BASE | OF |
| OFFLINE | ON | ONLINE |
| OPEN | OPTION | OR |
| ORDER | OTHERS | OUT |
| PACKAGE | PARTITION | PCTFREE |
| PLS_INTEGER | POSITIVE | POSITIVEN |
| PRAGMA | PRIOR | PRIVATE |
| PRIVILEGES | PROCEDURE | PUBLIC |
| RAISE | RANGE | RAW |
| REAL | RECORD | REF |

***Table 3–4 Oracle Reserved Words***

| | | |
|---|---|---|
| RELEASE | REMR | RENAME |
| RESOURCE | RETURN | REVERSE |
| REVOKE | ROLLBACK | ROW |
| ROWID | ROWLABEL | ROWNUM |
| ROWS | ROWTYPE | RUN |
| SAVEPOINT | SCHEMA | SELECT |
| SEPERATE | SESSION | SET |
| SHARE | SIGNTYPE | SIZE |
| SMALLINT | SPACE | SQL |
| SQLCODE | SQLERRM | START |
| STATEMENT | STDDEV | SUBTYPE |
| SUCCESSFUL | SUM | SYNONYM |
| SYSDATE | TABAUTH | TABLE |
| TABLES | TASK | TERMINATE |
| THEN | TO | TRIGGER |
| TRUE | TYPE | UID |
| UNION | UNIQUE | UPDATE |
| USE | USER | VALIDATE |
| VALUES | VARCHAR | VARCHAR2 |
| VARIANCE | VIEW | VIEWS |
| WHEN | WHENEVER | WHERE |
| WHILE | WITH | WORK |
| WRITE | XOR | |

# MS Access Functions

Table-based default values and field validation rules in MS Access can contain MS Access functions and operators. Corresponding functions and operators are generated as triggers in Oracle to support the operation of these default values and field validation rules whenever possible.

The Migration Workbench uses one of the following techniques to convert these functions and operators to Oracle:

| Technique | Description |
|---|---|
| NO ACTION | There is a direct 1:1 mapping between the MS Access function / operator and Oracle. |
| REPLACE IN PLACE | The MS Access function / operator is directly replaced by the equivalent Oracle function / operator. |
| CODE | An Oracle function is written to duplicate the MS Access functionality. This function is created in an Oracle server during the migration. |
| * | This is unhandled which means that you must change your MS Access application so this function is not required in the default value or field validation rule. |

The following is a list of MS Access functions and how they are converted to Oracle:

*Table 3–5 Conversion of MS Access Functions to Oracle*

| MS Access | Oracle | Conversion Action |
|---|---|---|
| Abs | Abs | NO ACTION |
| Asc | Ascii | REPLACE IN PLACE |
| Atn | - | * |
| CCur | To_Number | CODE |
| CDbl | To_Number | CODE |
| Chr | Chr | NO ACTION |
| Chr$ | Chr | NO ACTION |
| CInt | To_Number | CODE |
| CLng | To_Number | CODE |
| Command | - | * |

*Table 3–5 Conversion of MS Access Functions to Oracle*

| MS Access | Oracle | Conversion Action |
|---|---|---|
| Command$ | - | * |
| Cos | COS | NO ACTION |
| CSng | To_Number | REPLACE IN PLACE |
| CStr | To_Char | REPLACE IN PLACE |
| CVar | To_Char | REPLACE IN PLACE |
| CVDate | - | * |
| Date | SYSDATE | REPLACE IN PLACE |
| Date$ | SYSDATE | REPLACE IN PLACE |
| DateAdd | | CODE |
| DateDiff | - | * |
| DatePart | - | * |
| DateSerial | - | * |
| DateValue | - | * |
| To_Date | - | * |
| Day | - | * |
| Environ | - | * |
| Environ$ | - | * |
| Exp | EXP | NO ACTION |
| Fix | Trunc | REPLACE IN PLACE |
| Format | - | * |
| Format$ | - | * |
| Hex | - | * |
| Hex$ | - | * |
| Hour | - | * |
| In | - | CODE (not supported in def) |
| InStr | InStr | NO ACTION |
| Int | Trunc | CODE |
| Is Not Null | | CODE (not supported in def) |
| Is Null | | CODE (not supported in def) |
| IsDate | - | * |
| LCase | LOWER | REPLACE IN PLACE |

***Table 3–5 Conversion of MS Access Functions to Oracle***

| MS Access | Oracle | Conversion Action |
|---|---|---|
| LCase$ | LOWER | REPLACE IN PLACE |
| Left | SUBSTR | CODE |
| Left$ | SUBSTR | CODE |
| Len | LENGTH | REPLACE IN PLACE |
| Like | * | |
| Log | LOG | NO ACTION |
| LTrim | LTRIM | NO ACTION |
| LTrim$ | LTRIM | NO ACTION |
| Mid | SUBSTR | CODE |
| Mid$ | SUBSTR | CODE |
| Minute | - | * |
| Month | - | * |
| Now | SYSDATE | REPLACE IN PLACE |
| Oct | - | * |
| Oct$ | - | * |
| RGB | - | * |
| Right | SUBSTR | CODE |
| Right$ | SUBSTR | CODE |
| Rnd | Rnd | REPLACE IN PLACE |
| RTrim | RTRIM | NO ACTION |
| RTrim$ | RTRIM | NO ACTION |
| Second | - | * |
| Sgn | SIGN | REPLACE IN PLACE |
| Sin | SIN | NO ACTION |
| Space | LPAD/RPAD | CODE |
| Space$ | LPAD/RPAD | CODE |
| Sqr | SQRT | REPLACE IN PLACE |
| Str | TO_Char | REPLACE IN PLACE |
| Str$ | TO_Char | REPLACE IN PLACE |
| StrComp | - | * |
| String | LPAD/RPAD | CODE |

***Table 3–5 Conversion of MS Access Functions to Oracle***

| MS Access | Oracle | Conversion Action |
|---|---|---|
| String$ | LPAD/RPAD | CODE |
| Tan | TAN | NO ACTION |
| Time | SYSDATE | CODE |
| Time$ | SYSDATE | CODE |
| Timer | - | * |
| TimeSerial | - | * |
| TimeValue | - | * |
| Trim | TRIM | NO ACTION |
| Trim$ | TRIM | NO ACTION |
| UCase | UPPER | REPLACE IN PLACE |
| UCase$ | UPPER | REPLACE IN PLACE |
| Val | TO_NUMBER | REPLACE IN PLACE |
| Weekday | - | * |
| Year | - | * |

The following is a list of MS Access operators and how they are converted to Oracle:

***Table 3–6 Conversion of MS Access Operators to Oracle***

| MS Access | Oracle | Conversion Action |
|---|---|---|
| ^ | Power(m,n) | REPLACE IN PLACE |
| & | - | CODE |
| Between | - | CODE |
| Eqv | - | * |
| Imp | - | * |
| Mod | MOD(m,n) | REPLACE IN PLACE |
| Xor | - | * |

# 4

# Troubleshooting

This chapter guides you through a manual conversion of your MS Access database to Oracle should this be necessary. It assumes that you are starting with separate application and data “mdb” files named app.mdb and data.mdb and instructs you how to separate these files prior to performing the manual conversion.

## Separation of Data and Application Files

The following steps outline how to separate your data and application files. They assume that you will start with a file called app.mdb and end with two files called app.mdb and data.mdb.

