

Microsoft Dynamics™ NAV 5.00

Microsoft Dynamics™ NAV ODBC Driver 5.0 Guide

Microsoft Dynamics™ NAV ODBC Driver 5.0 Guide

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2007 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, Windows, Windows Server, Windows Vista, Application Server for Microsoft Dynamics NAV, AssistButton, C/AL, C/Front, C/Side, FlowField, FlowFilter, C/Side Database Server for Microsoft Dynamics NAV, Microsoft Business Solutions–Navision, Microsoft Dynamics NAV, Microsoft Dynamics NAV Debugger, Navision, NAV ODBC, SIFT, SIFTWARE, SQL Server, SumIndex, SumIndexField are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are property of their respective owners.

PREFACE

This book is a manual for Microsoft Dynamics™ NAV. This book is part of a comprehensive set of documentation and Help materials for the Dynamics NAV enterprise business solution.

The manual describes how to install and run Microsoft Dynamics NAV ODBC Driver 5.0. It also contains a reference guide to SQL statements.

You should also be familiar with the symbols and typographical conventions used in the Dynamics NAV manuals. In the list below, you can see how various elements of the program are distinguished by special typefaces and symbols:

Appearance	Element
CTRL	Keys on the keyboard. They are written in small capitals.
Address	Field names. They appear in bold and start with a capital letter.
<i>Department</i>	Names of windows, boxes and tabs. They appear in bold italics and start with a capital letter.
<i>Hansen</i>	Text that you must enter, for example: "...enter Yes in this field." It is written in italics.
fin.flf	File names. They are written with the Courier font and lowercase letters.
↑ ↓ ▼ *▶ ...	The special symbols that can be seen in the windows on the screen.

TABLE OF CONTENTS

Chapter 1 Introduction	1
What is the Dynamics NAV ODBC Driver?	2
Chapter 2 Installation and Configuration	3
Installation	4
Configuration	5
Chapter 3 Technical Documentation	13
Overview	14
Establishing a Connection	15
Microsoft Dynamics NAV ODBC Driver Functionality	17
Data Types	20
Multilanguage Functionality	24
Appendix A SQL Statement Reference Guide	27
Introduction to the Reference Guide	28
SELECT Statement	30
Parameter Markers	32
Keys and Performance	33
Predicates in WHERE Clauses	34
GROUP BY Clause	36
HAVING Clause	37
ORDER BY Clause	38
INSERT Statement	39
DELETE Statement	41
UPDATE Statement	42
CREATE TABLE Statement	43
DROP TABLE Statement	45
Microsoft Dynamics NAV FlowFields	46

Chapter 1

Introduction

This chapter gives a brief overview of this manual and the Dynamics NAV ODBC Driver, including definitions of technical terms used when working with the Dynamics NAV ODBC Driver.

The chapter contains the following section:

- What is the Dynamics NAV ODBC Driver?

1.1 What is the Dynamics NAV ODBC Driver?

The Dynamics NAV ODBC Driver is the implementation of Open Database Connectivity (ODBC) for Microsoft Dynamics™ NAV. The Dynamics NAV ODBC Driver lets you transfer data between a Dynamics NAV database and any program that supports ODBC.

Dynamics NAV ODBC functions largely in the same way as ordinary clients in Dynamics NAV.

The main differences are:

- a Dynamics NAV ODBC client is a program separate from Dynamics NAV, such as a spreadsheet or word processing program.
- triggers are not run when a Dynamics NAV ODBC client writes data in a Dynamics NAV database.

An explanation of how the Dynamics NAV ODBC Driver works with a Dynamics NAV database can be found on page 13.

Special Terminology

There are a number of terms used in this book that are not used in Dynamics NAV. The following is a short explanation of these terms. If you need a more detailed explanation, see the documentation for the product in which the term is used:

Term	Program	Definition
SQL	Microsoft Excel and Microsoft Query	Structured Query Language: a programming language that is specially designed for queries in databases.
Add-in	Microsoft Excel	A command or function that gives a program additional capabilities.
Visual Basic	Microsoft Excel	A programming language used for programming macros in Microsoft Excel, among other things.
Criteria	Microsoft Query	The same as a filter in Dynamics NAV.

Chapter 2

Installation and Configuration

This chapter explains how to install and set up the Dynamics NAV ODBC Driver.

The chapter contains the following sections:

- Installation
- Configuration

2.1 Installation

To install the Dynamics NAV ODBC Driver on Windows XP or Windows Server 2003, follow this procedure:

- 1 Start the Dynamics NAV ODBC Driver setup program. You find this in the NODBC subfolder on the Dynamics NAV product DVD.

The **Welcome** window appears. This is the first of three windows in a standard Windows Installer wizard.

- 2 Click Next to continue and follow the instructions in the wizard.

The installer copies the necessary files to the (Program Files\Common Files\Dynamics NAV\ODBC) folder. The installer then registers the Dynamics NAV ODBC Driver and creates a sample Dynamics NAV ODBC data source.

Cancelling the Installation

You can cancel the installation at any time. If you choose to cancel the installation, a dialog box appears asking you to confirm your decision. If you click No, the installation process will continue. If you click Yes, Windows Installer will perform a full rollback and restore the computer to the state it was in before the installation process began.

Uninstalling and Repairing the Dynamics NAV ODBC Driver

You can also use Windows Installer to repair or remove the Dynamics NAV ODBC Driver.

- 1 In the Control Panel, select Add/Remove Programs.
- 2 Select Microsoft Dynamics NAV 5.0 ODBC.
- 3 Click the Change button to change your installation of the Dynamics NAV ODBC Driver, or click the Remove button to remove the driver from your system.

Windows Installer will repair or remove the Dynamics NAV ODBC Driver automatically, depending on your choice.

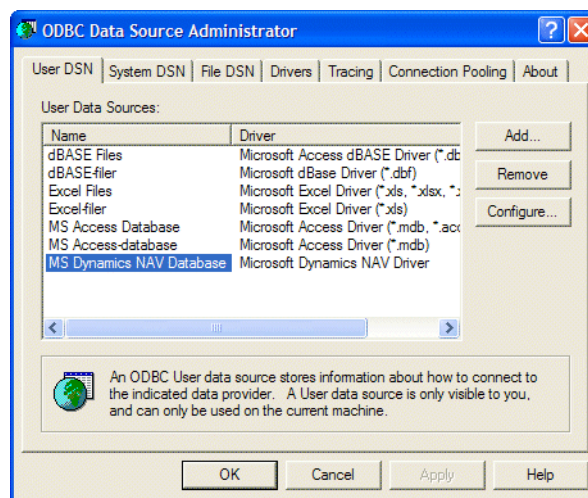
2.2 Configuration

After you have installed the driver, you can set up the sample data source that has been created by the installation program, and you can add new data sources. This is done from the Control Panel.

Setting Up a Data Source

To set up a user data source, follow this procedure:

- 1 In the Control Panel, click Administrative Tools, Data Sources (ODBC). The following window appears, displaying a list of the data sources that are available on your system:



A data source contains information about where to find the data and how the driver formats the data when it is returned to an application. Each data source is identified by a unique name followed by the name of the driver. You can read about adding, changing and deleting data sources on page 10, and the online Help for the ODBC Data Source Administrator explains about User, System and File Data Sources.

64 Bit

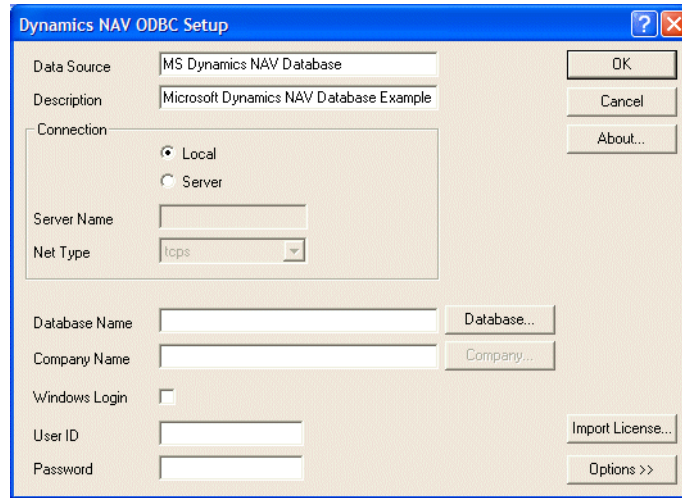
You can only run NAV ODBC on 64 bit systems that support Window on Windows (WOW).

