MODULE 2: TABLES

Module Overview

Tables are the most fundamental objects in Microsoft Dynamics NAV. They store records that are collected through pages, for example customers, sales, and inventories. These records are then presented to users through pages and reports.

Objectives

The objectives are:

- Explain the concepts of tables and table components.
- Examine the concept behind primary and secondary keys, and explain how to set them.
- Create a simple table with primary and secondary keys, and add data to the table.
- Review the concept of table relation.
- Set table relations with a filter and condition.
- Describe the special table fields.

Use special table fields to improve table features.

Table Fundamentals

Records in Microsoft Dynamics NAV databases are stored in tables. A table can be visualized as a two-dimensional matrix, consisting of columns and rows. Each row is a single record, and each column is a field in that record.

A table consists of table data and the table description. The table data contains the actual records and their fields. The table description specifies the layout and properties of the table and the fields.

The table description is not directly visible to the user. When a developer creates a table, he or she assigns the table name, ID number, and fields. This establishes the table description.

Field characteristics, such as field name, ID number, data type, and initial value—together with the primary and secondary keys (used to sequence data)—are also part of the table description.

The table description contains properties, triggers, fields, and keys. The "Table Components and Their Relation" figure shows components of the table description and how they are related.

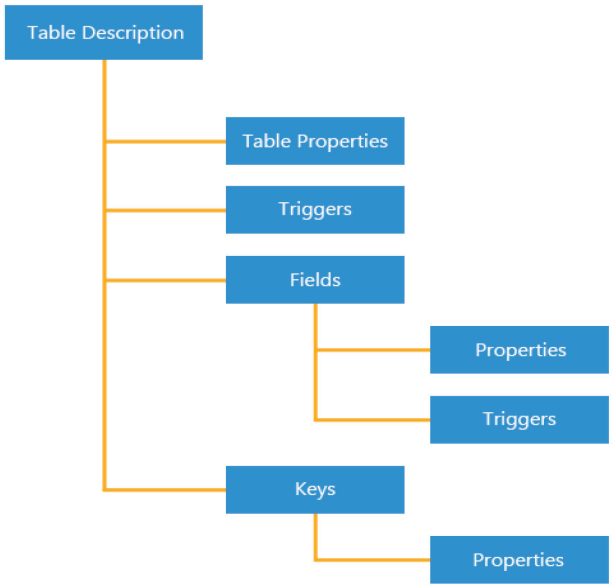

FIGURE 2.1: TABLE COMPONENTS AND THEIR RELATION

Properties

The table description contains some properties that are related to the table and those that are related to the fields or the keys in the table.

When a table is created, the development environment automatically defines several default values for table properties. Depending on the table's purpose and its relation to other application objects, these default values might have to be changed.

Some standard table properties—such as **ID**, **Name**, and **Caption**—differentiate from one table to another. The **Permissions** property establishes specific permission for users. Other examples of table properties include **LookupPageID** and **DrillDownPageID**, which specify the page ID that is used to look up and drill-down information in the table.

You can view or modify table properties from the **Properties** window, which is accessed from the Table Designer. The following steps show how to open the **Properties** window for the **Customer** table and examine several table properties.

Follow these steps in the Microsoft Dynamics NAV Development Environment:

1. On the **Tools** menu, click **Object Designer**.
2. Click **Table** to open the Table list.
3. Select table **18**, **Customer**, and then click **Design** to design the **Customer** table.

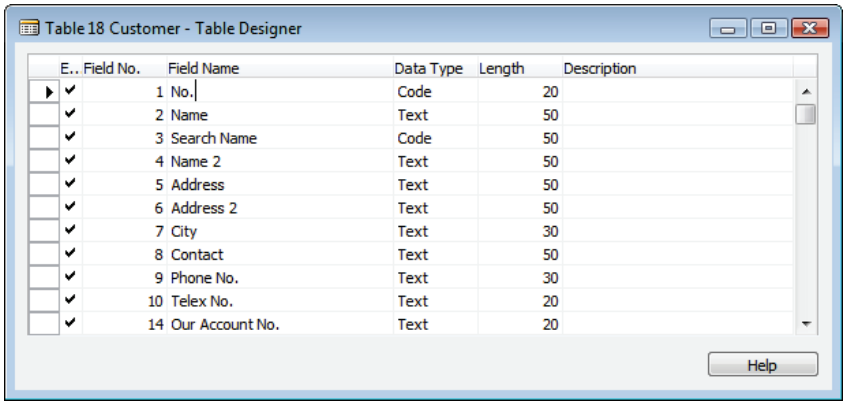

FIGURE 2.2: CUSTOMER TABLE IN TABLE DESIGNER

4. Scroll down and put the pointer on an empty line at the bottom of the Table Designer.
5. Click **View > Properties**, or click the **Properties** button on the Toolbar. The **Table - Properties** window opens and shows the properties for the table. Developers can then view and modify properties for the Customer table.