1. Make a backup of app.mdb.
2. Start MS Access and use the File -> Compact Database menu to compress app.mdb.
3. In File Manager, copy app.mdb to data.mdb.
4. Open data.mdb in MS Access, delete all forms, reports, modules, and macros.
5. In MS Access, open app.mdb and delete all the tables.
6. While still in app.mdb, use the File -> Attach Table menu item to make an attachment in app.mdb to each table in data.mdb.

Now your application should run as it did before you split it into two mdb files.

## Step by Step Manual Migration

The following steps guide you through the manual conversion process:

1. Make backup copies of app.mdb and data.mdb.

2. Ensure that an ODBC connection exists to your Oracle database.
3. In app.mdb, rename each attached table to be suffixed with “_L”. For example, rename <*tablename*> to <*tablename*>_L.
4. Open data.mdb inside MS Access and prepare each table for migration as outlined at the end of this section.
5. Close the data.mdb file.
6. Use File…Compact with data.mdb to recover the space used for the table copies.
7. Open the app.mdb file.
8. Attach each <*TABLENAME*> table from Oracle to app.mdb as follows:
    - Select the File -> Attach Table menu item.
    - Select <SQL Database>.
    - Select your ODBC DSN.
    - Supply your Oracle logon information.
    - Select the table and press the Attach button.
9. Change the names of the attached tables from <*TABLENAME*> to <*tablename*>_R.
10. Create a mapping query for each attached Oracle table. The name of the query will be <*tablename*>, the original name of the table, as seen by your application. Make sure that the name of each column is mapped back to the original name found in the original tables.
11. Open the attached tables in datasheet view or open a form on the tables to make sure that the exporting and mapping steps have been successful. You will notice that you are unable to update the data in the tables. After you complete the migration steps and build primary keys, you will be able to modify your data.
12. Perform the following steps for each column that was derived from an MS Access COUNTER field (COUNTER fields are mapped to NUMBER(10, 0)):
    - Create a sequence for each COUNTER. You can choose any starting number and increment number; however, you must ensure that you do not overlap numbers that have been exported from MS Access. You might want to start the sequence at the next major increment. For example, if 258 is the largest value in a COUNTER field, you might want to start at 1000; this will make it clear which records were inserted after the move to Oracle.

- Create trigger code for the COUNTER field that uses the sequence.
- Make the COUNTER field column the primary key or at least create a unique index on the column; you must insure that this index is the one selected by MS Access as the ‘key value’ index, so you may want to prepend “aaaa” to the name of the index.

13. Create primary key and foreign key definitions in Oracle to match the MS Access structure. Oracle supports declarative CASCADE DELETE relationships, but not CASCADE UPDATEs. To support CASCADE UPDATEs, you need to write trigger code in Oracle.
14. Map default value definitions to Oracle.
15. Map row and table validation statements to Oracle CHECK statements.
16. If you have any tables in Oracle that you need to update from MS Access and which do not have a primary key, you must define one. As in step 12 above, you must make the primary key index the first index in alphabetic order for the table.
17. Finally, use the MS Access Attachment Manager to refresh the attached table connections. This step insures that Jet caches the latest information about primary keys and other table parameters.  After you complete this step you will be able to update your tables.
18. Open the tables in app.mdb in a datasheet or form to ensure that the migration was successful.

## Preparing Tables for Migration

The steps outlined below explain how to prepare your tables for migration.

1. Make a copy of the table, naming the new table <*TABLENAME*>. That is, rename the table name in capital letters.
2. Make sure that the column names follow the requirements for names in Oracle. Change all column names to upper case.
3. Select the File -> Export menu item in MS Access.
4. The instructions that appear in the dialog boxes together with the following six steps outline how to export each table:
    - Select export to <SQL Database>.
    - Select the table to export.

- Name the Oracle table (use <*TABLENAME*>).
- Select your ODBC DSN.
- Supply your logon information. If you have checked for Oracle reserved words, there should be no errors. However, if your tables are large, it could take up to several hours to export each table. MS Access provides a percentage complete indicator while exporting each table.
- Delete <*TABLENAME*> from the data.mdb file.

# 5

# Performance Tuning

This chapter provides suggestions for tuning and customizing the way Oracle and MS Access work together.

## ODBC Driver Selection

There are a number of ODBC drivers available for Oracle. In addition to the driver supplied by Oracle, drivers are also available from Microsoft, Visigenic, Intersolve, and others.

The performance of ODBC drivers can vary. If you are building a large-scale application, you will need to profile the different ODBC drivers with your application. The best way to determine the performance is with ODBC or OCI 'spy' programs. These programs will show you the calls that Jet makes to the ODBC API and will show you the calls the Oracle ODBC driver makes against OCI.

### Server-based Parameter Table

When MS Access first makes a connection to an Oracle database, it checks to see if a special parameter table is present. The table has the following definition:

```
create table MSysConf(
          CONFIG NUMBER,
          nValue INTEGER
          )
```

The following table illustrates how `Config` and `nValue` column values can customize the way MS Access works with Oracle:

| Config | nValue | Meaning |
|---|---|---|
| 101 | 0 | Do not allow the user to store the USERID and PASSWORD in attachments. This is important to set for secure installations. |
| 101 | 1 | Allow the user to store the USERID and PASSWORD in table attachments (default). |
| 102 | D | MS Access delays D seconds between each background chunk fetch when managing dynasets. |
| 103 | N | MS Access fetches N rows on each background chunk fetch when populating a dynaset. |

It is recommended that you create an MSysConf table in each Oracle database even if you plan on using the defaults. It will be easier to change the values in the table than to remember how to create and name the table at a later time.

## Jet’s Query Processor

Jet’s Query Processor includes a cost-based optimizer that can make decisions on joining heterogeneous data via an ODBC connection.

## Application Startup Performance

You can tune MS Access to speed up the process of establishing an ODBC connection at application startup time. In the case of MS Access 2.0, you make entries or changes in the MSACC20.INI file, found in the Windows subdirectory. In MS Access 95, you modify the registry entries under Jet⇒3.0⇒Engines⇒ODBC.

When MS Access opens a connection to an ODBC database, it goes through several steps to determine the level of functionality provided by the particular ODBC driver.

If you are relying on Oracle to provide full security, you can bypass attempts by MS Access to login to Oracle using its own user/group/password information with the following entry:

```
TryJetAuth = 0 (MS Access 2.0)
JetTryAuth = 0 (MS Access 95)
TryJetAuth = 0 (MS Access 97)
```

This step will save between one and two seconds when making the first connection to Oracle.

## Runtime Performance

The most important issue related to runtime performance in a client/server configuration is the reduction of network traffic.

### Form Loading Time

Performance is both perceived and actual. You should eliminate any requests for data from Oracle while a form is loading. You can do this by including a button that will retrieve information. You can also cache the last information that a form displayed locally.

### Qualified and Restricted Queries

You can reduce network traffic by requesting only the columns you need from a table. You should also use the most restrictive qualifications possible to reduce the size of the query result set.

### Snapshots vs. Dynasets

If possible, use Forward Only Snapshots to work with Oracle data, especially when the result set is small. For larger result sets, and for queries which must be updated, use a dynaset. Even if you are not going to update data, a dynaset will be faster than a snapshot if the result set is large. This is because only the key values are retrieved for the dynaset, not the full set of complete rows.