To set up a data source on a 64 bit operating system, you **must** click Start, Run and enter the following command in the **Open** field and click OK.

```
%systemroot%\SysWOW64\odbcad32.exe
```

- 2 In the **ODBC Data Source Administrator** window that appears, select the Dynamics NAV database that you want to use and click Configure.

- 3 Click Dynamics NAV Database, and click Configure. The **Dynamics NAV ODBC Setup** window appears:



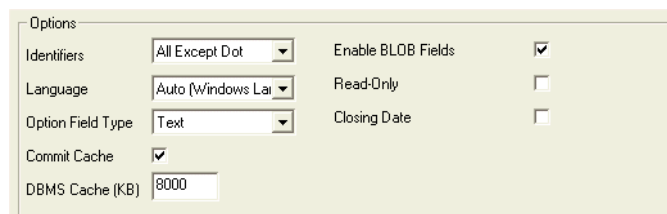
Fill in the fields according to these guidelines:

Field	Comments
Data Source	The field is already filled in with a default data source name. You can change this name, as explained on page 10.
Description	This field contains a description of the data source.
Connection	<p>You must specify here whether Dynamics NAV is installed as a single-user or multiuser system (client/server):</p> <p><i>Local</i> Click here if the driver will function in a single-user installation.</p> <p><i>Server</i> Click here if the driver will function in a multiuser installation.</p> <p>The default value is Local.</p>
Server Name	If you have selected a server connection in the Connection field, enter the name of the server, that is, the server where the Dynamics NAV database is located. (If you have selected a local connection, leave this field empty.)
Net Type	<p>In a multiuser installation, enter the name of the network protocol program, that is, <i>tcps</i> (default value). <i>tcp</i> (for TCP/IP) or <i>netb</i> for NetBios.</p> <p>TCPS is a secure version of TCP/IP and uses the Security Support Provider Interface (SSPI) with encryption enabled and Kerberos authentication. We recommend that you use the TCPS protocol.</p>
Database Name	Enter the name of the database you want to connect to. You can see a list of available databases by clicking Database. Browse to the relevant folder, click the database file name, and then click Open to copy the name to the field.

Field	Comments
Company Name	Enter the company name from which you want to retrieve data. You can see a list of available company names by clicking Company. Click the company name, and then click OK to copy it to the field.
Windows Login	Place a check mark in this field if you want to use your Windows login to access the database. Using a Windows login means that you do not have a separate user ID and password.
User ID	Enter the user ID that you use when logging in.
Password	<p>Enter the password for the user ID.</p> <p>There will usually be a special user ID and password set up for the Dynamics NAV ODBC Driver— for example, a user ID with read permissions that has been created for a specific reason. This is what you should use. (You should not enter your personal user ID or password here because others will be able to see it.)</p> <p>If you do not enter anything in the User ID and Password fields, you will have to enter an ID and password every time you want to open the database from another program.</p> <p>You can read more about security in the manual <i>Installation and System Management: C/SIDE Database Server for Microsoft Dynamics NAV</i> or <i>Installation and System Management: SQL Server Option for Microsoft Dynamics NAV</i>.</p>
About...	Opens the About dialog for the driver and displays your license information (if a database and a company are available),
Import License	Click this button to open a standard Windows browse dialog. Locate and select your license file (.flf). This file is renamed <code>fin.flf</code> and copied to the directory where the driver is installed.

Specifying Options

In the **Dynamics NAV ODBC Setup** window, click the Options button to specify the data source options. The **Options** box appears:



Identifiers	All Except Dot	Enable BLOB Fields	<input checked="" type="checkbox"/>
Language	Auto (Windows Lai...	Read-Only	<input type="checkbox"/>
Option Field Type	Text	Closing Date	<input type="checkbox"/>
Commit Cache	<input checked="" type="checkbox"/>		
DBMS Cache (KB)	8000		

Fill in the fields according to these guidelines:

Field	Comments				
Identifiers	In this field, select one of four options. The options in this field and their implications are described in detail on page 9.				
Language	<p>All possible languages are shown in the Language list box. Besides the specific language options, the Dynamics NAV ODBC Driver offers the following general options:</p> <table border="0"> <tr> <td>Neutral</td> <td>This option disables multilanguage functionality. The Dynamics NAV ODBC Driver will only show Name properties and not captions.</td> </tr> <tr> <td>Auto (Windows Language)</td> <td>This option uses multilanguage functionality to show captions in the language of the operating system's regional settings.</td> </tr> </table>	Neutral	This option disables multilanguage functionality. The Dynamics NAV ODBC Driver will only show Name properties and not captions.	Auto (Windows Language)	This option uses multilanguage functionality to show captions in the language of the operating system's regional settings.
Neutral	This option disables multilanguage functionality. The Dynamics NAV ODBC Driver will only show Name properties and not captions.				
Auto (Windows Language)	This option uses multilanguage functionality to show captions in the language of the operating system's regional settings.				
Option Field Type	<p>Click the AssistButton to select one of two ways that option field values can be transferred:</p> <table border="0"> <tr> <td>Text</td> <td>Values are transferred as text strings, that is, the texts that appear in the drop-down list.</td> </tr> <tr> <td>Integer</td> <td>Values are transferred as integers. The options in a drop-down list are numbered 0, 1, 2, 3 . . .</td> </tr> </table>	Text	Values are transferred as text strings, that is, the texts that appear in the drop-down list.	Integer	Values are transferred as integers. The options in a drop-down list are numbered 0, 1, 2, 3 . . .
Text	Values are transferred as text strings, that is, the texts that appear in the drop-down list.				
Integer	Values are transferred as integers. The options in a drop-down list are numbered 0, 1, 2, 3 . . .				
Commit Cache	<p>Specifies whether commit cache should be used:</p> <table border="0"> <tr> <td>Yes (checked)</td> <td>Use commit cache.</td> </tr> <tr> <td>No (unchecked)</td> <td>Do not use commit cache.</td> </tr> </table>	Yes (checked)	Use commit cache.	No (unchecked)	Do not use commit cache.
Yes (checked)	Use commit cache.				
No (unchecked)	Do not use commit cache.				
DBMS Cache (KB)	Enter the size of the cache (0–30,000 KB)				
Enable BLOB Fields	<p>Specifies whether BLOB fields should be visible from ODBC:</p> <table border="0"> <tr> <td>Yes (checked)</td> <td>BLOB fields can be seen from ODBC.</td> </tr> <tr> <td>No (unchecked)</td> <td>BLOB fields are hidden.</td> </tr> </table>	Yes (checked)	BLOB fields can be seen from ODBC.	No (unchecked)	BLOB fields are hidden.
Yes (checked)	BLOB fields can be seen from ODBC.				
No (unchecked)	BLOB fields are hidden.				
Read-Only	<p>Specifies whether access to the database should be read-only.</p> <table border="0"> <tr> <td>Yes (checked)</td> <td>Access is read-only.</td> </tr> <tr> <td>No (unchecked)</td> <td>Access is read/write.</td> </tr> </table>	Yes (checked)	Access is read-only.	No (unchecked)	Access is read/write.
Yes (checked)	Access is read-only.				
No (unchecked)	Access is read/write.				

Field	Comments
Closing Date	Specifies whether the connection supports closing dates.
	Yes (checked) Closing dates are supported.
	No (unchecked) Closing dates are not supported.

Converting Identifiers

The option you select in the **Identifiers** field controls the way identifiers such as table names and field names are transferred from Dynamics NAV to an external program. The choice you make affects the way you use identifiers in external programs and the way you must write SQL statements. You can read about this on page 10.

The various options are:

Option	Comments
<i>All Except Dot</i>	Letters, numbers, symbols and spaces are transferred unchanged. Dots and question marks are converted to underscores (_).
<i>All Characters</i>	Letters, numbers, symbols, dots and spaces are transferred unchanged.
<i>All Except Space</i>	Letters, numbers and symbols are transferred unchanged. Spaces, dots and question marks are converted to underscores (_).
<i>a-z,A-Z,0-9,_</i>	Only letters and numbers are transferred unchanged. Symbols (except %), spaces, dots, parentheses and question marks are converted to underscores (_). The % sign is converted to PCT. The \$ sign is converted to USD.