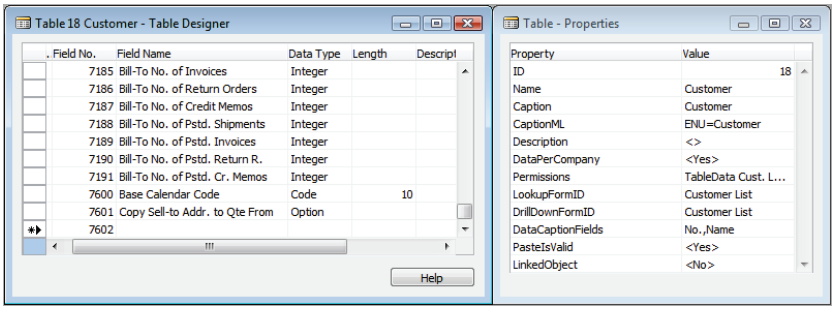

FIGURE 2.3: CUSTOMER TABLE AND ITS PROPERTIES

6. Locate the **DataPerCompany** property.
   The value of this property is <Yes>. The angle brackets (<>) indicate that Yes is a default value that is assigned to this property when the table is created and it has not been changed.

Note: *To modify the value of a property, select or type a new value in the* ***Value*** *column, and then update the property by either pressing* ***Enter*** *or moving the pointer away from the field.*

---

7. Locate the **LookupPageID** property.
   The value of this property is Customer List. This property specifies that the **Customer List** page is used to look up values in the **Customer** table.
8. Select the **DrillDownFormID** property and press **F1**.
   The Microsoft Dynamics NAV Help opens and shows a Help topic for the **DrillDownPageID** property.

Triggers

Triggers are predefined functions that start when certain events occur. The bodies of these functions are first empty and must be defined by the developer. Defining C/AL code in triggers lets developers change the system's default behavior.

Triggers in a table can be divided into two categories:

- Table triggers
- Field triggers

Examples of table triggers include **OnInsert**, which contains statements that implement after a new record is inserted into the table. The **OnModify** trigger contains statements that implement when a record in the table is modified. Triggers in a table are edited in the C/AL Editor, which you can access from the Table Designer.

Fields

A table must at least have one field. Fields define the actual information that is kept in a table. Each field has its own properties and triggers.

Each field has an appropriate data type. Each data type holds a specific kind of information, such as text, numbers, dates, and so on. The following table shows most common data types that are available in Microsoft Dynamics NAV.

Data Type	Description
BigInteger	This data type is a 64-bit integer. It is used to store large whole numbers.
Binary	This data type contains binary data. The binary data is stored in the record. The corresponding SQL data type is **VARBINARY**.
BLOB	A Binary Large Object (BLOB) is used to store bitmaps and memos. The BLOB is not stored in the record, but in the BLOB area of the table. The corresponding SQL data type is **IMAGE**.
Boolean	This data type indicates the values of **TRUE** or **FALSE**. When formatted, a Boolean field displays as **Yes** or **No**. The corresponding SQL data type is **TINYINT**.
Code	This data type represents a special type of alphanumeric string that is right-justified if the contents are numbers only. If letters or blanks occur among the numbers, the contents are left-justified. All letters are converted to uppercase upon entry. The field must be defined to be between 1 and 250 characters. The **SQL Data Type** property can be used to indicate whether code fields can contain integers or text strings. The corresponding SQL data type is **NVARCHAR**.
Date	This data type indicates a date value in the range from January 1, 1753 to December 31, 9999. An undefined date is expressed as 0D. All dates have a corresponding closing date. The system considers the closing date for a given date as a period that follows the given date but comes before the next regular date. That is,, a closing date is sorted immediately after the corresponding regular date but before the next regular date. The corresponding SQL data type is **DATETIME**.