### Drop-Down Lists

MS Access tries to minimize the amount of network traffic when it needs to populate a drop-down list box. When a snapshot is used to populate a drop-down list, MS Access uses the same batch fetching of records that it uses to populate a grid or a form. MS Access will fetch an initial ‘chunk’ of data (100 rows) and then periodically retrieve sets of 100 rows from the server.

This process works smoothly unless you makes an entry that does not match a row already fetched. In this case, MS Access will begin fetching records from the server until a match is found or until all records are retrieved. If the returned set is large, this step can be lengthy and will freeze the User Interface.

MS Access does not share queries for drop-down lists. You cannot define a snapshot query and reference it from multiple list boxes and expect to use information that is retrieved once. Instead, MS Access will treat each ‘activation’ of a query as independent.

If a drop-down list is short (< 100 records), it is probably sufficient to have MS Access perform its normal operations. If the list is long, you may want to build a synchronized shadow table in MS Access. Store the table information locally in MS Access and periodically synchronize the local table with information from Oracle.

# B

# MS Access Error Messages

This appendix presents the complete list of error numbers and error messages returned by the Microsoft Jet database engine. It also indicates the class to which each error belongs.

## Error Message Classes

The following table describes the classes to which the MS Jet error messages belong:

***Table B–1 Error Message Classes***

| Class | Description |
|---|---|
| BTRIEVE | Btrieve installable ISAM-specific errors (MS Jet 2.5 and earlier). |
| DAO | DAO-specific errors. |
| DBASE | dBASE installable ISAM-specific errors. |
| DDL | Data Definition Language-specific errors. |
| EXCEL | MS Excel installable ISAM-specific errors. |
| EXTENDED | Errors that may have extended error information. |
| IMEX | Generic import/export errors. |
| INST ISAM | Generic installable ISAM errors. |
| INTERNET | Internet-specific errors. |
| ISAM | Generic MS Jet ISAM errors. |
| JPM | MS Jet errors related to property management. |
| MISC | MS Jet errors that don’t fit into another category. |
| PARADOX | Paradaox installable ISAM-specific errors. |

*Table B–1 Error Message Classes*

| Class | Description |
|---|---|
| PARSE | MS Jet expression parsing errors. |
| QUERY | MS Jet errors related to queries. |
| REF INTEGRITY | MS Jet errors related to referential integrity. |
| REMOTE | MS Jet errors specific to ODBC. |
| REPLICATOR | MS Jet errors related to replication. |
| SECURITY | MS Jet errors related to security. |
| TEXT | Text installable ISAM-specific errors. |
| TLV | Table-level validation errors. |
| UNUSED | MS Jet errors that are no longer used or that have special meaning. Errors that have special meaning are usually translations from other errors and are not generated in the MS Jet code. |

## MS Jet Error Messages

The following table lists the MS Jet database engine error messages. An asterisk (*) means that there is no MS Jet error message text for a particular error message. An italicized item between angled brackets, such as <*name*>, represents a placeholder for a value that is given when the error message is displayed.

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 2420 | Syntax error in number. | PARSE |
| 2421 | Syntax error in date. | PARSE |
| 2422 | Syntax error in string. | PARSE |
| 2423 | Invalid use of '.', '!', or '()'. | PARSE |
| 2424 | Unknown name. | PARSE |
| 2425 | Unknown function name. | PARSE |
| 2426 | Function isn't available in expressions. | PARSE |
| 2427 | Object has no value. | PARSE |
| 2428 | Invalid arguments used with domain function. | PARSE |

### *Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 2429 | **In** Operator without (). | PARSE |
| 2430 | **Between** operator without **And**. | PARSE |
| 2431 | Syntax error (missing operator). | PARSE |
| 2432 | Syntax error (comma). | PARSE |
| 2433 | Syntax error. | PARSE |
| 2434 | Syntax error (missing operator). | PARSE |
| 2435 | Extra ). | PARSE |
| 2436 | Missing ), ], or Item. | PARSE |
| 2437 | Invalid use of vertical bars. | PARSE |
| 2438 | Syntax error. | PARSE |
| 2439 | Wrong number of arguments with function. | PARSE |
| 2440 | **IIf** function without (). | PARSE |
| 2442 | Invalid use of parentheses. | PARSE |
| 2443 | Invalid use of **Is** operator. | PARSE |
| 2445 | Expression too complex. | PARSE |
| 2446 | Out of memory during calculation. | PARSE |
| 2447 | Invalid use of '.', '!', or '()'. | PARSE |
| 2448 | Can't set value. | PARSE |
| 3000 | Reserved error <*item*>; there is no message for this error. | UNUSED |
| 3001 | Invalid argument. | MISC |
| 3002 | Couldn't start session. | ISAM |
| 3003 | Couldn't start transaction; too many transactions already nested. | ISAM |
| 3004* | | UNUSED |
| 3005 | <*Database name*> isn't a valid database name. | ISAM |
| 3006 | Database <*name*> is exclusively locked. | ISAM |
| 3007 | Can't open library database <*name*>. | ISAM |

***Table B–2 MS Jet Error Messages***

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3008 | The table <*name*> is already opened exclusively by another user, or it is already open through the user interface and cannot be manipulated programatically. | ISAM |
| 3009 | You tried to lock table <*name*> while opening it, but the table can't be locked because it is currently in use. Wait a moment, and then try the operation again. | ISAM |
| 3010 | Table <*name*> already exists. | MISC |
| 3011 | The MS Jet database engine could not find the object <*name*>. Make sure the object exists and that you spell its name and the path name correctly. | MISC |
| 3012 | Object <*name*> already exists. | ISAM |
| 3013 | Couldn't rename installable ISAM file. | ISAM |
| 3014 | Can't open any more tables. | ISAM |
| 3015 | <*Index name*> isn't an index in this table. Look in the **Indexes** collection of the **TableDef** object to determine the valid index names. | ISAM |
| 3016 | Field won't fit in record. | ISAM |
| 3017 | The size of a field is too long. | MISC |
| 3018 | Couldn't find field <*name*>. | MISC |
| 3019 | Operation invalid without a current index. | ISAM |
| 3020 | You tried to call **Update** or **CancelUpdate** or attempted to update a **Field** in a recordset without first calling **AddNew** or **Edit**. | MISC |
| 3021 | No current record. | MISC |

***Table B–2 MS Jet Error Messages***

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3022 | The changes you requested to the table were not successful because they would create duplicate values in the index, primary key, or relationship. Change the data in the field or fields that contain duplicate data, remove the index, or redefine the index to permit duplicate values and try again. | ISAM |
| 3023 | **AddNew** or **Edit** already used. | QUERY |
| 3024 | Couldn’t find file <*name*>. | MISC |
| 3025 | Can’t open any more files. | ISAM |
| 3026 | Not enough space on disk. | ISAM |
| 3027 | Can’t update. Database or object is read-only. | MISC |
| 3028 | Can’t start your application. The workgroup information file is missing or opened exclusively by another user. | ISAM |
| 3029 | Not a valid account name or password. | SECURITY |
| 3030 | <*Account name*> isn’t a valid account name. | SECURITY |
| 3031 | Not a valid password. | SECURITY |
| 3032 | Can’t perform this operation. | SECURITY |
| 3033 | You don’t have the necessary permissions to use the <name> object. Have your system administrator of the person who created this object establish the appropriate permissions for you. | MISC |
| 3034 | You tried to commit or roll back a transaction without first using **BeginTrans**. | ISAM |
| 3035* | | |
| 3036 | Database has reached maximum size. | ISAM |
| 3037 | Can’t open any more tables or queries. | MISC |