Example

This table shows how the names of the **No.**, **Sales (LCY)**, **Profit %** and **Shelf/Bin No.** fields from the **Item** table are converted with the four different options:

Field Name	All Except Dot	All Characters	All Except Space	a-z,A_Z,0-9,_
No.	No_	No.	No_	No_
Sales (LCY)	Sales (LCY)	Sales (LCY)	Sales_(LCY)	Sales_LCY_
Profit %	Profit %	Profit %	Profit_%	Profit_PCT
Shelf/Bin No.	Shelf/Bin No_	Shelf/Bin No.	Shelf/Bin_No_	Shelf_Bin_No_

Using Identifiers in External Programs

In some cases, field names and table names with spaces and/or symbols must be converted by the Dynamics NAV ODBC Driver when they are returned as identifiers to an external program. This is necessary if the external program does not support spaces and/or symbols in identifiers (this may differ from program to program). You specify

the kind of conversion that is necessary by choosing one of the options described in the preceding table.

As an example, Microsoft Query does not support identifiers with dots (for example, the **No.** field in many tables). To have Microsoft Query handle these names correctly, use a data source with the *All Except Dot* option in the **Identifiers** field.

Writing SQL Statements

When writing SQL statements, you must write field names according to the identifier option that has been chosen. On page 9, you can read about how the various options work and see some examples of how field names are converted.

If you have chosen the *All Characters*, *All Except Space* or *All Except Dot* option in the **Identifiers** field, you must use quoted identifiers, that is, include field names in quotation marks. For example, if you have chosen the *All Except Space* option, the **Sales_(LCY)** field name must be written as "Sales_(LCY)".

If you have chosen the *a-z,A-Z,0-9,_* option, you do not have to use quoted identifiers.

Adding a Data Source

You can set up multiple data sources with the same driver. For example, you can have different data sources with different databases or you can have data sources with different options. See the online Help in the ODBC Data Source Administrator for more information about the various types of data sources.

Note

NAV ODBC does not support File DSN.

To set up a new user data source:

- 1 In the Control Panel, click Administrative Tools, Data Sources (ODBC).
- 2 In the **ODBC Data Source Administrator** window, click the **User DSN** tab and then click Add.
- 3 Select the Dynamics NAV ODBC Driver, and click Finish.
- 4 Enter information in the **Dynamics NAV ODBC Setup** window as described on page 6.
- 5 Click OK to close the window.

Changing a Data Source

You can change the information in the data source at any time with the following steps:

- 1 In the Control Panel, click Administrative Tools, Data Sources (ODBC).
- 2 Select the data source you want to change, and click Configure.

- 3 Change the necessary fields in the **Dynamics NAV ODBC Setup** window by typing or selecting the new names or values.
- 4 Click OK to close the window.

Deleting a Data Source

If you no longer need a data source, you can delete it:

- 1 In the Control Panel, click Administrative Tools, Data Sources (ODBC).
- 2 Select the data source you want to delete, and click Remove.
- 3 Confirm the message that appears by clicking OK.

Chapter 3

Technical Documentation

The Dynamics NAV ODBC Driver makes data in a database stored locally or in a C/SIDE Database Server accessible to ODBC-enabled applications. You can use the Dynamics NAV ODBC Driver to retrieve Dynamics NAV data and insert it into an application such as a word processor or spreadsheet.

This chapter assumes that you are familiar with Dynamics NAV and with ODBC functionality. For additional information, refer to the Dynamics NAV documentation, as well as to documentation for the applications into which the Dynamics NAV data will be inserted.

The chapter contains the following sections:

- Overview
- Establishing a Connection
- Microsoft Dynamics NAV ODBC Driver Functionality
- Data Types
- Multilanguage Functionality

3.1 Overview

Open Database Connectivity (ODBC) is an interface defined by Microsoft Corporation as a standard interface to database management systems in the Windows XP and Windows Server 2003 environments. Applications using the ODBC interface can work with many different database systems.

The Dynamics NAV ODBC Driver opens a C/SIDE Database Server or local database to ODBC-enabled applications so that they can retrieve data from and write data to the database.

The Dynamics NAV ODBC Driver operates in the Windows XP and Windows Server 2003 environments. In these environments, it can function either as a stand-alone or as a client in a client/server configuration. You do not need to have a Dynamics NAV client installed to use the Dynamics NAV ODBC Driver.

Note that you cannot use the Dynamics NAV ODBC Driver with the SQL Server Option for Dynamics NAV.

3.2 Establishing a Connection

The following are examples of how to connect a Dynamics NAV ODBC data source, depending on the programming language you use.

Using C#

When using C# to connect, you could use the following example:

```
string myConnection = "DSN=Dynamics NAV Database";
OdbcConnection myConn = new OdbcConnection(myConnection);
myConn.Open();
...
myConn.Close();
```

Using Visual Basic
.NET

The following is an example of how you could connect to a Dynamics NAV ODBC data source when using Visual Basic .NET:

```
oOdbcConnection As New Odbc.OdbcConnection("DSN=Dynamics NAV
Database")
oOdbcConnection.Open()
...
oOdbcConnection.Close()
```

Using C++

The following gives an example of how you could connect to a Dynamics NAV ODBC data source when using C++:

```
SQLHENV henv;
SQLHDBC hdbc;
SQLHSTMT hstmt;
SQLRETURN retcode;

// Allocate environment handle
retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);

if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
// Set the ODBC version environment attribute
retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, 0);

if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
// Allocate connection handle
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);

if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
// Set login timeout to 5 seconds.
SQLSetConnectAttr(hdbc, (void*)SQL_LOGIN_TIMEOUT, 5, 0);
```

```
// Connect to data source
retcode = SQLConnect(hdbc, (SQLCHAR*)"Dynamics NAV Database",
SQL_NTS,
(SQLCHAR*)"Admin", SQL_NTS,
(SQLCHAR*)"Pass", SQL_NTS);
...
```


3.3 Microsoft Dynamics NAV ODBC Driver Functionality

The Dynamics NAV ODBC Driver has the functionality required for it to be used as a read/write data source.

SQL Conformance

The Dynamics NAV ODBC Driver is developed using ODBC version 2.5. The specification for this version provides three levels of SQL grammar conformance: Minimum, Core, and Extended. Each higher level provides more fully-implemented data definition and data manipulation language support. ODBC version 2.5 fully supports the Minimum SQL grammar, as well as many Core and Extended grammar statements. The Dynamics NAV ODBC Driver's support for SQL grammar is summarized in the following table.

SQL Grammar Statement	Minimum	Core	Extended
Create Table	o		
Create View		o	
Delete (searched)	o		
Drop Index		o	
Drop Table	o		
Drop View		o	
Insert	o		
Left Outer Join			o
Select	o		
- approximate numeric literal		o	
- between predicate		o	
- correlation name		o	
- date arithmetic			o
- date literal			o
- exact numeric literal		o	
- extended predicates			o
- in predicate		o	
- set function		o	
- time literal			o
- timestamp literal			o
Subqueries		o	
Unions			o
Update (searched)	o		
Use (this is an extension to ODBC SQL grammar)			o

ODBC API Conformance

The Dynamics NAV ODBC Driver fully conforms to the ODBC version 2.5 specification for Core API and Level 1 API, and supports most of the Level 2 function calls. The following table lists the ODBC API functions supported by the Dynamics NAV ODBC Driver.

ODBC Function	Conformance Level
SQLAllocConnect	Core
SQLAllocEnv	Core
SQLAllocStmt	Core
SQLBindCol	Core
SQLBindParameter ¹	Level 1
SQLBrowseConnect	Level 2
SQLCancel	Core
SQLColAttributes	Core
SQLColumns	Level 1
SQLColumnPrivileges	Level 2
SQLConnect	Core
SQLDataSources	Level 2
SQLDescribeCol	Core
SQLDescribeParam	Level 2
SQLDisconnect	Core
SQLDriverConnect	Level 1
SQLDrivers	Level 2
SQLError	Core
SQLExecDirect	Core
SQLExecute	Core
SQLFetch	Core
SQLForeignKeys	Level 2
SQLFreeConnect	Core
SQLFreeEnv	Core
SQLFreeStmt	Core
SQLGetConnectOption	Level 1
SQLGetCursorName	Core
SQLGetData	Level 1
SQLGetFunctions	Level 1
SQLGetInfo	Level 1
SQLGetStmtOption	Level 1

ODBC Function	Conformance Level
SQLAllocConnect	Core
SQLGetTypeInfo	Level 1
SQLMoreResults	Level 2
SQLNativeSql	Level 2
SQLNumResultCols	Core
SQLNumParams	Level 2
SQLParamData	Level 1
SQLPrepare	Core

¹The size of the column in the linked table in Access is always equal to the size of the field in Dynamics NAV. However, NAV ODBC will automatically truncate the bound parameter if the size of the bound parameter exceeds the length of the field defined in Dynamics NAV.

