Microsoft Business Solutions

C/ODBC Guide

MICROSOFT BUSINESS SOLUTIONS–NAVISION

C/ODBC Guide

DISCLAIMER

This material is for informational purposes only. Microsoft Business Solutions ApS disclaims all warranties and conditions with regard to use of the material for other purposes. Microsoft Business Solutions ApS shall not, at any time, be liable for any special, direct, indirect or consequential damages, whether in an action of contract, negligence or other action arising out of or in connection with the use or performance of the material. Nothing herein should be construed as constituting any kind of warranty.

COPYRIGHT NOTICE

TRADEMARK NOTICE

Published by Microsoft Business Solutions ApS, Denmark.

Published in Denmark 2003.

DocID: NA-370-DVG-004-v01.00-W1W1

PREFACE

This book is a manual for Microsoft® Business Solutions–Navision®. This book is part of a comprehensive set of documentation and Help materials for the Navision enterprise business solution.

The manual describes how to install and run C/ODBC. It also contains a reference guide to SQL statements.

You should also be familiar with the symbols and typographical conventions used in the Navision 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.
Department	Names of windows, boxes and tabs. They appear in bold italics and start with a capital letter.
Hansen	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

Table of Contents

Chapter 1
Introduction

This chapter gives a brief overview of this manual and C/ODBC, including definitions of technical terms used in C/ODBC.

The chapter contains the following section:

- What is C/ODBC?

1.1 What is C/ODBC?

C/ODBC is the implementation of Open DataBase Connectivity (ODBC) for Microsoft® Business Solutions–Navision®. C/ODBC lets you transfer data between a Navision database and any program that supports ODBC.

C/ODBC functions largely in the same way as ordinary clients in Navision.

The main differences are:

- a C/ODBC client is a program other than Navision, such as a spreadsheet or word processing program.
- triggers are not run when a C/ODBC client writes data in a Navision database.

An explanation of how C/ODBC works with a Navision database can be found on page 13.

Special Terminology

There are a number of terms used in this book that are not used in Navision. Below 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 Navision.

Chapter 2
Installation and Configuration

This chapter explains how to install and set up C/ODBC.

The chapter contains the following sections:

- Installation
- Configuration

2.1 INSTALLATION

Before you install C/ODBC, you must install Navision.

To install the C/ODBC driver under Windows NT, Windows 98 or Windows 2000, follow this procedure:

1 Start the C/ODBC setup program. You find this in the CODBC subfolder on the Navision product CD.

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

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

The necessary files are copied to the windows\system folder on Windows 98 computers, winnt\system32 on Windows NT and Windows 2000 computers, C/ODBC is registered and a sample C/ODBC data source is installed. The installation is finished when the ***Setup Completed*** window appears.

Cancelling the Installation

The installation can be cancelled 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. This rollback functionality is a new feature provided by Windows Installer.

Uninstalling and Repairing C/ODBC

Windows Installer is also used for repairing and removing C/ODBC.

1 In the control panel, select Add/Remove Programs.

2 Select CODBC.

3 Choose to repair or remove C/ODBC.

Windows Installer will repair or remove C/ODBC automatically, depending on your choice.

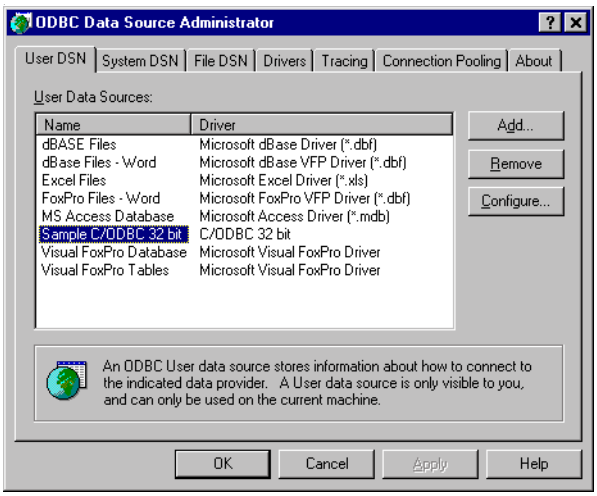
2.2 CONFIGURATION

After you have installed the necessary files, 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