***Table B–2 MS Jet Error Messages***

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3038* | | |
| 3039 | Couldn’t create index; too many indexes already defined. | ISAM |
| 3040 | Disk I/O error during read. | ISAM |
| 3041 | Can’t open a database created with a previous version of your application. | ISAM |
| 3042 | Out of MS-DOS file handles. | ISAM |
| 3043 | Disk or network error. | UNUSED |
| 3044 | <*Path name*> isn’t a valid path. Make sure that the path name is spelled correctly and that you are connected to the server on which the file resides. | ISAM |
| 3045 | Couldn’t use <*file name*>; file already in use. | ISAM |
| 3046 | Couldn’t save; currently locked by another user. | ISAM |
| 3047 | Record is too large. | ISAM |
| 3048 | Can’t open any more databases. | ISAM |
| 3049 | Can’t open database <*name*>. It may not be a database that your application recognizes, or the file may be corrupt. | MISC |
| 3050 | Couldn’t lock file. | ISAM |
| 3051 | The MS Jet database engine cannot open the file <*name*>. It is already opened exclusively by another user, or you need permission to view its data. | MISC |
| 3052 | MS-DOS file sharing lock count exceeded. You need to increase the number of locks installed with Share.exe. | ISAM |
| 3053 | Too many client tasks. | MISC |
| 3054 | Too many MEMO or OLE Object fields. | UNUSED |
| 3055 | Not a valid file name. | MISC |
| 3056 | Couldn’t repair this database. | MISC |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3057 | Operation not supported on linked tables. | MISC |
| 3058 | Index or primary key can't contain a **NULL** value. | ISAM |
| 3059 | Operation canceled by user. | MISC |
| 3060 | Wrong data type for parameter <parameter>. | QUERY |
| 3061 | Too few parameters. Expected <*number*>. | EXTENDED |
| 3062 | Duplicate output alias <*name*>. | EXTENDED |
| 3063 | Duplicate output destination <*field name*>. | EXTENDED |
| 3064 | Can't open action query <*name*>. | QUERY |
| 3065 | Can't execute a select query. | QUERY |
| 3066 | Query must have at least one destination field. | EXTENDED |
| 3067 | Query input must contain at least one table or query. | EXTENDED |
| 3068 | Not a valid alias name. | QUERY |
| 3069 | The action query <*name*> cannot be used as a row source. | EXTENDED |
| 3070 | The MS Jet database engine does not recognize <*name*> as a valid field name or expression. | QUERY |
| 3071 | This expression is typed incorrectly, or it is too complex to be evaluated. For example, a numeric expression may contain too many complicated elements. Try simplifying the expression by assigning parts of the expression to variables. | QUERY |
| 3072 | <*Visual Basic error string*>. | EXTENDED |
| 3073 | Operation must use an updateable query. | QUERY |
| 3074 | Can't repeat table name <*name*> in FROM clause. | EXTENDED |
| 3075 | <*Message*> in query expression <*expression*>. | EXTENDED |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3076 | <*Message*> in criteria expression. | EXTENDED |
| 3077 | <*Message*> in expression. | EXTENDED |
| 3078 | The MS Jet database engine cannot find the input table or query <*name*>. Make sure it exists and that its name is spelled correctly. | EXTENDED |
| 3079 | The specified field <*field name*> could refer to more than one table listed in the FROM clause of your SQL statement. | EXTENDED |
| 3080 | Joined table <*name*> not listed in FROM clause. | EXTENDED |
| 3081 | Can't join more than one table with the same name <*name*>. | EXTENDED |
| 3082 | JOIN operation <*operation*> refers to a field that isn't in one of the joined tables. | EXTENDED |
| 3083 | Can't use internal report query. | QUERY |
| 3084 | Can't insert data with action query. | QUERY |
| 3085 | Undefined function <*name*> in expression. | EXTENDED |
| 3086 | Couldn't delete from specified tables. | QUERY |
| 3087 | Too many expressions in GROUP BY clause. | QUERY |
| 3088 | Too many expressions in ORDER BY clause. | QUERY |
| 3089 | Too many expressions in DISTINCT clause. | QUERY |
| 3090 | Resultant table not allowed to have more than one AutoNumber field. | ISAM |
| 3091 | HAVING clause <*clause*> without grouping or aggregation. | UNUSED |
| 3092 | Can't use HAVING clause in TRANSFORM statement. | EXTENDED |

### *Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3093 | ORDER BY clause <*clause*> conflicts with DISTINCT. | EXTENDED |
| 3094 | ORDER BY clause <*clause*> conflicts with GROUP BY clause. | EXTENDED |
| 3095 | Can’t have aggregate function in expression <*expression*>. | EXTENDED |
| 3096 | Can’t have aggregate function in WHERE clause <*clause*>. | EXTENDED |
| 3097 | Can’t have aggregate function in ORDER BY clause <*clause*>. | EXTENDED |
| 3098 | Can’t have aggregate function in GROUP BY clause <*clause*>. | EXTENDED |
| 3099 | Can’t have aggregate function in JOIN operation <*operation*>. | EXTENDED |
| 3100 | Can’t set field <*name*> in join key to **Null**. | EXTENDED |
| 3101 | The MS Jet database engine can’t find a record in the table <*table name*> with key matching field(s) <*field name*>. | EXTENDED |
| 3102 | Circular reference caused by <*query reference*>. | EXTENDED |
| 3103 | Circular reference caused by alias <*name*> in query definition’s SELECT list. | EXTENDED |
| 3104 | Can’t specify fixed column heading <*value*> in a crosstab query more than once. | EXTENDED |
| 3105 | Missing destination field name in SELECT INTO statement <*statement*>. | EXTENDED |
| 3106 | Missing destination field name in UPDATE statement <*statement*>. | EXTENDED |
| 3107 | Record(s) can’t be added; no insert permission on <*name*>. | EXTENDED |
| 3108 | Record(s) can’t be edited; no update permission on <*name*>. | EXTENDED |
| 3109 | Record(s) can’t be deleted; no delete permission on <*name*>. | EXTENDED |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3110 | Couldn't read definitions; no read definitions permission for table or query <*name*>. | EXTENDED |
| 3111 | Couldn't create; no modify design permission for table or query <*name*>. | EXTENDED |
| 3112 | Record(s) can't be read; no read permission on <*name*>. | EXTENDED |
| 3113 | Can't update <*field name*>; field not updateable. | UNUSED |
| 3114 | Can't include MEMO or OLE Object when you select unique values <*statement*>. | EXTENDED |
| 3115 | Can't have MEMO or OLE Object fields in aggregate argument <*statement*>. | EXTENDED |
| 3116 | Can't have MEMO or OLE Object fields in criteria <*statement*> for aggregate function. | EXTENDED |
| 3117 | Can't sort on MEMO or OLE Object <*clause*>. | EXTENDED |
| 3118 | Can't join on MEMO or OLE Object <*name*>. | EXTENDED |
| 3119 | Can't group on MEMO or OLE Object <*clause*>. | EXTENDED |
| 3120 | Can't group on fields selected with '*' <*table name*>. | EXTENDED |
| 3121 | Can't group on fields selected with '*'. | EXTENDED |
| 3122 | You tried to execute a query that doesn't include the specified expression<*name*> as part of an aggregate function. | EXTENDED |
| 3123 | Can't use '*' in crosstab query. | EXTENDED |
| 3124 | Can't input from internal report query <*name*>. | QUERY |