3.4 Data Types

The Dynamics NAV ODBC Driver supports the following SQL data types:

Dynamics NAV Data Types	SQL Data Types
BIGINTEGER	SQL_VARCHAR
BINARY	SQL_VARBINARY
BLOB	SQL_LONGVARBINARY
BOOLEAN	SQL_BIT
CODE	SQL_VARCHAR
DATE	SQL_DATE / SQL_TIMESTAMP
DATEFORMULA	SQL_VARCHAR
DATETIME	SQL_TIMESTAMP
DECIMAL	SQL_DECIMAL
DURATION	SQL_VARCHAR
GUID	SQL_VARCHAR
INTEGER	SQL_INTEGER
OPTION	SQL_INTEGER / SQL_VARCHAR
RECORDID	SQL_VARCHAR
TABLEFILTER	SQL_VARCHAR
TEXT	SQL_VARCHAR
TIME	SQL_TIME

When using the SQL statement CREATE TABLE, the Dynamics NAV ODBC Driver supports the following data types:

- Binary
- Boolean
- BLOB
- String
- Code
- BCD
- S32
- Date
- Time
- Timestamp

The following rules apply to entering search conditions in the <whereclause> – see page 31.

Dynamics NAV Data Types	Rules
BOOLEAN	Enter the value 0 or 1 (No is 0, Yes is 1).
INTEGER	Enter a signed integer in the range -2,147,483,648 to 2,147,483,647 (do not enter the commas).
BIGINT	Use this data type to store very large whole numbers. This data type is a 64 bit integer. Note that the SQL data type is SQL_VARCHAR, so you must enter data as a string using single quotes, for example '123456789'.
OPTION	If you have selected <i>Text</i> in the Option Field Type field in the Options box in the Dynamics NAV ODBC Setup window, then enter option fields as the option string (in single quotes). If you have selected <i>Integer</i> as the Option Field Type , then you should enter option fields as the numerical option value. Thus, if the first option string is <i>Open</i> , this value will be entered as <i>Open</i> if the Option Field Type is <i>Text</i> , or as 0 (zero) if the Option Field Type is <i>Integer</i> . Option strings in a set are numbered from 0 (zero) upwards.
DATE	Enter a date in this format: {d 'yyyy-mm-dd'} where y=year (1752-9999), m=month (01-12) and d=day (01-31). Note that if you have closing date support, you must use the timestamp format described on page 23.
TIME	Enter a time in this format: {t 'hh:mm:ss'} where h=hour, m=minute and s=second.
DATETIME	Use this data type to store timestamps in this format: {ts 'yyyy-mm-dd hh:mm:ss'} where y=year, m=month, d=day, h=hour, m=minute and s=second. The timestamp is always shown in local time. DATETIME is always stored in the same format regardless of closing date support.
DURATION	Use this data type to represent the difference between two datetimes, in milliseconds. This value can be negative. It is a 64 bit integer, and you must enter data as a string using single quotes. You can enter the data either as a number, such as '122', or as text, such as '2 min 2 sec'. Note that the data type has a property <i>Standard date time unit</i> where you can set the standard unit of measure. Dynamics NAV ODBC users must know this unit of measure if they enter data as a number, since Dynamics NAV interprets input such as 60 as 60 milliseconds, seconds, minutes, hours or days depending on the standard date time unit. The duration will always be displayed in a readable format such as 2 min, 2 sec rather than the number 122.
TEXT / CODE	Enter a string in single quotes.

Dynamics NAV Data Types	Rules
DECIMAL	Enter a number (without quotes). The decimal precision of Dynamics NAV ODBC is 16 significant digits, giving you a range from -999 999 999 999.99 to +999 999 999 999.99. Significant digits include +,- and the decimal point.
DATEFORMULA	Enter a date formula such as 1Q or 1W+1D.
GUID	Use this data type to give a unique identification number to any database object in this format: {'12345678-1234-1234-1234-1234567890AB'}. Each character denotes a hexadecimal character.

If an invalid value is used, an error message will be displayed.

Comparison Operators

The Dynamics NAV ODBC Driver uses the following comparison operators:

Operator	Function
=	Equals
<=	Less than or equal to
>=	Greater than or equal to
<	Less than
>	Greater than
<>	Not equal to

Operator Precedence

An important property of an operator is its precedence. Precedence determines the order in which the Dynamics NAV ODBC Driver evaluates different operators in the same expression. When evaluating an expression containing multiple operators, the driver evaluates operators with higher precedence before those with lower precedence. The driver evaluates operators with equal precedence from left to right within an expression.

The following table lists the levels of precedence among SQL operators from high to low. Operators listed on the same line have the same precedence.

Precedence	Operator	Associate
1. (Highest)	()	Left to right
2.	MUL DIV	Left to right
3.	ADD SUB	Left to right
4.	EQ GE GT LE LT NE	Right to left
5.	NOT	Left to right
6.	AND	Left to right

Precedence	Operator	Associate
7.	ALL ANY BETWEEN IN LIKE OR	Left to right

Parentheses within an expression override operator precedence. The driver evaluates expressions inside parentheses before those outside.

Data Type Comparison Rules

This section describes how the driver compares values within each data type.

Numerical Values

A larger value is considered greater than a smaller one. All negative numbers are less than all positive numbers. Thus, -1 is less than 100; -100 is less than -1.

Date Values

A later date is considered greater than an earlier one.

A date entered in the SQL statement {d '1995-12-31'} is considered an ordinary date – not a closing date.

However, the Dynamics NAV ODBC Driver supports closing dates. To support closing dates, in the **Dynamics NAV ODBC Setup** window, click Options and place a check mark in the **Closing Date** field in the **Options** box that appears. When you have enabled closing date support, you must enter data in a field with the DATE data type in the following format: {ts '2001-01-01 23:59:59'}. The time part can hold one of two values: 23:59:59, which means a closing date, and 00:00:00, which means an ordinary date. The default setting of the **Closing Date** option is disabled.

Note that even with the use of the SQL statement {ts '2001-01-01 23:59:59'}, fields with the DATE data type do not store time stamps. To store the actual time stamp, use the DATETIME data type.

Character String Values

Character values are compared using non-padded comparison semantics. This means that the driver compares two values character-by-character until it finds a character that varies. The value with the greater character in that position is considered the greater value. If two values of different lengths are identical up to the end of the shorter one, the longer value is considered greater. If two values of equal length have no differing characters, then the values are considered equal.

Example: 'Str 2' is greater than 'Str 10'.

Comparing Option Fields

If, when setting up the options in the **Options** box in the **Dynamics NAV ODBC Setup** window, you selected *Text* as the **Option Field Type**, the comparison operators (see page 22) use the option strings as a basis for comparison for option fields. If you selected *Integer* as the **Option Field Type**, then the comparison operators use the numerical values of the options to compare the values in option fields.

3.5 Multilanguage Functionality

The Dynamics NAV ODBC Driver handles the multilanguage functionality in Dynamics NAV. The Dynamics NAV ODBC Driver can retrieve the application data from Dynamics NAV in different languages independent of the current Dynamics NAV application language.

When you are running Dynamics NAV ODBC and you open the **ODBC Data Source Administrator** window from the operating system's Control Panel, you can set up the Dynamics NAV ODBC Driver as described on page 5. Use the **Language** field properties to set up the connection appropriate to the user.

C/SIDE uses the following hierarchy when showing the application data:

- 1 Global language
- 2 Primary language of global language
- 3 Application language
- 4 Primary language of application language.

For more information about multilanguage functionality, see the manual *Application Designer's Guide*.

Specifications

The Dynamics NAV ODBC Driver covers the following multilanguage features:

- Table name
- Field name
- OptionString value

When you link a table by selecting an application language other than the default language and this language has a corresponding output from Dynamics NAV, you will notice that the value of the table name, all the field names, and the option fields within that table are shown in the language you selected.

Limitations

You cannot use the Dynamics NAV ODBC to create a table with multilanguage *Caption* properties. This means that no matter what language has been chosen in the **Options** box in the **Dynamics NAV ODBC Setup** window, the table that is created for any Dynamics NAV ODBC application will use the Name property. No Caption property can be written to the application database.

You will not be able to change the language choice in real-time mode. Dynamics NAV ODBC can only accept one setting at the time of loading. To switch the language when using the Dynamics NAV ODBC Driver, you must to close the Dynamics NAV ODBC connection and thereby release it from the memory, change to the preferred language in the **Options** box in the **Dynamics NAV ODBC Setup** window, and start the Dynamics NAV ODBC connection again.