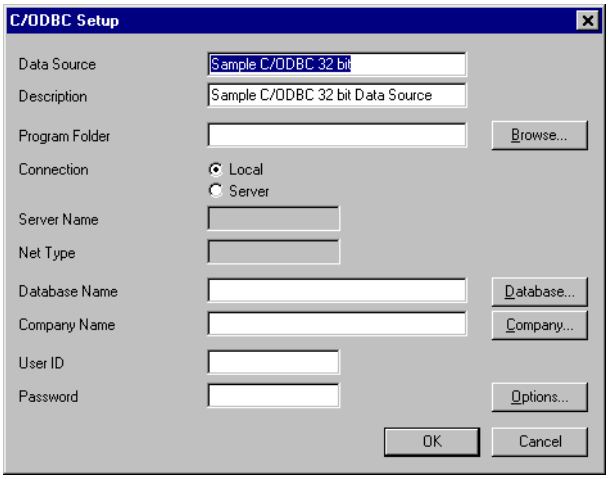
To set up a user data source, follow this procedure:

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 (in parentheses). You can read about adding, changing and deleting data sources on page 11, and the online Help for the ODBC Data Source Adminstrator explains about User, System and File Data Sources.

2 Click Sample C/ODBC 32 bit, and click Configure. The C/ODBC ***Setup*** window appears:



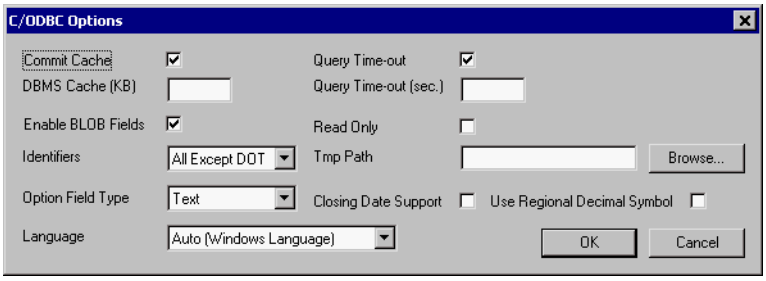
Fill in the fields according to these guidelines:

Field	Comments
Data Source	The field is already filled in with a default data source name. This name can be changed, as explained on page 11.
Description	Enter a description of the data source.
Program Folder	If Navision has been installed using the setup program, the Windows registry will contain the necessary information for the C/ODBC driver to locate the program files, and you do not have to fill in this field. Otherwise (for example, if you have moved the program files after the installation), click **Browse** and select the correct folder.
Connection	You must specify here whether Navision is installed as a single-user or multiuser system (client/server): *Local* — Click here if the driver will function in a single-user installation. *Server* — Click here if the driver will function in a multiuser installation. The default value is Local.
Server Name	If you have clicked the **Server** field, enter the name of the server, that is, the server where the Navision database is located. (If you have clicked the **Local** field, leave this field empty.)
Net Type	In a multiuser installation, enter the name of the network protocol program, that is, *tcp* (for TCP/IP) or *netb* for NetBios.

Field	Comments
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. If a database consists of several parts (files), you must enter all the file names, separated by plus signs (+). That is, enter a plus sign (+) after a file name in the **Database Name** field before typing another file name.
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.
User ID	Enter the user ID that you use when logging in.
Password	Enter the password for the user ID. There will usually be a special user ID and password set up for C/ODBC – 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.) If you do not write anything, you will have to enter an ID and password every time you want to open the database from another program. You can read more about security in the manual *Installation and System Management: Microsoft Business Solutions–Navision Database Server* or *Installation and System Management: Microsoft Business Solutions–Navision SQL Server Option*.

Specifying Options

In the C/ODBC ***Setup*** window, click the Options button to specify the data source options. The C/ODBC ***Options*** window appears:



Fill in the fields according to these guidelines:

Field	Comments	
Commit Cache	Specifies whether commit cache should be used:	
	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	Specifies whether BLOB fields should be visible from ODBC:	
	Yes (checked)	BLOB fields can be seen from ODBC.
	No (unchecked)	BLOB fields are hidden.
Identifiers	In this field, click the AssistButton® to select one of four options. The options in this field and their implications are described in detail on page 9.	
Option Field Type	Click the AssistButton to choose one of two options:	
	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.
Language	When you click the AssistButton, all languages that your license file gives you access to will be shown in the **Language** list box. Click the AssistButton to select one of the following options:	
	Language Neutral	This options disables multilanguage functionality. C/ODBC 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.
Query Time-out	Click the field to enter a ✓ if you want to enable the time-out facility. If the facility is enabled, and a specified time elapses while the ODBC driver is running a query, the driver pauses and prompts you to abort or resume a query.	
Query Time-out (sec.)	Enter the number of seconds to allow before the driver pauses. If you do not enter anything, a default value of 120 seconds is used.	