Data Type	Description
DateFormula	This data type is used to contain a date formula that has the same capabilities as an ordinary input string for the **CALCDATE** Function (**DATE**). The following are examples of DateFormula: • 30D (30 days) • CM+1M (current month plus one month) • D15 (the 15th of each month) ***Note:*** *The date part indicators in DateFormula fields, such as D for day or M for Month, are always displayed and users must enter them in the currently selected language. English language uses D, W, M, Q, Y, and C to indicate day, week, month, quarter, year, and current, respectively, but other languages use different indicators, typically corresponding to the first letter of the time measurement unit in the respective language.*
DateTime	This data type represents a point in time as a combined date and time. The **DateTime** is stored in the database as Coordinated Universal Time (UTC) and is always displayed as local time in Microsoft Dynamics NAV. Local time is determined by the time zone regional settings that are used by the client computer. DateTimes must always be entered as local time. When a **DateTime** is entered as local time, it is converted to UTC by using the current settings for the time zone and daylight saving time. The **DateTime** data type does not support closing dates.
Decimal	This data type denotes a decimal number from -999,999,999,999,999.99 to +999,999,999,999,999.99. In Microsoft Dynamics NAV, the **Decimal** data type is mapped to the Microsoft .NET common language runtime (CLR) Decimal data type. The precision and limits behave slightly different from the Binary Coded Decimal (BCD) data type in earlier versions of C/AL.
Duration	This data type represents the difference between two points in time, in milliseconds. This value can be negative.
GUID	A Globally unique identifier (GUID) is used for the global identification of objects, programs, records, and so on.

Data Type	Description
Integer	This data type indicates a whole number between -2,147,483,647 and 2,147,483,647. The corresponding SQL data type is **INTEGER**.
Option	This data type indicates an option value that is an integer in the range -2,147,483,647 and 2,147,483,647. An option field is defined by using an option string. This is a comma-separated list of strings that represent each valid value of the field. This string is used when an **Option** field is formatted and its value is converted into a string. For example: • The **Option** field **Color** is defined by using the option string 'Red,Green,Blue'. • Valid values of the field are then 0, 1 and 2, with 0 representing Red and so on. • When the **Color** field is formatted, 0 is converted into the string Red, 1 into Green, and 2 into Blue.
RecordID	This data type contains the table number and the primary key of a table.
TableFilter	This data type is used to apply a filter to another table. This can only be used to apply security filters from the **Permission** table.
Text	This data type denotes an alphanumeric string. The string must be defined to be between 1 and 250 characters. An empty text string has the length of zero. The corresponding SQL data type is **NVARCHAR**.
Time	This data type indicates any time in the range 00:00:00 to 23:59:59.999. An undefined time is expressed as 0T. The corresponding SQL data type is **DATETIME**.

Field Properties

Some field properties are used to identify the field among other fields, such as **ID**, **Name**, and **Caption**. Other field properties are used to establish data type and set the fields behavior, such as the **DataType**, **Enabled**, and **NotBlank** properties. The following steps show how to open the **Properties** window for one of the fields (the **No.** field) in the **Customer** table.

1. Design table **18**, **Customer** from the Object Designer.
2. Select the **No.** field and then click **View > Properties**, or click the **Properties** button on the Toolbar.

3. The **No. - Properties** window opens and shows the properties for the field. Developers can now view and modify properties for the **No.** field.

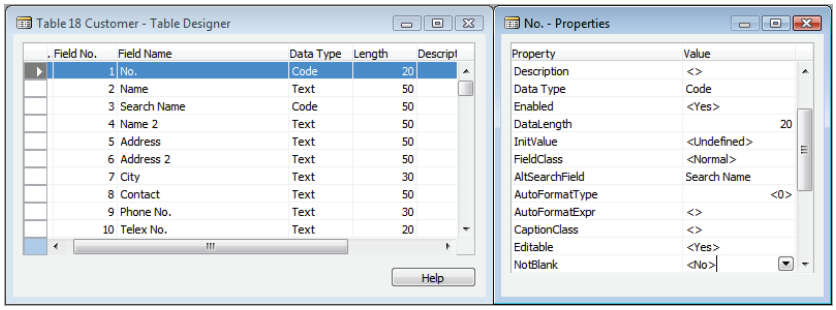

FIGURE 2.4: NO. FIELD PROPERTIES

4. Locate the **NotBlank** property.
   The value of this property is <**No**>. This property prevents users from leaving this field blank. It is used on most primary key fields.