Scenarios

There are three different scenarios that are possible when running multilanguage for the Dynamics NAV ODBC Driver. In the following scenarios, the user has the following settings:

Property	Setting
Operation system regional setting	German (Austrian)
Code base language	English (United States)
Available license file granules and language folders	English (United Kingdom) German (Standard) German (Austrian)

Neutral

In the first scenario, the global language of the Dynamics NAV application is English (United Kingdom), and the **Language** field in the **Dynamics NAV ODBC Options** box is set to *Neutral*.

Dynamics NAV ODBC Result The application data is retrieved and shown using the field and table names as they are in their respective Name properties in Dynamics NAV, and the multilanguage captions are not used.

Auto (Windows Language)

In this scenario, the global language of the Dynamics NAV application is English (United Kingdom), but the **Language** field in the choice in the **Dynamics NAV ODBC Options** box is set to *Auto (Windows Language)*.

Dynamics NAV ODBC Result The application data retrieved is shown in German (Austrian) if the selected objects have multilanguage captions for the language code German (Austrian).

If the selected objects do not have multilanguage captions with the language code for German (Austrian), the application data is shown in the code base language, English (United States). If there is no caption in Dynamics NAV for English (United States), the Dynamics NAV ODBC Driver uses the field and table names as they are defined in their respective Name properties in Dynamics NAV.

Specific Language

In this scenario, the global language of the Dynamics NAV application is still English (United Kingdom), but the **Language** field in the **Dynamics NAV ODBC Options** box is set to German (Austrian).

Dynamics NAV ODBC Result The application data retrieved is shown in German (Austrian) if the selected objects have multilanguage captions for the language code for German (Austrian).

If the selected objects do not have multilanguage captions with the language code for German (Austrian), the application data is shown using the field and table names as they are defined in Dynamics NAV.

Appendix A

SQL Statement Reference Guide

This appendix describes all the supported SQL statements and serves as a reference guide.

The appendix contains the following sections:

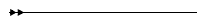
- Introduction to the Reference Guide
- SELECT Statement
- Parameter Markers
- Keys and Performance
- Predicates in WHERE Clauses
- GROUP BY Clause
- HAVING Clause
- ORDER BY Clause
- INSERT Statement
- DELETE Statement
- UPDATE Statement
- CREATE TABLE Statement
- DROP TABLE Statement
- Microsoft Dynamics NAV FlowFields

A.1 Introduction to the Reference Guide

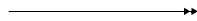
The SQL statement reference guide is organized top-down, starting with the statements and proceeding to a description of the possible elements in the statements (clauses and predicates).

Conventions Used in the Reference Guide

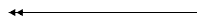
The following graphical conventions are used:



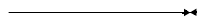
Indicates the beginning of a statement.



Indicates that the statement syntax is continued on the next line.



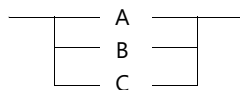
Indicates that the statement syntax is continued from the previous line.



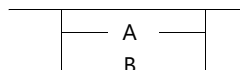
Indicates the end of a statement.



Denotes the repeat symbol. Terms enclosed within the repeat symbol may be repeated any number of times with varying values.



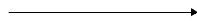
Multiple choices for parameters are enclosed in boxes with horizontal lines. There will be as many lines as there are choices.



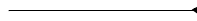
Optional parameters are enclosed in lines descending from the main diagram line, as shown in the diagram above. The statement is correct without the optional parameters. If the parameter is not specified, an underscore indicates the default value.

Some complex diagrams have been broken up by grouping several parameters and clauses by a specified name in the main diagram. This specified name is enclosed in angle brackets (<>). Such complex statements are later represented using sub-unit

diagrams. A sub-unit diagram starts with



and ends with

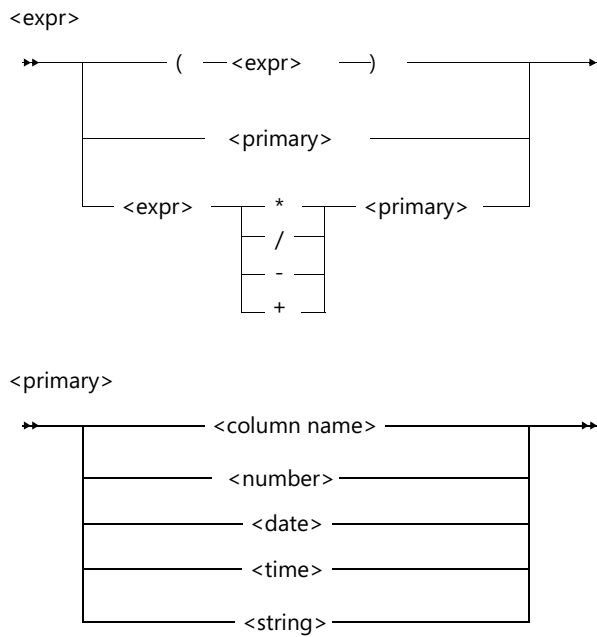


Uppercase letters denote keywords. Note that this convention is used only to make the syntax diagrams easier to read: the SQL reserved words (keywords) are not case-sensitive.

Notice that all the examples of SQL statements in this section are written assuming that the driver has been set up with the *a-z,A-Z,0-9,_* option in the **Identifiers** field (see page 9).

Expressions

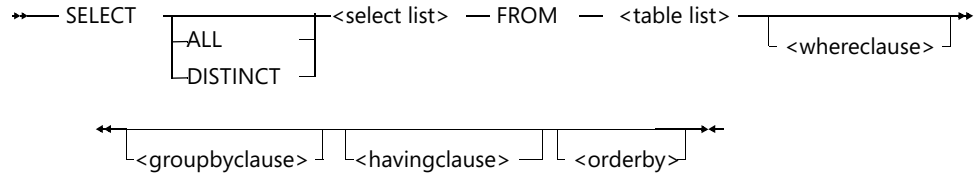
The following diagram illustrates how expressions are constructed in the Dynamics NAV ODBC Driver:



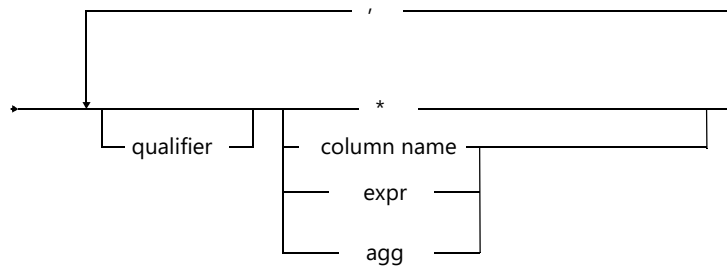
A.2 SELECT Statement

Selects rows from one or more tables.

Syntax



<select list>



General Rules

The SELECT statement retrieves data from one or more tables. It takes the tables listed in the <tablelist> as input and produces an output table that includes only those rows that satisfy the search condition specified in the <whereclause>.

By default, all rows that satisfy the search condition are included in the output table. You can, however, prevent duplicate rows from being included by using the DISTINCT keyword

Syntax Rules

qualifier: the name of a table or its alias, if one has been specified, in the FROM clause. If only one table is specified, the qualifier is not needed.

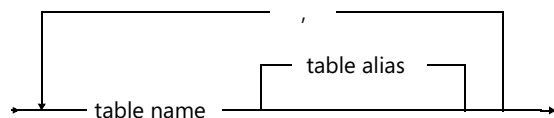
asterisk (*): this symbol includes all columns of the table.

column name: the specific column.

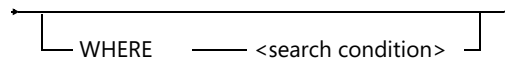
expr: a field that contains an expression, with or without a column name. See page 23 for a diagram of expression syntax.

agg: an aggregate function. There are these aggregate functions: COUNT(* | expr), AVG(expr), MAX(expr), MIN(expr), SUM(expr).

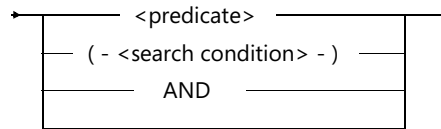
<tablelist> The <tablelist> lists the tables (and aliases) used in the SELECT statement.