Field	Comments	
Read Only	Specifies whether access to the database stored on a Navision Database Server should be read-only.	
	Yes (checked)	Access is read-only.
	No (unchecked)	Access is read/write.
Tmp Path	Enter the name of the folder that will be used to store temporary files. You can browse through the folders on the computer by clicking **Browse**. If a temporary path is not entered, the folder to use will be decided according to this scheme: First choice is the contents of TMP environment variable. If this variable is not defined, then the second choice is the contents of the TEMP environment variable. If this is not defined either, then the current folder will be used.	
Closing Date Support	Specifies whether the connection supports closing dates.	
	Yes (checked)	Closing dates are supported.
	No (unchecked)	Closing dates are not supported.
Use Regional Decimal Symbol	Specifies whether or not the regional decimal symbol should be used.	
	Yes (checked)	The regional decimal symbol is used.
	No (unchecked)	The regional decimal symbol is not used. This is the default value and ensures that '.' is used as the decimal symbol.
	This option ensures compatibility with older applications, such as Access 97, that are used to display data that is extracted with C/ODBC. This option must not be used with newer applications because the decimal symbol must always be '.'.	

Converting Identifiers

The options in the **Identifiers** field control the way identifiers (table names and field names) are transferred from Navision 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
All Characters	Letters, numbers, symbols, dots and spaces are transferred unchanged.
All Except Space	Letters, numbers and symbols are transferred unchanged. Spaces, dots and question marks are converted to underscores (_).
a-z,A-Z,0-9	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.
All Except DOT	Letters, numbers, symbols and spaces are transferred unchanged. Dots and question marks are converted to underscores (_).

EXAMPLE

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

Field Name	All Characters	All Except Space	a-z,A_Z,0-9	All Except DOT
No.	No.	No_	No_	No_
Sales (LCY)	Sales (LCY)	Sales_(LCY)	Sales__LCY_	Sales (LCY)
Profit %	Profit %	Profit_%	Profit_Pct	Profit %
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 C/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 10 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 (User, System and File).

To set up a new user data source:

1. In the control panel, click Data Sources (ODBC).
2. In the ***ODBC Data Source Administrator*** window, click the User DSN tab and then click Add.
3. Select the C/ODBC driver, and click OK.
4. Enter information in the C/ODBC ***Setup*** window as described on page 6.
5. Click OK to close the window.

Adding a File Data Source

Microsoft Query can only use a file data source. You can create a file data source, based on an existing data source, like this:

1. In the control panel, click Data Sources (ODBC).
2. In the ***ODBC Data Source Administrator*** window, click the File DSN tab and then click Add.
3. Select the C/ODBC driver, and click OK.
4. Enter a name for the file in which the data source is saved (for example, `codbc.dsn`).
5. Click Finish to close the window.

Now you will see the same window as when you added the user data source. The data source could be the name of an existing user data source (for example: Sample C/ODBC 32 bit). When you have entered the necessary information (in the same way as when adding a user or system data source), you will have a file data source that can be used with, for example, Microsoft Query.

Changing a Data Source

The information in the data source can be changed at any time. This is how to do it:

1. In the control panel, click Data Sources (ODBC).

2 Select the data source you want to change, and click Setup.

3 Change the necessary fields in the C/ODBC ***Setup*** window by typing or selecting the new names or values.

4 Click OK to close the window.

Deleting a Data Source

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

1 In the control panel, click 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 C/ODBC Database Driver makes data in a database stored locally or in a Navision Database Server accessible to ODBC-enabled applications. You can use the ODBC driver to retrieve Navision data into an application such as a word processor or spreadsheet.

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

The chapter contains the following sections:

- Overview
- Establishing a Connection
- C/ODBC 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 98, Windows NT and Windows 2000 environments. Applications using the ODBC interface can work with many different database systems.

C/ODBC opens a Navision Database Server or local database to ODBC-enabled applications so that they can retrieve data from the database.

The ODBC driver operates in the Windows 98, Windows NT and Windows 2000 environments. In these environments, it can function either as a stand-alone or as a client in a client/server configuration.

Note that you cannot use C/ODBC with SQL Server Option for Navision.

3.2 ESTABLISHING A CONNECTION

The SQLConnect and SQLDriverConnect functions in the C/ODBC driver contain the following parameters:

CSF Specifies whether the driver operates as a client in a client/server environment or as a stand-alone.

- Yes (Client in a client/server environment)
- No (Stand-alone)

NType The name of the network protocol module (*tcp* or *netb*).

SName The name of the server host computer.

Database The name(s) of the database file(s).

CS The cache size in KB.

CN The account/company to open. If the company name has leading spaces, it must be enclosed in quotes (as in " My Company").