**Table B–2 MS Jet Error Messages**

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3125 | The database engine can't find <*name*>. Make sure it is a valid parameter or alisa name, that it doesn't include characters or punctuation, and that the name isn't too long. | MISC |
| 3126 | Invalid bracketing of name <*name*>. | EXTENDED |
| 3127 | The INSERT INTO statement contains the following unknown field name: <*field name*>. Make sure you've typed the name correctly, and try the operation again. | EXTENDED |
| 3128 | Specify the table containing the records you want to delete. | QUERY |
| 3129 | Invalid SQL statement; expected 'DELETE', 'INSERT', 'PROCEDURE', 'SELECT', or 'UPDATE'. | QUERY |
| 3130 | Syntax error in DELETE statement. | QUERY |
| 3131 | Syntax error in FROM clause. | QUERY |
| 3132 | Syntax error in GROUP BY clause. | QUERY |
| 3133 | Syntax error in HAVING clause. | QUERY |
| 3134 | Syntax error in INSERT INTO statement. | QUERY |
| 3135 | Syntax error in JOIN operation. | QUERY |
| 3136 | The LEVEL clause includes a reserved word or argument that is misspelled or missing, or the punctuation is incorrect. | QUERY |
| 3137 | Missing semicolon (;) at end of SQL statement. | QUERY |
| 3138 | Syntax error in ORDER BY clause. | QUERY |
| 3139 | Syntax error in PARAMETER clause. | QUERY |
| 3140 | Syntax error in PROCEDURE clause. | QUERY |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3141 | The SELECT statement includes a reserved word or an argument name that is misspelled or missing, or the punctuation is incorrect. | QUERY |
| 3142 | Characters found after end of SQL statement. | QUERY |
| 3143 | Syntax error in TRANSFORM statement. | QUERY |
| 3144 | Syntaxt error in UPDATE statement. | QUERY |
| 3145 | Syntax error in WHERE clause. | QUERY |
| 3146 | ODBC - call failed. | UNUSED |
| 3147 | * | UNUSED |
| 3148 | * | UNUSED |
| 3149 | * | UNUSED |
| 3150 | * | UNUSED |
| 3151 | ODBC - connection to <*name*> failed. | EXTENDED |
| 3152 | * | UNUSED |
| 3153 | * | UNUSED |
| 3154 | ODBC - couldn't find DLL <*name*>. | REMOTE |
| 3155 | ODBC - insert on a linked table <*name*> failed. | EXTENDED |
| 3156 | ODBC - delete on a linked table <*name*> failed. | EXTENDED |
| 3157 | ODBC - update on a linked table <*name*> failed. | EXTENDED |
| 3158 | Couldn't save record; currently locked by another user. | INST ISAM |
| 3159 | Not a valid bookmark. | MISC |
| 3160 | Table isn't open. | INST ISAM |
| 3161 | Couldn't decrypt file. | INST ISAM |
| 3162 | You tried to assign the **Null** value to a variable that isn't a **Variant** data type. | MISC |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3163 | The field is too small to accept the amount of data you attempted to add. Try inserting or pasting less data. | MISC |
| 3164 | The field can’t be updated because another user or process has locked the corresponding record or table. | MISC |
| 3165 | Couldn’t open .inf file. | DBASE |
| 3166 | Cannot locate the requested Xbase memo file. | DBASE |
| 3167 | Record is deleted. | MISC |
| 3168 | Invalid .inf file. | DBASE |
| 3169 | The MS Jet database engine could not execute the SQL statement because it contains a field that has an invalid data type. | QUERY |
| 3170 | Couldn’t find installable ISAM. | UNUSED |
| 3171 | Couldn’t find network path or user name. | PARADOX |
| 3172 | Couldn’t open Paradox.net. | UNUSED |
| 3173 | Couldn’t open table ’MSysAccounts’ in the workgroup information file. | SECURITY |
| 3174 | Couldn’t open table ’MSysGroups’ in the workgroup information file. | SECURITY |
| 3175 | Date is out of range or is in an invalid format. | INST ISAM |
| 3176 | Couldn’t open file <*name*>. | IMEX |
| 3177 | Not a valid table name. | IMEX |
| 3178 | * | |
| 3179 | Encountered unexpected end of file. | IMEX |
| 3180 | Couldn’t write to file <*name*>. | IMEX |
| 3181 | Invalid range. | IMEX |
| 3182 | Invalid file format. | IMEX |

***Table B–2 MS Jet Error Messages***

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3183 | Not enough space on temporary disk. | ISAM |
| 3184 | Couldn’t execute query; couldn’t find linked table. | EXTENDED |
| 3185 | SELECT INTO on a remote database tried to produce too many fields. | EXTENDED |
| 3186 | Couldn’t save; currently locked by user <*name*> on machine <*name*>. | EXTENDED |
| 3187 | Couldn’t read; currently locked by user <*name*> on machine <*name*>. | EXTENDED |
| 3188 | Couldn’t update; currently locked by another session on this machine. | ISAM |
| 3189 | Table <*name*> is exclusively locked by user <*name*> on machine <*name*>. | UNUSED |
| 3190 | Too many fields defined. | ISAM |
| 3191 | Can’t define field more than once. | ISAM |
| 3192 | Couldn’t find output table <*name*>. | EXTENDED |
| 3193 | (unknown). | UNUSED |
| 3194 | (unknown). | UNUSED |
| 3195 | (expression). | UNUSED |
| 3196 | The database <*name*> is already in use by another person or process. When the database is available, try the operation again. | ISAM |
| 3197 | The MS Jet database engine stopped the process because you and another user are attempting to change the same data at the same time. | MISC |
| 3198 | Couldn’t start session. Too many sessions already active. | ISAM |
| 3199 | Couldn’t find reference. | REF INTEGRITY |