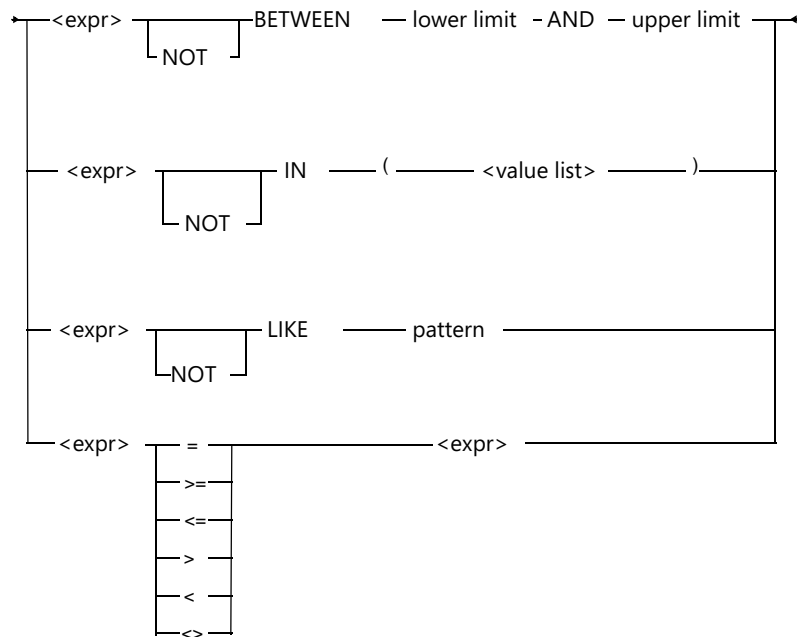
<whereclause> The <whereclause> specifies the search condition against which the rows in the <tablelist> are evaluated.



<search condition>



<predicate>



Example

This example produces a list of customers whose balance is greater than or equal to 20000:

```
SELECT * FROM Customer WHERE Balance >= 20000
```

A.3 Parameter Markers

A parameter is a variable in an SQL statement. For example, suppose an *Item* table has columns named **No.**, **Description**, and **Price**. To add a part without parameters would require constructing an SQL statement such as:

```
INSERT INTO Item (No_, Description, Unit_Price) VALUES ('70012',  
'Glass Door', 75)
```

Although this statement inserts a new item record, it is not a good solution for an item entry application because the values to insert cannot be hard-coded in the application.

An alternative is to construct the SQL statement at run time using the values to be inserted. This also is not a good solution because of the complexity of constructing statements at run time.

The best solution, if the client application supports it, is to replace the elements of the VALUES clause with question marks (?) or parameter markers:

```
INSERT INTO Item (No_, Description, Unit_Price) VALUES (?, ?, ?)
```

The parameter markers are then bound to application variables. To add a new row, the application has only to set the values of the variables and execute the statement. The driver then retrieves the current values of the variables and sends them to the data source.

An application cannot place parameters in the following locations:

- In a SELECT list
- As both expressions in a comparison-predicate
- As both operands of a binary operator
- As both the first and second operands of a BETWEEN operation
- As both the first and third operands of a BETWEEN operation
- As both the expression and the first value of an IN operation
- As the operand of a unary + or – operation

A.4 Keys and Performance

When you are filtering and sorting records in Dynamics NAV ODBC, you should use the most appropriate fields in the WHERE clause.

This field **must** be the first field in one of the keys that have been defined for the table you are accessing.

If the table you want to access does not contain a key that has this field as the first field, you **must** define such a key.

Therefore, you must ensure that the tables you access contain keys in which the filtering field is the first field. Using these keys will greatly increase the performance of NAV ODBC.

Example :

```
SELECT * FROM Table  
WHERE fld1 < {d 2007-01-01}
```

Assume that Table contains the following fields: fld1, fld2, fld3, fld4, fld5.

The following keys are appropriate for this query:

Key 1: "fld1"

Key 2: "fld1, fld2, fld3, fld4, fld5"

The following keys are not appropriate and will seriously affect performance:

Key 1: "fld2,fld1"

Key 2: "fld2,fld3,fld4,fld1"

A.5 Predicates in WHERE Clauses

WHERE Clause BETWEEN Predicate

Compares a value to a range of values.

Syntax

← <expr> [NOT] BETWEEN — lower limit — AND — upper limit →

General Rules

The BETWEEN predicate checks a value against a range bounded by the lower and upper limits. The condition is true if the value being checked is greater than or equal to the lower limit and less than or equal to the upper limit. Each row for which this condition is true is included in the result set. The value being compared should be comparable with the lower and upper limits.

By using the logical operator NOT, you can test a value outside the specified range.

One important thing to remember about the BETWEEN predicate is the order of the lower and upper limits. The lower limit must be less than or equal to the upper limit.

Syntax Rules

lower limit: the lower limit of the range that is being checked.

upper limit: the upper limit of the range that is being checked.

Example

This example retrieves all customers with a post code in the range of 1000 to 1234:

```
SELECT * FROM Customer WHERE Post_Code BETWEEN '1000' AND '1234'
```

WHERE Clause IN Predicate

Compares a value against a list of values for equality.

Syntax

← <expr> [NOT] IN — (— <value list> —) →

General Rules

The IN predicate checks a value against a set of values for equality. The condition is true if the value being compared matches any of the values in the value list. If you use the logical operator NOT, the checking principle is reversed.

Syntax Rules

value list: a list of values against which a value is checked for equality.

Example

This example retrieves all customers whose post code is 1000, 2000 or 3000:

```
SELECT * FROM Customer WHERE Post_Code IN ('1000','2000','3000')
```

WHERE Clause LIKE Predicate

Compares a string value against a pattern for equality.

Syntax

```

  <expr>
  └──┬──┘
     │
     └── NOT ───┘
  ─── LIKE ─── pattern ───

```

General Rules

The LIKE predicate compares a string type value with a pattern. The condition is true if a match is found, false if it is not. Every row for which the condition is true is included in the result set.

The pattern is any character pattern against which the value is compared, and it may include some special characters. A percent sign (%) matches any number of characters including zero characters in the same position.

If you want to include a percent sign in the search pattern, then enter a backslash (\) before it to remove its special meaning. For example, \\ represents the backslash itself.

The logical operator NOT negates the LIKE predicate.

Syntax Rules

pattern: the string against which a value is compared.

Example

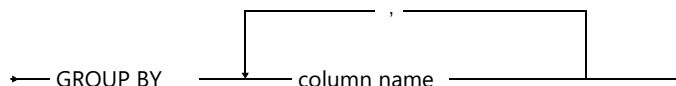
This example retrieves all customers whose name contains Hansen:

```
SELECT * FROM Customer WHERE Name LIKE '%Hansen%'
```

A.6 GROUP BY Clause

Groups rows of data based on the value in one or more columns.

Syntax



General Rules

The GROUP BY clause groups the selected rows of data according to values in the group column referred to by the column name. This produces an output that contains one row for each distinct value in the group column.

When you use the GROUP BY clause, the value expression list in the SELECT statement can only contain group columns, expressions containing group columns and aggregate functions. Otherwise, an error will occur.

If you specify multiple columns, the selected rows will be grouped first according to the first group column and within that grouping according to the second group column.

See page 30 for a list of the aggregate functions that are available. In all of the aggregate functions, the value expression can be quantified by a quantifier that can be either DISTINCT or ALL. The default is ALL, which means that all values of the value expression for all rows in the group should be considered. If the quantifier is DISTINCT, only distinct values of the value expression in the rows of the group are considered for computing the value of the function.

Syntax Rules

column name: the name of the column on which rows will be grouped.

Example

This example retrieves the number of customers per country/region. We join the **Customer** and the **Country/Region** tables in order to get the names of the countries/regions instead of the country/region codes:

```
SELECT a.Name, Count(*) FROM Country_Region a, Customer b
WHERE a.Code = b.Country_Region_Code GROUP BY a.Name
```

A.7 HAVING Clause

Specifies conditions for including groups in the output.

Syntax

← HAVING — search condition —————→

General Rules

The HAVING clause makes it possible to specify conditions on grouped data so as to eliminate some of them and include the rest in the output.

There is an important difference between the HAVING and the WHERE clauses. The WHERE clause filters rows before they are passed on to the GROUP BY clause. The HAVING clause filters the output of the GROUP BY clause, and you can use aggregate functions in the HAVING clause.

Syntax Rules

search condition: refers to the search condition that you may specify on grouped rows so as to include them selectively in the output. See page 31 for a description of search conditions.