OPT Specifies how the contents of a Navision option field are transferred to an application.

- Text
- Integer

UID The user ID for logging in.

PWD The password for the user ID (UID).

PPath The name of the folder where the program files are located.

TP The name of the folder that will be used to store temporary files.

QT The number of seconds to elapse before time-out.

QTYesNo Enables or disables query time-out:

- Yes (enable)
- No (disable)

IT Specify the way identifiers are returned to an external application:

- All Characters
  Letters, numbers, symbols, dots and spaces are transferred unchanged.
- All Except Space
  Letters, numbers and symbols are transferred unchanged. Spaces, dots and question marks are converted to underscores (_).
- a-z,A-Z,0-9
  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.

- All Except DOT
  Letters, numbers, symbols and spaces are transferred unchanged. Dots and question marks are converted to underscores (_).

RO Specifies whether access to the Microsoft Business Solutions database should be read-only.

- Yes (access will be read-only)
- No (access will be read/write)

CC Specifies whether the commit cache should be used.

- Yes (use commit cache)
- No (do not use commit cache)

BE Specifies whether BLOB fields should be visible from ODBC.

- Yes (BLOB fields can be seen from ODBC)
- No (BLOB fields are hidden)

CD Specifies whether the connection supports closing date.

- Yes (closing date support)
- No (No closing date support)

ML Specifies if the user wants to choose a language other than the language one global language. You must use the four-digit Windows Language ID, for example ML=1033 for English (United States).

These parameters are normally set by ODBC Admin and need not be specified by the connection function.

The driver retrieves the setup from the operating system’s registry and overwrites it with the values in the connection string.

The following lines are an example of a complete connection string.

```
DSN=My Source;PPath=C\FIN:;TP=C:\WINDOWS\TEMP;CSF=No;
Database=C:\FIN\DATABASE.FDB;
CN=My Company;UID=;PWD=;CS=100;OPT=Text; QTYesNo=Yes;QT=200;IT=All
Characters
RO=No;CC=Yes;BE=No
```

3.3 C/ODBC FUNCTIONALITY

C/ODBC has the functionality required for it to be used as a read/write data source.

The SQL conformance level is the CORE level.

The API conformance level is most of Extension Level 1. The driver does not support all the options defined as comprising ODBC functionality.

3.4 DATA TYPES

The C/ODBC driver supports the following SQL data types:

Navision Data Types	SQL Data Types
BOOLEAN	SQL_INTEGER
BINARY	SQL_BINARY
INTEGER	SQL_INTEGER
BIGINT	SQL_VARCHAR
OPTION	SQL_INTEGER / SQL_VARCHAR
DATE	SQL_DATE / SQL_TIMESTAMP
TIME	SQL_TIME
DATETIME	SQL_TIMESTAMP
DATEFORMULA	SQL_VARCHAR
DURATION	SQL_VARCHAR
CODE	SQL_VARCHAR
TEXT	SQL_VARCHAR
DECIMAL	SQL_VARCHAR / SQL_DOUBLE
MEMO	SQL_VARCHAR
USERDEF	SQL_LONGVARCHAR
BITMAP	SQL_LONGVARCHAR
GUID	SQL_VARCHAR

If possible, the driver converts one data type to another. All types can be converted to SQL_VARCHAR.

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

Navision Data Types	Rules
BOOLEAN	Enter either the string (‘No’ or ‘Yes’) or the associated value (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	Option fields can be entered either as the option string (in single quotes) or as the option value. Thus, if the first option string is Open, this value can be entered either as *Open* or as *0* (zero), because option strings in a set are numbered from 0 (zero) and upwards.

Navision Data Types	**Rules**
DATE	Enter a date in this format: {*d 'yyyy-mm-dd'*} where y=year (0000-9999), 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 21.
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 *Standard date time unit* where you can set the standard unit of measure. C/ODBC users must know this unit of measure if they enter data as a number, since Navision 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.
DECIMAL	Enter a number (without quotes). The decimal precision of C/ODBC is 18 giving you a range from -999 999 999 999.99999 to +999 999 999 999.99999. Note that C/ODBC always uses 5 decimal places no matter which scale the underlying Navision database uses.
DATEFORMULA	Enter a date the same way as with the DATE datatype. The information will be stored independently of language settings.
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.

If the name of a table field corresponds to an SQL reserved word, an underscore (_) is automatically appended to the name of the table field in order to resolve the ambiguity. For example, Order would be changed to Order_.