Note: *When you select the* ***Phone No.*** *field without closing the* ***No. - Property*** *window, notice that now the* ***Phone No. – Property*** *window is shown.*

By selecting a value in the ***ExtendedDataType*** *property, you will change the layout and behavior of controls on a page. Use the value to add an icon next to an input field to indicate whether the field relates to a phone number, email address, or URL.*

Field Triggers

The following table shows each field and its triggers.

Field's Trigger Name	When it is performed.
OnValidate	Data is entered in a field or when **<Record>.VALIDATE** is performed in C/AL code.
OnLookup	Lookup is activated.

Table and field triggers can be viewed and modified from the C/AL Editor, which you can access from the Table Designer. The following steps show how to view and modify the **Customer** table's fields and table triggers.

1. Design table **18**, **Customer** from the Object Designer.
2. Click **View > C/AL Code**, or click the **C/AL Code** button on the Toolbar. The C/AL Editor opens and shows all the triggers that are available to the **Customer** table and its fields. From the top, the table

triggers are shown, followed by the field triggers and the function triggers at the bottom.

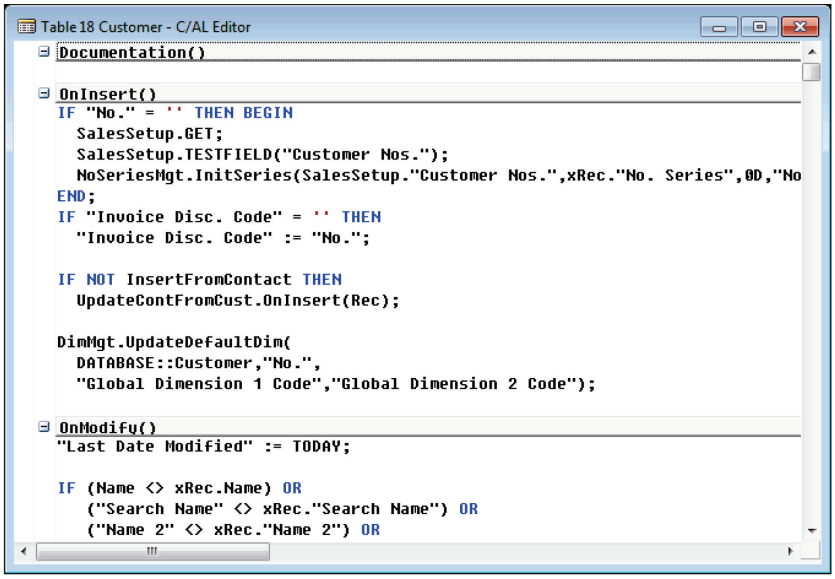

FIGURE 2.5: CUSTOMER TABLE TRIGGERS

Keys

The table description contains a list of keys. A key is a sequence of one or more field IDs from the table. Up to 40 keys can be associated to a table. Keys are used to define the sorting method of records in a table, although this is not the only use of keys.

The following steps show how to view the **Customer** table keys.

1. Design table **18**, **Customer** from the Object Designer.
2. Select **View > Keys**. The **Keys** window opens. It shows the keys that are defined for the **Customer** table.

The keys that are associated with a table also have properties that describe their behavior. The following steps show how to open the **Properties** window for the **Customer** table key.

1. In the **Keys** window, select any of the keys.
2. Click **View > Properties**, or click the **Properties** button on the Toolbar.

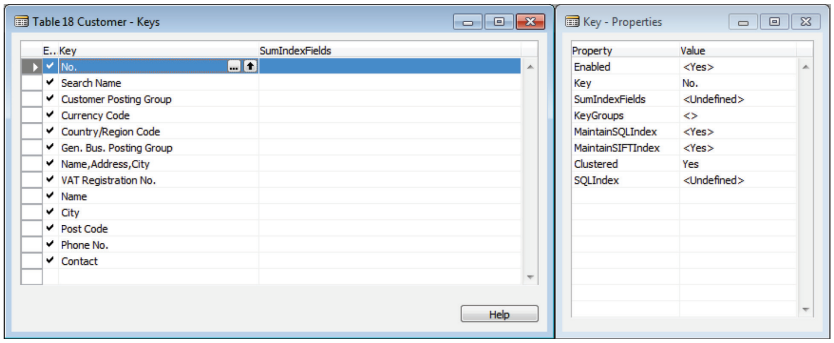

FIGURE 2.6: CUSTOMER TABLE KEY AND ITS PROPERTIES

Field Groups

In the RoleTailored client, some fields are enabled with filter-as-you-type functionality. When you type text into these fields, a drop-down control appears that displays data from a source-table. To specify which fields are displayed in the drop-down control, you must use Field Groups.