**Table B–2 MS Jet Error Messages**

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3200 | The record cannot be deleted or changed because table <*name*> includes related records. | EXTENDED |
| 3201 | You can't add or change a record because a related record is required in table <*name*>. | EXTENDED |
| 3202 | Couldn't save; currently locked by another user. | ISAM |
| 3203 | Subqueries cannot be used in the expression <*expression*>. | EXTENDED |
| 3204 | Database already exists. | ISAM |
| 3205 | Too many crosstab column headers <*value*>. | EXTENDED |
| 3206 | Can't create a relationship between a field and itself. | REF INTEGRITY |
| 3207 | Operation not supported on a Paradox table with no primary key. | PARADOX |
| 3208 | Invalid deleted setting in the Xbase key of the Windows registry. | DBASE |
| 3209 | * | UNUSED |
| 3210 | The connection string is too long. | QUERY |
| 3211 | The database engine couldn't lock table <*name*> because it's already in use by another person or process. | EXTENDED |
| 3212 | Couldn't lock table <*name*>; currently in use by user <*name*> on machine <*name*>. | UNUSED |
| 3213 | Invalid Date setting in the Xbase key of the Windows registry. | DBASE |
| 3214 | Invalid Mark setting in the Xbase key of the Windows registry. | DBASE |
| 3215 | Too many Btrieve tasks. | BTRIEVE |
| 3216 | Parameter <*name*> specified where a table name is required. | EXTENDED |
| 3217 | Parameter <*name*> specified where a database name is required. | EXTENDED |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3218 | Couldn't update; currently locked. | ISAM |
| 3219 | Invalid operation. | MISC |
| 3220 | Incorrect collating sequence. | PARADOX |
| 3221 | Invalid settings in the Btrieve key of the Windows registry. | BTRIEVE |
| 3222 | Query can't contain a Database parameter. | QUERY |
| 3223 | <*Parameter name*> is invalid because it is too long or contains invalid characters. | EXTENDED |
| 3224 | Can't read Btrieve data dictionary. | BTRIEVE |
| 3225 | Encountered a record locking deadlock while performing a Btrieve operation. | BTRIEVE |
| 3226 | Errors encountered while using the Btrieve DLL. | BTRIEVE |
| 3227 | Invalid Century setting in the Xbase key of the Windows registry. | DBASE |
| 3228 | Invalid Collating Sequence setting in the Paradox key of the Windows registry. | PARADOX |
| 3229 | Btrieve - can't change field. | BTRIEVE |
| 3230 | Out-of-date Paradox lock file. | PARADOX |
| 3231 | ODBC - field would be too long; data truncated. | REMOTE |
| 3232 | ODBC - couldn't create table. | REMOTE |
| 3233 | * | UNUSED |
| 3234 | ODBC - remote query timeout expired. | REMOTE |
| 3235 | ODBC - data type not supported on server. | REMOTE |
| 3236 | * | UNUSED |
| 3237 | * | UNUSED |
| 3238 | ODBC - data out of range. | REMOTE |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3239 | Too many active users. | ISAM |
| 3240 | Btrieve - missing Btrieve engine. | BTRIEVE |
| 3241 | Btrieve - out of resources. | BTRIEVE |
| 3242 | Invalid reference in SELECT statement. | EXTENDED |
| 3243 | None of the import field names match fields in the appended table. | IMEX |
| 3244 | Can't import password-protected spreadsheet. | IMEX |
| 3245 | Couldn't parse field names from the first row of the import table. | IMEX |
| 3246 | Operation not supported in transactions. | MISC |
| 3247 | ODBC - linked table definition has changed. | REMOTE |
| 3248 | Invalid NetworkAccess setting in the Windows registry. | INST ISAM |
| 3249 | Invalid PageTimeout setting in the Windows registry. | INST ISAM |
| 3250 | Couldn't build key. | ISAM |
| 3251 | Operation is not supported for this type of object. | MISC |
| 3252 | Can't open a form whose underlying query contains a user-defined function that attempts to set or get the form's **RecordsetClone** property. | MISC |
| 3253 | * | UNUSED |
| 3254 | ODBC - can't lock all records. | REMOTE |
| 3255 | * | UNUSED |
| 3256 | Index file not found. | DBASE |
| 3257 | Syntax error in WITH OWNERACCESS OPTION declaration. | QUERY |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3258 | The SQL statement couldn’t be executed because it contains ambiguous outer joins. To force one of the joins to be performed first, create a separate query that performs the first join and then include that query in your SQL statement. | QUERY |
| 3259 | Invalid field data type. | MISC |
| 3260 | Couldn’t update; currently locked by user <*name*> on machine <*name*>. | EXTENDED |
| 3261 | Table <*name*> is exclusively locked by user <*name*> on machine <*name*>. | EXTENDED |
| 3262 | Couldn’t lock table <*name*>; currently in use by user <*name*> on machine <*name*>. | EXTENDED |
| 3263 | Invalid database object. | MISC |
| 3264 | No field defined - cannot append **TableDef** or **Index**. | DAO |
| 3265 | Item not found in this collection. | DAO |
| 3266 | Can’t append a **Field** that is already a part of a **Fields** collection. | DAO |
| 3267 | Property can be set only when the **Field** is part of a **Recordset** object’s **Fields** collection. | DAO |
| 3268 | Can’t set this property once the object is part of a collection. | DAO |
| 3269 | Can’t append an **Index** that is already part of an **Indexes** collection. | DAO |
| 3270 | Property not found. | DAO |
| 3271 | Invalid property value. | DAO |
| 3272 | Object isn’t a collection. | DAO |
| 3273 | Method not applicable for this object. | DAO |
| 3274 | External table isn’t in the expected format. | INST ISAM |

**Table B–2 MS Jet Error Messages**

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3275 | Unexpected error from the external database driver <*error number*>. | INST ISAM |
| 3276 | Invalid database object reference. | MISC |
| 3277 | Can’t have more than 10 fields in an index. | ISAM |
| 3278 | The MS Jet database engine has not been initialized. | MISC |
| 3279 | The MS Jet database engine has already been initialized. | MISC |
| 3280 | Can’t delete a field that is part of an index or is need by the system. | ISAM |
| 3281 | Can’t delete this index or table. It is either the current index or is used in a relationship. | ISAM |
| 3282 | Operation is not supported on a table that contains data. | ISAM |
| 3283 | Primary key already exists. | ISAM |
| 3284 | Index already exists. | ISAM |
| 3285 | Invalid index definition. | ISAM |
| 3286 | Format of memo file doesn’t match specified external database format. | INST ISAM |
| 3287 | Can’t create index on the given field. | ISAM |
| 3288 | Paradox index is not primary. | PARADOX |
| 3289 | Syntax error in CONSTRAINT clause. | DDL |
| 3290 | Syntax error in CREATE TABLE statement. | DDL |
| 3291 | Syntax error in CREATE INDEX statement. | DDL |
| 3292 | Syntax error in field definition. | DDL |
| 3293 | Syntax error in ALTER TABLE statement. | DDL |
| 3294 | Syntax error in DROP INDEX statement. | DDL |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3295 | Syntax error in DROP TABLE or DROP INDEX. | DDL |
| 3296 | Join expression not supported. | MISC |
| 3297 | Couldn't import table or query. No records found, or all records contain errors. | IMEX |
| 3298 | There are several tables with than name. Please specify owner in the format 'owner.table'. | REMOTE |
| 3299 | ODBC Specification Conformance Error <*message*>. Report this error to the developer of your application. | UNUSED |
| 3300 | Can't create a relationship. | REF INTEGRITY |
| 3301 | Can't perform this operation; features in this version are not available in databases with older formats. | MISC |
| 3302 | Can't change a rule while the rules for this table are in use. | TLV |
| 3303 | Can't delete this field. It's part of one or more relationships. | REF INTEGRITY |
| 3304 | You must enter a personal identifier (PID) consisting of at least four and no more than 20 characters and digits. | SECURITY |
| 3305 | Invalid connection string in pass-through query. | REMOTE |
| 3306 | You've written a subquery that can return more than one field without using the EXISTS reserved word in the main query's FROM clause. Revise the SELECT statement of the subquery to request only one field. | QUERY |
| 3307 | The number of columns in the two selected tables or queries of a union query don't match. | QUERY |
| 3308 | Invalid TOP argument in select query. | EXTENDED |