Comparison Operators

C/ODBC 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
!=	Not equal to

Operator Precedence

An important property of an operator is its precedence. Precedence determines the order in which C/ODBC 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 table below 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
7.	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, C/ODBC supports closing dates.To support closing dates, in the C/ODBC ***DSN Option*** window, place a check mark in the **Closing Date Support** field. 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 59:59:59'*}. 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 Support* option is disabled.

Note that even with the use of the SQL statement `{ts '2001-01-01 59:59:59'}`, fields with the DATE data type do not store timestamps. To store the actual timestamp, 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 Values in Option Fields

The comparison operators (see page 20) operate on the option *values*, not the option *strings*. Thus, when two option strings, called ‘Open’ and ‘Close’ are compared, the result of the comparison depends on the options values of these strings. If Open is 0 (zero) and Close is 1, ‘Open’ would be considered less than ‘Close’ (Open < Close).

3.5 Multilanguage Functionality

C/ODBC has been changed to handle the multilanguage functionality in Navision. It is now possible for the C/ODBC user to retrieve the application data from Navision in different languages independent of the current Navision application language. In order to make this possible, you must ensure that you are using the new `codbc.dll` after making a backup of the original file. Normally you will find the `codbc.dll` file in the ...\Program Files\Common Files\Navision\CODBC folder on your computer.

When you are running the correct `codbc.dll` and you select open the ***ODBC Data Source Administrator*** window from the operating system's control panel, you can set up C/ODBC as described on page 5. Use the **Language** field properties to set up the connection to suit the user the best.

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

C/ODBC covers the following multilanguage features:

- Table name
- Field name
- OptionString value
- Date Formula

When you link a table by setting another application language than the default language and this language has a corresponding output from Navision, you will notice that the value of the table name, all the field names, and the option fields within that table will be shown in the chosen language.

Limitations

You will not be able to create the table with multilanguage *Caption* properties using C/ODBC. This means that no matter what language has been chosen in the C/ODBC ***Options*** window, the table that is created for any C/ODBC application will be using 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. C/ODBC can only accept one setting at the time of loading. To switch language when using

C/ODBC, you must to close the C/ODBC connection and thereby release it from the memory, change to the preferred language in the C/ODBC ***Options*** window, and start the C/ODBC connection again.

Scenarios

There are three different scenarios that are possible when running multilanguage for C/ODBC. 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)

Language Neutral

In the first scenario, the global language of the Navision application is English (United Kingdom) and the **Language** field in the choice in the C/ODBC ***Options*** window is set to *Language Neutral*.

C/ODBC Result

The application data retrieved is shown in English (United States).

Auto (Windows Language)

In this scenario, the global language of the Navision application is still English (United Kingdom) but the **Language** field in the choice in the C/ODBC ***Options*** window is set to *Auto (Windows Language)*.

C/ODBC Result

The application data retrieved is shown in German (Austrian) if the selected objects have strings with the language code for German (Austrian).

If the selected objects do not have strings with the language code for German (Austrian), the application data retrieved is shown in German (Standard).

If the selected objects do not have strings with the language code for German (Standard) either, the application data retrieved is shown in English (United Kingdom).

Specific Language

In this scenario, the global language of the Navision application is still English (United Kingdom) but the **Language** field in the choice in the C/ODBC ***Options*** window is set to a specific language.

C/ODBC Result

If the **Language** field in the choice in the C/ODBC ***Options*** window is set to German (Austrian), the application data retrieved is shown in German (Austrian).

If the **Language** field in the choice in the C/ODBC ***Options*** window is set to German (Standard), the application data retrieved is shown in German (Standard).

If the **Language** field in the choice in the C/ODBC ***Options*** window is set to German (Austrian) but the selected objects do not have strings with the language code for German (Austrian), the application data retrieved is shown in German (Standard).

If the **Language** field in the choice in the C/ODBC ***Options*** window is set to German (Austrian) but the selected objects do not have strings with the language code for neither German (Austrian) not German (Standard), the application data retrieved is shown in English (United States).

Date Formula

You can enter dates as two different data types:

- DATE
- DATEFORMULA

If you want dates to be multilanguage enabled, use DATEFORMULA.

When a date calculation formula is stored in a DateFormula field, it is converted to a generic, nonlanguage dependent format. When a date calculation formula is retrieved from a DateFormula field, it is converted to a valid date conversion string for the currently selected language.

Appendix A
SQL Statement Reference Guide

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

The appendix contains the following sections:

- Introduction the Reference Guide
- SELECT Statement
- Parameter Markers
- Predicates in WHERE Clauses
- GROUP BY Clause
- HAVING Clause
- ORDER BY Clause
- INSERT Statement
- DELETE Statement
- UPDATE Statement
- CREATE TABLE Statement
- DROP TABLE Statement
- Navision FlowFields

A.1 Introduction 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

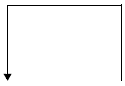
The following graphical conventions are used:

Indicates the beginning of a statement.