The following steps show how to view the **Customer** table Field Groups.

1. Design table **18, Customer** from the Object Designer.
2. Select **View > Field Groups**. The **Field Groups** window opens. It shows the Field Groups that are defined for the **Customer** table.

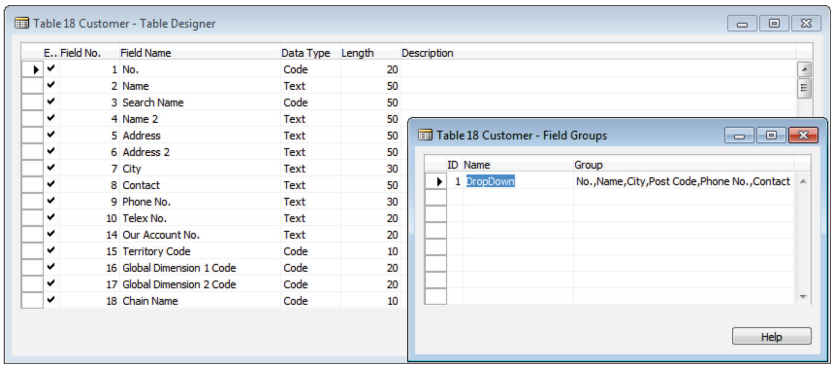

FIGURE 2.7: CUSTOMER TABLE FIELD GROUPS

An example of the Field Groups is used in the **Sales Order** page. The **Sell-to Customer No.** field has a table relation to the **Customer** table. On the page, when you select the **Sell-to Customer No.** field, a drop-down control shows the fields that are defined in the **Customer** table's Field Groups.

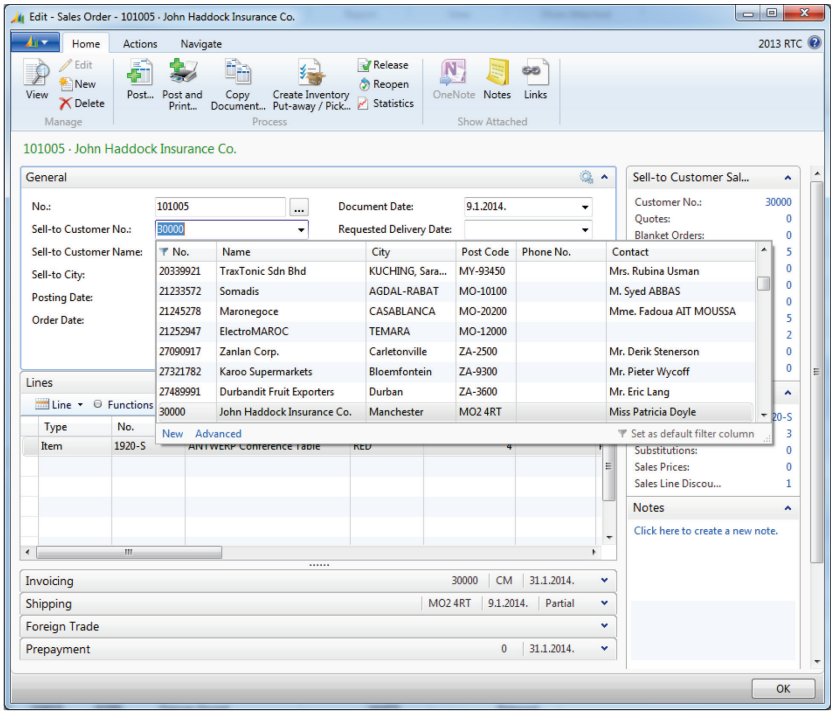

FIGURE 2.8: FIELD GROUPS IN A SALES ORDER'S SELL-TO CUSTOMER NO. FIELD

Note: *If you do not define any Field Groups for the drop-down control, by default, you will only see data from two fields in the source table: the primary key and description fields, because these are indexed automatically.*

Primary and Secondary Keys

The two kinds of keys are primary and secondary.

Up to 40 keys can be associated to a table, and the first on the list is the primary key. All other keys are secondary keys and optional.