*Table B–2 MS Jet Error Messages*

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3309 | Property setting can’t be larger than 2K. | JPM |
| 3310 | This property isn’t supported for external data sources for for databases created with a previous version of MS Jet. | JPM |
| 3311 | Property specified already exists. | JPM |
| 3312 | Validation rules and default values can’t be placed on system or linked tables. | TLV |
| 3313 | Can’t place this validation expression on this field. | TLV |
| 3314 | The field <*name*> can’t contain a **Null** value because the Required property for this field is set to **True**. Enter a value in this field. | EXTENDED |
| 3315 | Field <*name*> can’t be a zero-lenght string. | EXTENDED |
| 3316 | <*Table-level validation text*>. | EXTENDED |
| 3317 | One or more values are prohibited by the validation rule <*rule*> set for <*field name*>. Enter a value that the expression for this field can accept. | UNUSED |
| 3318 | Values specified in a TOP clause are not allowed in delete queries or reports. | EXTENDED |
| 3319 | Syntax error in union query. | QUERY |
| 3320 | <*Error*> in table-level validation expression. | EXTENDED |
| 3321 | No database specified in connection string or **IN** clause. | REMOTE |
| 3322 | Crosstab query contains one or more invalid fixed column headings. | EXTENDED |
| 3323 | The query cannot be used as a row source. | QUERY |
| 3324 | The query is a DDL query and cannot be used as a row source. | QUERY |

***Table B–2 MS Jet Error Messages***

| Error Number | Microsoft Jet Error Message | Class |
|---|---|---|
| 3325 | Pass-through query with **ReturnsRecords** property set to **True** did not return any records. | REMOTE |
| 3326 | This **Recordset** is not updateable. | EXTENDED |
| 3327 | Field <*name*> is based on an expression and can't be edited. | EXTENDED |
| 3328 | Table <*name*> is read-only. | EXTENDED |
| 3329 | Record in table <*name*> was deleted by another user. | EXTENDED |
| 3330 | Record in table <*name*> is locked by another user. | EXTENDED |
| 3331 | To make changes to this field, first save the record. | EXTENDED |
| 3332 | Can't enter value into blank field on 'one' side of outer join. | EXTENDED |
| 3333 | Records in table <*name*> would have no record on the 'one' side. | EXTENDED |
| 3334 | Can be present only in version 1.0 format. | ISAM |
| 3335 | DeleteOnly called with non-zero cbData. | JPM |

# A

# Code and Query Samples

The following section contains samples that support emulation of the COUNTER data type, CASCADE UPDATE referential integrity, and name mapping queries.

## COUNTER Data Type Emulation

MS Access supports a COUNTER data type. The COUNTER data type provides a monotonically increasing sequence of long integers for a column in a native Jet DBMS file. Oracle supports sequences. Sequences generate a set of numbers that can be used in columns as unique identifiers. An important difference between MS Access COUNTERs and Oracle sequences is that a trigger code is required in Oracle to place a sequence number in a column when a new record is inserted into a table.

When Jet has an attachment to an Oracle table and an Oracle trigger changes or initializes the key values at the time of an insert (not UPDATEs), Jet performs a sequence of queries to retrieve the new key value so that the inserted row can become a member of the dynaset. If Jet has trouble re-selecting the inserted row, the rows appears as #DELETED to the user.

The example below shows how to emulate a COUNTER data type in Oracle. The Oracle table is defined as:

```
CREATE TABLE OTBLCOUNTERTEST(
PK NUMBER(10,0),
NAME VARCHAR2(50),
CONSTRAINT PK_OTBLCOUNTERTEST PRIMARY KEY (PK))
```

An Oracle sequence is defined as:

```
CREATE SEQUENCE   TEST     INCREMENT BY 1    START WITH 1000
```

The trigger code is indicated below:

```
Create Trigger TRG_CNT_OTBLCOUNTERTEST
Before INSERT OR UPDATE on OTBLCOUNTERTEST
FOR EACH ROW
DECLARE
   iCounter SCOTT.OTBLCOUNTERTEST.PRIMARYKEY%TYPE;
   cannot_change_counter EXCEPTION;

BEGIN
   IF INSERTING THEN
      SELECT TEST.NEXTVAL into iCounter FROM dual;
      :new.PRIMARYKEY := iCounter;
   END IF; -- End of Inserting Code

   IF UPDATING THEN
      -- Do not allow the PK to be changed.
      IF NOT(:new.PRIMARYKEY = :old.PRIMARYKEY) THEN
         RAISE cannot_change_counter;
      END IF;

   END IF; -- End of Updating Code

EXCEPTION
   WHEN cannot_change_counter THEN
      raise_application_error(-20000,'Cannot Change Counter Value');
END;
```

This trigger emulates the COUNTER data type by trapping both INSERT and UPDATE operations on a table. On any insert the trigger will get the next value in the sequence “TEST” for the PRIMARYKEY column. On UPDATEs, the trigger checks to see if the user is trying to update the COUNTER; if so, an exception is raised and the error is passed back to MS Access.

It is not recommended to silently protect the COUNTER on UPDATE. For example, with the following code, Jet becomes confused in its management of the dynaset and produces strange results:

```
IF UPDATING THEN
   -- Do not allow the PK to be changed.

   IF NOT(:new.PRIMARYKEY = :old.PRIMARYKEY) THEN
      :new.PRIMARYKEY := :old.PRIMARYKEY);
   END IF;

END IF; -- End of Updating Code
```

As a possible enhancement to strict COUNTER field emulation, you could use the following code in the trigger to allow MS Access to pass a value for the COUNTER on a row insert:

```
IF INSERTING THEN
   IF (:new.PRIMARYKEY IS NULL) THEN
      SELECT test.NEXTVAL into iCounter FROM dual;
      :new.PRIMARYKEY := iCounter;
   END IF:
END IF; -- End of Inserting Code
```

This code will generate a new COUNTER value only if the passed value is NULL.

## Name Mapping Query

To begin building a name mapping query in MS Access, use either the QBE or SQL window to define the query. In this example, the original MS Access table is called SeqDateTable and is exported to Oracle as O_SEQDATETABLE. After the export, the table is attached to Jet as R_SeqDateTable.

When the following query is saved as SeqDateTable, it will take the place of the original table and complete the mapping to Oracle. The query maps the column names PRIMARYKEY, O_SEQUENCE and FIRSTDATE to PrimaryKey, Sequence and FirstDate for use by MS Access.

```
SELECT NameMapper.PRIMARYKEY AS PrimaryKey,
NameMapper.O_SEQUENCE AS Sequence,
NameMapper.FIRSTDATE AS FirstDate
FROM R_SEQDATETABLE;
```

## Default Values

Oracle supports declarative default values. However, when moving an application from MS Access to Oracle, you may encounter situations where you need an insert trigger to support defaults. A reasonable design decision is to move all default processing to triggers to centralize the code and reduce maintenance complexity. The following code sample demonstrates supporting default values in a trigger:

```
CREATE OR REPLACE TRIGGER BIU_M2
 BEFORE INSERT OR UPDATE
 ON M2
 FOR EACH ROW

   BEGIN
      IF INSERTING THEN
      /* Manage Default Values if a new value is NULL */
         IF :new.Address IS NULL THEN
            :new.Address := 'Default';
         END IF;
      END IF; -- End of Inserting Code
   END; -- Trigger BI_M2
```

## Column and Table Validation

Oracle supports CHECK statements, which can be used to enforce table and column constraints. However, when moving an application from MS Access to Oracle, you may encounter situations where you need an insert trigger to support validation. The following code sample demonstrates supporting validation in a trigger. Notice that <`Access Validation Code`> indicates where you can insert the validation code from an MS Access Application.

```
CREATE OR REPLACE TRIGGER BIU_M2
 BEFORE INSERT OR UPDATE
 ON M2
 FOR EACH ROW

 BEGIN
      -- Validation Code
      IF NOT ( <Access Validation Code > ) THEN
         raise_application_error (-20000, '<Access Error Message>');
      END IF;
END; -- Trigger BI_M2
```

## CASCADE UPDATE Trigger Code

Oracle does not provide direct support for CASCADE UPDATE referential integrity constraints. CASCADE UPDATE support means that when a primary key is changed, that change is made to all associated foreign keys in linked tables. CASCADE UPDATE is not a common design feature in applications. Primary keys are supposed to be stable, usually for the life of an application.