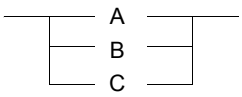
Indicates that the statement syntax is continued on the next line.

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

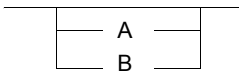
Indicates the end of a statement.



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

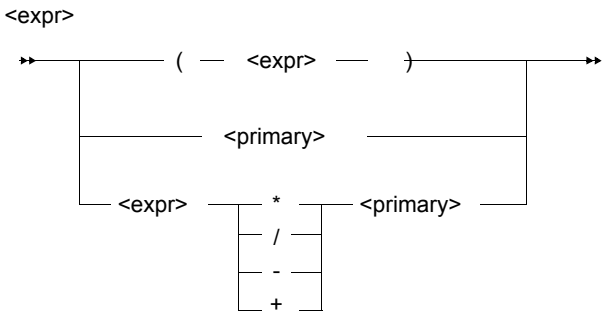
and ends with

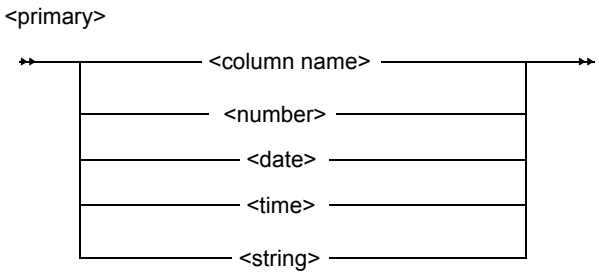
Uppercase letters denote keywords. Note that this convention is used only to make the syntax diagrams easier to read: the SQL reserved words (keywords) are not case-sensitive.

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 C/ODBC:

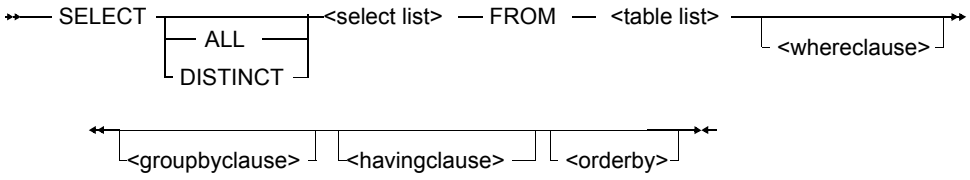




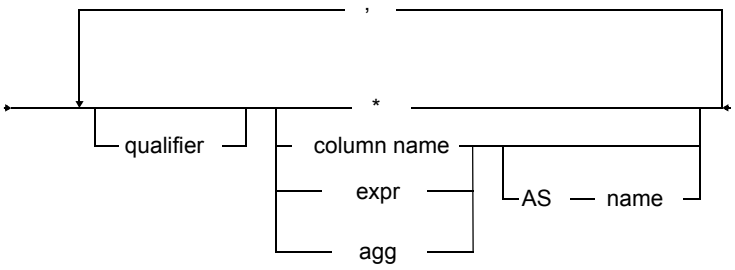
A.2 SELECT STATEMENT

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 (default is ALL).

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.

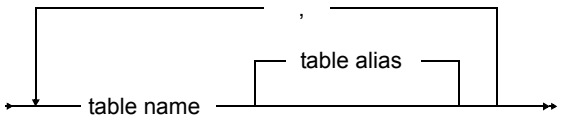
column name: the specific column.

expr: a field that contains an expression, with or without a column name. See page 20 for a diagram of expression syntax.

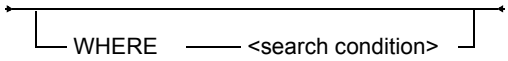
agg: an aggregate function. There are these aggregate functions: COUNT(* | expr), AVG(expr), MAX(expr), MIN(expr), SUM(expr).

AS: a keyword that permits the column to be aliased.

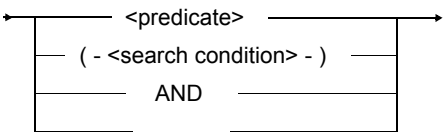
<tablelist> The <tablelist> lists the tables (and aliases) used in the SELECT statement.