Example

This example retrieves the name and balance of all customers with a balance that is greater than 5000 and sorts the list by balance. This statement could be rewritten using WHERE.

```
SELECT Name, Balance FROM Customer
GROUP BY Name, Balance HAVING Balance > 5000
ORDER BY Balance DESC
```

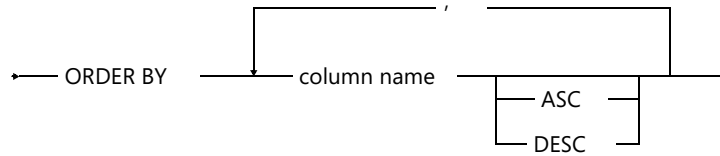
This example retrieves information from the **Sales Invoice Header** table. It produces a list of customers and the average, maximum and minimum amounts of their sales invoices, but only if there is more than one invoice for the customer. This statement could not be rewritten using WHERE.

```
SELECT Sell_to_Customer_Name, Avg(Amount), Max(Amount), Min(Amount)
FROM Sales_Invoice_Header
GROUP BY Sell_to_Customer_Name HAVING Count(*)>1
```

A.8 ORDER BY Clause

Sorts the output of a query in ascending or descending order on the basis of values in one or more columns.

Syntax



General Rules

The ORDER BY clause sorts the output of a query in the desired order. By default, rows are sorted in ascending order of values. To reverse the order of the sort, use the keyword DESC.

You can sort the output of a query by sorting multiple columns. In this case, the output is first sorted by the values in the first column. Within each distinct value in the first column, the rows are sorted by the values in the second column, and so on. When sorting rows based on multiple columns, each column can be assigned its own sorting order with ASC or DESC.

Syntax Rules

column name: the column in which the data will be sorted.

Example

This example retrieves all customers sorted by post code and name:

```
SELECT * FROM Customer ORDER BY Post_Code, Name
```

A.9 INSERT Statement

Inserts rows into a table.

Syntax

```

-- -- INSERT [INTO] tablename -- <insertvals> --
          |
          |----<br>
          |
          |
          |
          |-----<br>
          |-----VALUES -- ( [value] ) --
          |
          |-----<br>
          |-----columnlist
  
```

General Rules

The INSERT statement inserts data into a table. If you are inserting data in only some of the columns in the table, you must explicitly mention these column names in the INSERT statement. If you do not provide a column list explicitly, the INSERT statement will try to insert values in all columns of the table.

There should be as many data values as there are columns in the table or the column list and the corresponding data types should match. If they do not, an error will be reported and no values will be inserted.

You can insert explicit data values one row at a time using the VALUES clause. Each data value should be separated from the next by a comma.

You can only insert records in tables where your license gives you permission to insert. You cannot insert into a virtual table.

You should be aware that triggers are *not* run when you insert records through the Dynamics NAV ODBC Driver.

Syntax Rules

tablename: name of the table into which you are inserting rows.

columnlist: a list of columns in the table into which you are inserting data values. The column names must be separated by commas.

value: refers to a data value that is being inserted into a column. When specifying a data value, follow these conventions:

- Character-type values must be enclosed in single quotes.
- Date-type and time-type values must be enclosed in the {d' <date value>'} and {t' <time value>'} formats respectively, unless you have closing date support in which case you must use the {ts' <date and time value>} format instead.

You can also use the DATETIME data type to store the actual timestamp in the following format: {ts' <date and time value>}

For more information about date formats, see page 23.

- All numeric values can be entered literally as values.
- A data value can be expressed as an expression provided the expression evaluates to a type that is compatible with the base data type of the column.
- To identify dates as undefined dates, use the date 1753-01-01.
- In the case of a partial column list, the values for columns that are not in the column will be set to default values.

Example

This example inserts a record in the **Country/Region** table. The column list contains only two of the columns of the table. The rest of the columns will be inserted as default values.

```
INSERT INTO Country/Region (Code, Name)
VALUES ('NZ', 'New Zealand')
```


A.10 DELETE Statement

Deletes rows in the specified table.

Syntax

```

→DELETE FROM  — tablename —————→
                |
                | WHERE <search condition> |
                |
  
```

General Rules

The DELETE statement deletes rows from a table. If no conditions are specified, all rows in the table will be deleted.

You can optionally specify a WHERE clause to select the rows from the table for deletion. The WHERE clause can specify any valid search condition that selects the rows. You can select the WHERE clause in the same way as you do in a SELECT statement.

You can only delete records in tables where your license gives you permission to delete. You cannot delete from a virtual table.

You should be aware that triggers are *not* run when you delete records through the Dynamics NAV ODBC Driver.

Syntax Rules

tablename: name of the table from which you are deleting rows.

search condition: this refers to a condition for choosing rows for deletion from the named table. You may specify any valid condition that you can use in the WHERE clause of a select statement (see page 30).

Example

This example deletes *all* rows from the **Customer** table:

```
DELETE FROM Customer
```

This example deletes a single customer from the **Customer** table:

```
DELETE FROM Customer WHERE No_ = '12345'
```

A.11 UPDATE Statement

Updates rows in a table.

Syntax

```
→ UPDATE — tablename — SET ————┐ columnname = value ───→  
←──┘ WHERE <search condition> ───┘
```

General Rules

The UPDATE statement updates data in a table. If no conditions are specified, all rows in the table are updated.

Set values into the columns to be updated by using the SET clause. The left-hand side of the SET clause must be a column in the table being updated. The right-hand side must contain a data value that can be assigned to the column.

You can optionally specify a WHERE clause to select the rows from the table for update. The WHERE clause can specify any valid search condition that selects the rows.

You can only update records in tables where your license gives you permission to update. You cannot update in a virtual table.

You should be aware that triggers are *not* run when you update records through the Dynamics NAV ODBC Driver.

Syntax Rules

tablename: name of the table in which you are updating rows.

columnname: this refers to the name of the column in the table whose data is being updated.

search condition: this refers to a condition for choosing rows for updating from the named table. You may specify any valid condition that you can use in the WHERE clause of a select statement (see page 30).

Examples

This example updates the **Unit Price** field in all records of the **Item** table:

```
UPDATE Item SET Unit_Price = Unit_Price * 1.25
```

This example updates the **Country/Region Code** field in selected records of the **Customer** table:

```
UPDATE Customer SET Country_Region_Code = 'CN'  
WHERE Country_Region_Code = 'HK'
```

A.12 CREATE TABLE Statement

Creates a table.

Syntax

```

-- CREATE TABLE -- tablename -- ( --> <column --> --<
<column
-- columnname -- <data type> --

```

General Rules

The CREATE TABLE statement creates a table in the database.

You can define the columns of the table by specifying a column name and its data type. The data types that are supported are shown in the syntax rules below. Some of the data types optionally take one numeric argument. In the case of character data types, the length of the column can be specified. In its absence, a default value of one (1) is assumed. The length specifies the maximum length of the string data that can be stored in the column (the declared length in C/SIDE®).

When a table is created, the Dynamics NAV ODBC Driver generates a table number automatically. You must have the necessary permissions to insert tables, and there must be free table numbers in the database. The range 49,999 – 99,999 is used, with allocation starting from the top.

Syntax Rules

tablename: name of the table you wish to create. A table name can be up to 30 characters long, and it must be unique within a database. A table number will be generated automatically.

columnname: name of the column (field) being defined. A column name can be 30 characters long, and it must be unique within the table. A field number will be generated automatically.

data type: the data type of the column (field) being defined. The following table shows the relationship between the data types you can use in the Dynamics NAV ODBC Driver and the C/SIDE data types.

Dynamics NAV ODBC Type	C/SIDE Type	Comments
BCD	Decimal	
BLOB	BLOB	
BINARY	BINARY	Takes one argument: length.
BOOL	Boolean	
CODE	Code	Takes one argument: length.
DATE	Date	
TIMESTAMP	DateTime	
S32	Integer	

Dynamics NAV ODBC Type	C/SIDE Type	Comments
STRING	Text	Takes one argument: length.
TIME	Time	

Example

This example creates a table with three fields: an Integer, a Decimal and a Text field:

```
CREATE TABLE Sample (
    Code S32,
    Value BCD
    Name STRING(50)
)
```

A.13 DROP TABLE Statement

Drops a table from the database.

Syntax

```
↔ DROP TABLE — tablename ↔
```

General Rules

The DROP TABLE statement drops the named table from the database. When a table is dropped (deleted), all data in the table is lost.

You must have the necessary permissions in order to drop (delete) a table.

Syntax Rules

tablename: name of the table to drop.

Example

The following example drops (deletes) the table named "Sample" from the database:

```
DROP TABLE Sample
```

A.14 Microsoft Dynamics NAV FlowFields

Dynamics NAV has a special field type called a FlowField, which contains values drawn from other tables. As the values in the original tables change, the values in the FlowField change accordingly. FlowField values are retrieved by applying a Dynamics NAV field class – called a FlowFilter® – to the FlowField.

The data type of a FlowFilter is always SQL_VARCHAR (string). The syntax of the FlowFilter is specific to Dynamics NAV.