The following code example is based on two tables:

```
create table M1 (
f1 number,
f2 number,
f3 number )
create table M2 (f1 number,
f2 number,
f3 number )
alter table M1 add primary key (f1)
alter table M2 add primary key (f1)
```

This definition will support one-to-many cardinality. To add support for one-to-one cardinality add the following:

```
alter table M1 add constraint uq_M1_001 unique (f2, f3)
alter table M2 add constraint uq_M2_001 unique (f2, f3)
```

The following code implements CASCADE UPDATE code for the two tables, M1 and M2. Note that this example uses two columns in the primary/foreign key relationships. This relationship is more complex than most and is used to fully illustrate the proper code.

Please note that declarative and procedural support for referential integrity cannot coexist between two tables. To support CASCADE UPDATE between two tables, all declarative primary/foreign key relationships and referential integrity between the tables must be removed and supported instead with procedural code. This is outlined in the following code sample:

```
CREATE OR REPLACE PACKAGE P_M1 AS
    fire_trigger boolean := TRUE;
 END P_M1;

 CREATE OR REPLACE PACKAGE P_M2 AS
    fire_trigger boolean := TRUE;
 END P_M2;

 CREATE OR REPLACE PACKAGE UQ_M1_M2 AS

 PROCEDURE cascade_update (
           o_F2    IN  number,
           o_F3    IN  number,
           n_F2    IN  number,
          n_F3    IN  number,
          bResult OUT boolean );

       PROCEDURE cascade_delete (
          F2      IN  number,
          F3      IN  number,
          bResult OUT boolean );

       FUNCTION pk_exists (
          F2      IN  number,
          F3      IN  number) RETURN boolean;

       FUNCTION fk_exists (
          F2      IN  number,
          F3      IN  number) RETURN boolean;

 END  UQ_M1_M2;

CREATE OR REPLACE PACKAGE BODY UQ_M1_M2 AS

   /* Procedure cascade_update is called when field(s) */
   /*   F2 or                                          */
   /*   F3                                             */
   /*   are changed in table M1.                       */
   /*   The changes are cascaded in table M2           */

   PROCEDURE cascade_update (
             o_F2    IN  number,
             o_F3    IN  number,
             n_F2    IN  number,
             n_F3    IN  number,
             bResult OUT boolean ) IS

   CURSOR d_cur (n1 number, n2 number) IS
      SELECT * FROM m2
      WHERE f2 = n1 AND f3 = n2
      FOR UPDATE of f2, f3;

   BEGIN

      FOR d_cur_rec IN d_cur ( o_F2, o_F3 )
      LOOP
         UPDATE M2 SET f2 = n_F2, f3 = n_F3
            WHERE CURRENT OF d_cur;
      END LOOP; -- Detail Record Loop
   bResult := true;

END cascade_update;

/* Procedure cascade_delete is called when a record */
/*   in M1 is being deleted and associated          */
/*   child records in M2 must also be deleted.      */

PROCEDURE cascade_delete (
        F2      IN  number,
        F3      IN  number,
        bResult OUT boolean ) IS

CURSOR d_cur (n1 number, n2 number) IS
   SELECT * FROM m2
   WHERE f2 = n1 AND f3 = n2
   FOR UPDATE;

BEGIN

   FOR d_cur_rec IN d_cur ( F2, F3 )
   LOOP
      DELETE FROM M2
         WHERE CURRENT OF d_cur;
   END LOOP; -- Detail Record Loop

   bResult := true;

END cascade_delete;

/* Procedure pk_exists is called to determine is a given
   primary key exists in table M1                          */

FUNCTION pk_exists (
     F2      IN  number,
     F3      IN  number) RETURN boolean IS

l_F2        number;
l_F3        number;
bResult     boolean;

CURSOR p_cur (n1 number, n2 number) IS
   SELECT F2, F3 FROM m1
   WHERE f2 = n1 AND f3 = n2;
```

```
BEGIN
   OPEN p_cur( F2, F3 );
   FETCH p_cur INTO l_F2, l_F3;
   IF p_cur%NOTFOUND THEN
      bResult := false;
   ELSE
      bResult := true;
   END IF;

   CLOSE p_cur;

   RETURN( bResult );

END pk_exists;

/* Procedure pk_exists is called to determine is a given
   primary key exists in table M1                           */

FUNCTION fk_exists (
     F2      IN  number,
     F3      IN  number) RETURN boolean IS

l_F2         number;
l_F3         number;
bResult      boolean;

CURSOR d_cur (n1 number, n2 number) IS
   SELECT F2, F3 FROM m2
   WHERE f2 = n1 AND f3 = n2;

BEGIN
   OPEN d_cur( F2, F3 );
   FETCH d_cur INTO l_F2, l_F3;
   IF d_cur%NOTFOUND THEN
      bResult := false;
   ELSE
      bResult := true;
   END IF;

   CLOSE d_cur;

   RETURN( bResult );

END fk_exists;
END UQ_M1_M2;


CREATE OR REPLACE TRIGGER AUD_M1
AFTER UPDATE OR DELETE
ON M1
FOR EACH ROW

  DECLARE
  bResult_OK      BOOLEAN;
  bCascadeDeletes BOOLEAN  := TRUE;

  BEGIN

     IF UPDATING THEN
        IF (:old.F2 <> :new.F2) OR (:old.F3 <> :new.F3) THEN
           P_M2.fire_trigger := FALSE;
           UQ_M1_M2.cascade_update( :old.F2, :old.F3, :new.F2, :new.F3,
                                    bResult_OK );
           P_M2.fire_trigger := TRUE;
        END IF;
     END IF; -- End of Updating Code

     IF DELETING THEN
        IF bCascadeDeletes THEN
           UQ_M1_M2.cascade_delete( :old.F2, :old.F3, bResult_OK );
        ELSE
           IF UQ_M1_M2.fk_exists( :old.F2, :old.F3 ) THEN
              raise_application_error( -20000, 'Rows exist in child table');
           END IF;
        END IF;
     END IF; -- End of Deleting Code

  END; -- Trigger AUD_M1


CREATE OR REPLACE TRIGGER AIU_M2
AFTER INSERT OR UPDATE
ON M2
FOR EACH ROW

  DECLARE
  bResult_OK    BOOLEAN;
```

```
   BEGIN

      IF INSERTING THEN
         IF NOT( UQ_M1_M2.pk_exists( :new.F2, :new.F3 ) ) THEN
            raise_application_error (-20000, 'No corresponding row in parent
                       table');
         END IF;
      END IF; -- End of Inserting Code

      IF ( UPDATING AND P_M2.fire_trigger ) THEN
         IF NOT( UQ_M1_M2.pk_exists( :new.F2, :new.F3 ) ) THEN
            raise_application_error (-20000, 'No corresponding row in parent
table');
         END IF;
      END IF; -- End of Updating Code

   END; -- Trigger AUD_M2
```

# Index