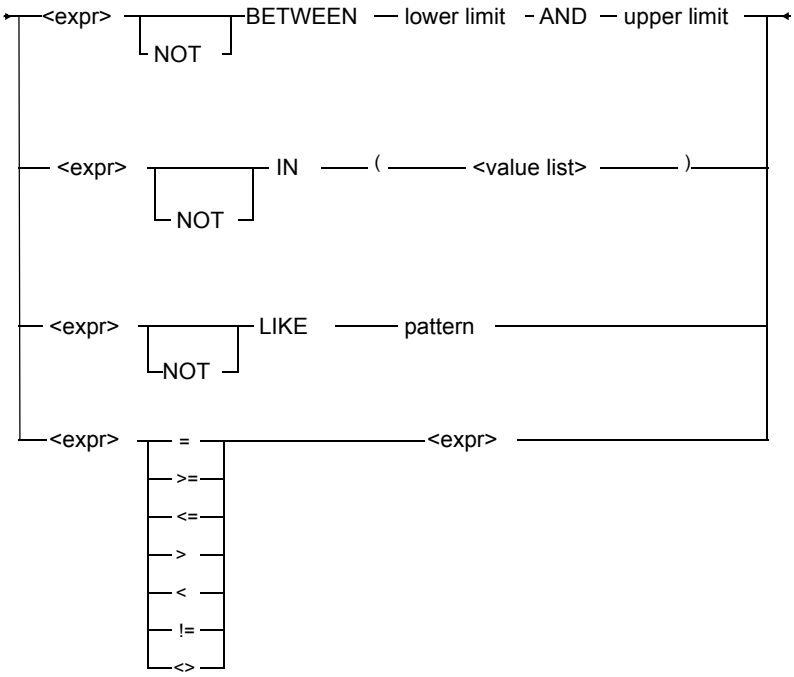
<whereclause> The <whereclause> specifies the search condition against which the rows in the <tablelist> are evaluated.



<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 order, it is not a good solution for an order 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:

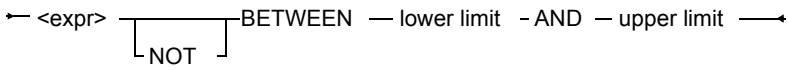
- In a SELECT list
- As both expressions in a comparison-predicate
- As both operands of a binary operator
- As both the first and second operands of a BETWEEN operation
- As both the first and third operands of a BETWEEN operation
- As both the expression and the first value of an IN operation
- As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

A.4 Predicates in WHERE Clauses

WHERE Clause BETWEEN Predicate

Compares a value to a range of values.

Syntax



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

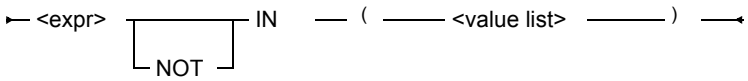
This example retrieves all customers with a post code in the range of 1000 to 1234:

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

WHERE Clause IN Predicate

Compares a value against a list of values for equality.

Syntax



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 postcode IN ('1000','2000','3000')
```

WHERE Clause LIKE Predicate

Compares a string value against a pattern for equality.

Syntax

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. An underscore (_) matches any single character in the same position. A percent sign (%) matches any number of characters including zero characters in the same position.

If you want to include an underscore or 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%'
```

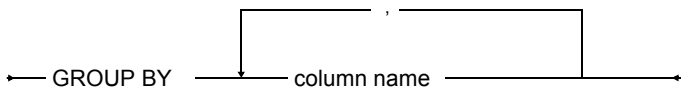
The next example retrieves all customers whose name has at least three letters (the underscore consists of three individual underscore characters):

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

A.5 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 28 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

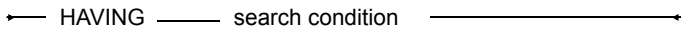
This example retrieves the number of customers per country. We join the ***Customer*** and the ***Country*** tables in order to get the names of the countries instead of the country codes:

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

A.6 HAVING CLAUSE

Specifies conditions for including groups in the output.

Syntax



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 29 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
```

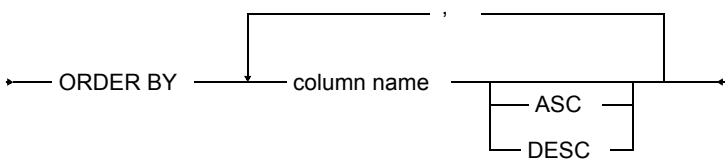
This example retrieves information from the Sales Invoice Header table. It produces a list of customers and the average, maximum and minimum amounts of their sales invoices, but only if there is more than one invoice for the customer. This statement could not be rewritten using WHERE.