Setting a FlowFilter on a FlowField is done as a work-around in the WHERE clause. An example of the syntax is:

```
{pred SetFlowFilterMultiple, "TableName". "FieldName", 'A  
searchString' }
```

This allows you to set both single and multiple values in the filter on the FlowField. This predicate also allows you to set dynamic parameters.

Note

.....
These new statements use a different syntax – the fully qualified name is enclosed in double quotation marks (") and a dot (.) separates the table and field names
.....

SetFlowFilterMultiple

This predicate works with single values and should be used instead of the old *SetFlowFilter* predicate because it is more flexible, will be used in future development and most importantly because it also supports joined tables.

Examples

The following example shows how to set a FlowFilter on the **Budget Filter** field in the **G/L Account** table:

```
SELECT * FROM G/L Account  
WHERE {pred SetFlowFilterMultiple, "G/L Account". "Budget_Filter",  
'10000..30000' }
```

SetFlowFilterMultiple is the name of the extended predicate, **G/L Account** is the table containing the filter, and *Budget_Filter* is the field being filtered. The expression '10000..30000' is the filter that is set on the **Budget Filter** field in the **G/L Account** table.

If you want to parameterize the query, enter a '?' in place of the filter:

```
SELECT * FROM G/L Account  
WHERE {pred SetFlowFilterMultiple, "G/L Account". "Budget_Filter",  
? }
```

When you run the query, you are prompted to enter the parameters.

The following example shows how to use *SetFlowFilterMultiple* to set a FlowFilter on joined tables:

```
SELECT  
Customer.Name,  
Customer.No_,
```

```

"Cust_ Ledger Entry"."Document No_",
"Cust_ Ledger Entry"."Remaining Amount"
FROM
Customer Customer ,
"Cust_ Ledger Entry" "Cust_ Ledger Entry"
WHERE
Customer.No_ = "Cust_ Ledger Entry"."Customer No_"
AND
{ pred SetFlowFilterMultiple, "Cust_ Ledger Entry"."Date Filter",
311200 }
ORDER BY
Customer.No_

```

This example links the **Customer** table and the **Cust. Ledger Entry** table. The tables are linked by the **Customer No.** field and the FlowFilter is set on the **Date** field in the **Cust. Ledger Entry** table. '311200' is the date filter that is set on the **Date** field in the **Cust. Ledger Entry** table. The query returns the Remaining Amount for each customer. The results are displayed by document for the date specified in the filter and they are sorted by the customer number.

Important

.....

If you want to filter using a date range, you must use the `SetFlowFilter` predicate. However, when you use this predicate you can only specify one date range per filter and you therefore cannot set filters that specify more than one date range.

.....

SetFlowFilter

The `SetFlowFilter` predicate is maintained for backwards compatibility and to support date ranges.

```
{pred SetFlowFilter, '<TableName>', '<FieldName>', 'A searchString'}
```

The section in brackets is called the extended predicate.

Examples

The following example shows how to set a FlowFilter on the **Customer Filter** field in the **Currency** table:

```

SELECT * FROM Currency
WHERE {pred SetFlowFilter, 'Currency', 'Customer_Filter',
'10000..40000' }

```

`SetFlowFilter` is the name of the extended predicate, **Currency** is the table containing the filter, and `Customer_Filter` is the field being filtered. The expression '10000..40000' is the filter that will be set on the **Customer Filter** field in the **Currency** table.

The extended predicate always returns a value of TRUE, so if you use an OR expression with an extended predicate, the value of the entire expression will always be TRUE. For example, consider the following statement:

```

SELECT * FROM Currency
WHERE {pred SetFlowFilter, 'Currency', 'Customer_Filter',
'10000..40000' } OR
Last_Date_Modified > 01.01.04

```

This SELECT statement returns all the records in the table because the first condition in the WHERE clause, {pred SetFlowFilter, 'Currency', 'Customer_Filter', '10000..40000'}, is always TRUE. The OR operator has no effect.

Consider the following statement:

```
SELECT * FROM Currency
WHERE {pred SetFlowFilter, 'Currency', 'Customer_Filter',
'10000..40000' } AND
Last_Date_Modified > 01.01.04
```

This will return all records where the the customer number falls within the filter and where Last Date Modified is greater than 01.01.04.

INDEX

Symbols

% sign 9, 35

A

ADD SUB (operator) 22
add-in 2
aggregate function 30
All Characters (menu option) 9
All Except Dot (menu option) 9
All Except Space (menu option) 9
AND (operator) 22
AS (keyword) 30
ASC (keyword) 38
Auto (Windows Language) 8, 25
available data sources 5
a-z,A-Z,0-9,_ (menu option) 9

B

backslash 35
BETWEEN predicate 34
BLOB 20

C

C# 15
C++ 15
character values 23
clause
 FROM 30
 GROUP BY 36
 HAVING 37
 ORDER BY 38
 SET 42
 WHERE 34, 35, 46
client 2
client/server 6, 14
closing date support 9, 23
Company Name (field) 7
comparison
 operators 22
 rules 23
configuration 5
Connection (field) 6
Control Panel 11
CREATE TABLE statement 43
criteria 2

D

data source
 adding 10
 available 5
 changing 10
 deleting 11
 modifying 5
 name 5
 new 10

sample 4
 setting up 5
Data Source (field) 6
data type
 BIGINT 20
 BINARY 20
 BOOLEAN 20
 CODE 20
 DATE 20
 DATEFORMULA 20
 DATETIME 20
 DECIMAL 20
 DURATION 20
 GUID 20
 INTEGER 20
 OPTION 20
 SQL_BINARY 20
 SQL_DATE 20
 SQL_INTEGER 20
 SQL_TIME 20
 SQL_TIMESTAMP 20
 SQL_VARCHAR 20
 TEXT 20
 TIME 20
database
 name 6

- Database Name (field) 6
- DBMS Cache (KB) (field) 8
- decimals 22
- DELETE statement 41
- DESC (keyword) 38
- Description (field) 6
- drop down list 8
- DROP TABLE statement 45

- E**
- EQ (operator) 22
- evaluate 22
- expression 22
- extended predicate 47

- F**
- filter 2
- FlowField 46
- FlowFilter 46
- FROM clause 30

- G**
- GE (operator) 22
- GROUP BY clause 36
- GT (operator) 22

- H**
- HAVING clause 37

- I**
- ID 7
- identifier
 - quoted 10
 - space in 9
 - symbol in 9
 - transfer 9
- Import License (button) 7
- IN predicate 34
- INSERT 39
- insert
 - triggers 39, 41, 42
- Installation of Dynamcis NAV ODBC 4
- Integer (menu option) 8
- interface 14

- K**
- keyword
 - ASC 38
 - DESC 38
 - DISTINCT 30

- L**
- language 25
- Language (field) 8
- LE (operator) 22
- LIKE predicate 35
- Local (setup) 6
- logging in 7
- LT (operator) 22

- M**
- macro 2
- Microsoft Excel 2
- Microsoft Query 2, 10
- MUL DIV (operator) 22
- multilanguage 8, 24
- multiuser 6

- N**
- name
 - company 7
 - database 6
 - server 6
- NE (operator) 22
- Net Type (field) 6
- network program 6
- Neutral 8
- non-padded 23

- O**
- Open DataBase Connectivity 2
- operator
 - () 22
 - ADD SUB 22
 - AND 22
 - EQ 22
 - GE 22
 - GT 22
 - LE 22
 - LT 22
 - MUL DIV 22
 - NE 22
 - OR 22
- operator precedence 22
- Option (field) 8
- optional parameter 28
- OR (operator) 48
- ORDER BY clause 38

- P**
- parentheses 23
- Password (field) 7
- percent sign (%) 35
- precision 22
- predicate
 - BETWEEN 34
 - IN 34
 - LIKE 35

predicate, extended	47
Q	
quoted identifiers	10
R	
RECORDID	20
S	
sample data source	4
security	7
SELECT statement	30, 35
server	6
Server (setup)	6
Server Name (field)	6
SET clause	42
setup	10
sign	35
single-user	6
spreadsheet	2
SQL	2
writing statements	10
SQL_DECIMAL	20
SQL_VARCHAR	46
statement	
CREATE TABLE	43
DELETE	41
DROP TABLE	45
INSERT	39
SELECT	30
UPDATE	42
T	
TABLEFILTER	20
Text (menu option)	8
text string	8
triggers during insert	39, 41, 42
U	
UPDATE statement	42
User ID	7
User ID (field)	7
V	
Visual Basic	2
Visual Basic .NET	15
W	
WHERE clause	34, 35, 38, 46
word processor	2