```
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.7 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

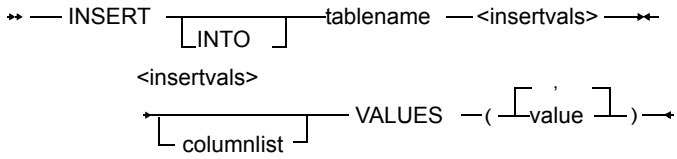
This example retrieves all customers sorted by post code and name:

```
SELECT * FROM Customer ORDER BY "Post Code", name
```

A.8 INSERT STATEMENT

Inserts rows into a table.

Syntax



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 C/ODBC.

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 21.
- 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 data as undefined data, 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 the null value implicitly.

Example

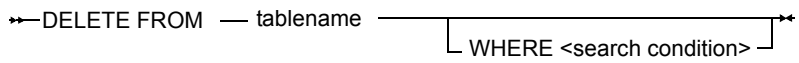
This example inserts a record in the Country table. The column list contains only two of the columns of the table. The rest of the columns will be inserted as null values.

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

A.9 DELETE STATEMENT

Deletes rows in the specified table.

Syntax



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 C/ODBC.

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 28).

Example

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

```
DELETE FROM Customer
```

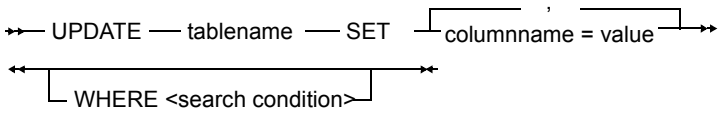
This example deletes a single customer from the ***Customer*** table:

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

A.10 UPDATE STATEMENT

Updates rows in a table.

Syntax



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 record 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 C/ODBC.

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 28).

Examples

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

```
UPDATE Item SET "Unit Price" = "Unit Price" * 1.25
```

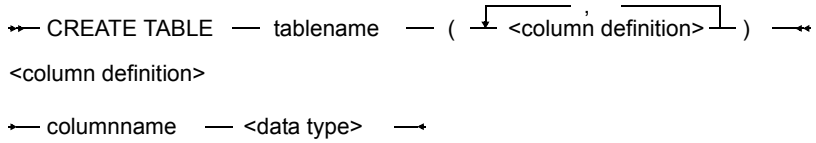
This example updates the **Country Code** field in selected records of the ***Customer*** table:

```
UPDATE Customer SET "Country Code" = 'CN'
WHERE "Country Code" = 'HK'
```

A.11 CREATE TABLE STATEMENT

Creates a table.

Syntax



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 or more numeric arguments. 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®). For exact numeric data types, you can specify precision and scale. Precision determines the number of digits that can be accommodated in the column. Scale denotes how many of these digits can be after the decimal point.

When a table is created, C/ODBC will generate 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 C/ODBC and the C/SIDE data types.

C/ODBC Type	C/SIDE Type	Comments
BCD	Decimal	Formally, this data type takes two arguments: precision (length) and scale (decimals), but the arguments are not used in C/SIDE for the Decimal format. Because the arguments are discarded, it does not matter what values you give them (but they must be present).
BITMAP		

C/ODBC Type	C/SIDE Type	Comments
BOOL	Boolean	
CODE	Code	Takes one argument: length.
DATE	Date	
MEMO		
OPTION	Option	
S32	Integer	
STRING	Text	Takes one argument: length.
TIME	Time	
USERDEFINED		

Example

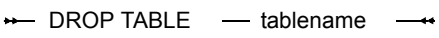
This example creates a table with three fields: an Integer, a Decimal and a Text field:

```
CREATE TABLE Sample (
   Code S32,
   Value BCD(12,4)
   Name STRING(50)
)
```

A.12 DROP TABLE STATEMENT

Drops a table from the database.

Syntax



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.13 NAVISION FLOWFIELDS

Navision has a special field type called a FlowField, which contains values 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 Navision 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 Navision.

Setting a FlowFilter on a FlowField is done as a work-around in the WHERE clause. The syntax is: FlowFilter field = ‘string’.

```
SELECT * FROM G_L_Account WHERE (Date_Filter = '010191..010192')
```

The WHERE element containing the FlowFilter is evaluated as TRUE.

```
SELECT * FROM G_L_Account WHERE (Date_Filter = '010191..010192') OR
(Name = 'Hansen')
```

The SELECT statement above returns all the records in the table because the WHERE clause `(Date_Filter = '010191..010192')` is always TRUE. The OR operator has no effect.

The statement should be written as follows:

```
SELECT * FROM G_L_Account WHERE (Date_Filter = '010191..010192') AND
(Name = 'Hansen')
```

It returns all records where Name equals “Hansen”. The FlowFilter field is Date_Filter, and the FlowFilter applied to the field is “010191..010192”.

INDEX

Index