Primary Keys

The database keeps track of each record by using the record's primary key. Consider the primary key of a record as the name of the record. The identifier of each record in a particular table for a particular company must be unique. This identifier is how Microsoft SQL Server retrieves and updates records.

The primary key is composed of up to 20 fields in a record. The combination of values in fields in the primary key makes it possible for Microsoft SQL Server to perform a unique identification of each record. The primary key determines the logical order in which records are stored, regardless of their physical placement on a disk.

Logically, the records are stored sequentially in ascending order, sorted according to the primary key. Before you add a new record to a table, Microsoft SQL Server checks that the information in primary key fields in the record is unique and only then inserts the record into its correct logical position. Because the records are sorted as they are entered, the database is always structurally correct. This enables fast data manipulation and retrieval.

The primary key is always active. Microsoft SQL Server keeps the table sorted in primary key order and rejects records with duplicate values in primary key fields. Therefore, the values in the primary key must always be unique. It is not the value in each field in the primary key that must be unique, but the combination of values in all the fields that consist of the primary key. Microsoft Dynamics NAV does not support unkeyed tables.

Secondary Keys

Secondary keys are used to view records in an order that differs from the one in which they are sorted, according to the primary key fields.

The number of fields in the primary key, together with all the fields in each secondary key, must not exceed 20. Each secondary key can contain up to 20 unique fields. However, these 20 unique fields must also include all the fields from the primary key. The primary key fields that are not mentioned specifically in the secondary key are added to the end of the key by Microsoft SQL Server.

This means that if the primary key includes four distinct fields, then a secondary key can include these four fields and, at most, 16 other fields. Correspondingly, if the primary key consists of 20 distinct fields, then any secondary key must consist only of combinations of these fields.

When a secondary key is defined and selected as active, the system automatically maintains an index that reflects the sorting order that the key defined. Several secondary keys can be active at the same time.

A secondary key can be changed into an inactive key. This means that Microsoft SQL Server does not use time during updates to maintain its index. Moreover, an inactive key does not occupy database space. Inactive keys can be reactivated. This process can consume some time, depending on the size of the table because Microsoft SQL Server has to scan the table to rebuild the index.

The fields that consist of the secondary keys are not guaranteed to contain unique data. Microsoft SQL Server does not reject records with duplicate data in secondary key fields. If two or more records contain identical information in the secondary key, Microsoft SQL Server uses the primary key for the table to resolve this conflict.

Demonstration: Create a simple table

The following demonstration shows how to create a simple table, set the primary key, create secondary keys, and add data to the table. The purpose of the table is to record information about vehicles. This includes the model, serial number, transmission type, and date of manufacturing.

Demonstration Steps

1. Create a new table.
   a. In the **Object Designer's Table** list, click **New**. The Table Designer opens.
   b. Type the following in the Table Designer:

Field No.	Field Name	Data Type	Length
10	Model	Code	20
20	Serial No.	Integer	
30	Description	Text	50
40	Transmission	Option	
50	List Price	Decimal	
60	Date of Manufacturing	Date	

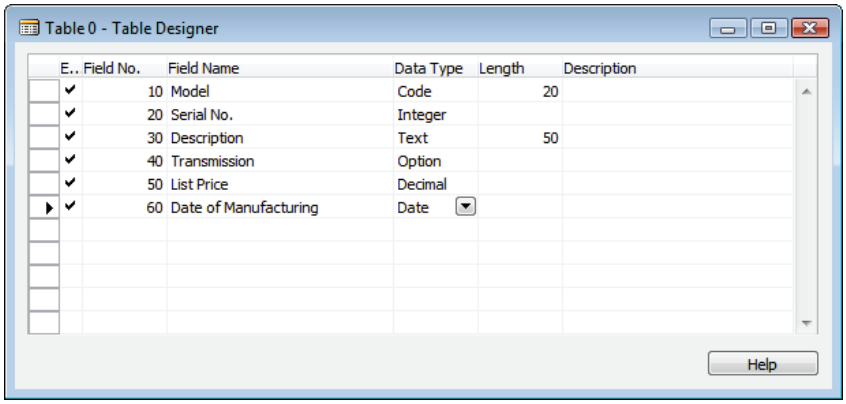

FIGURE 2.9: NEW TABLE

   c. Open the **Properties** window for the **Transmission** field, and set the following property:
      - **OptionString**: Automatic,4-Speed,5-Speed

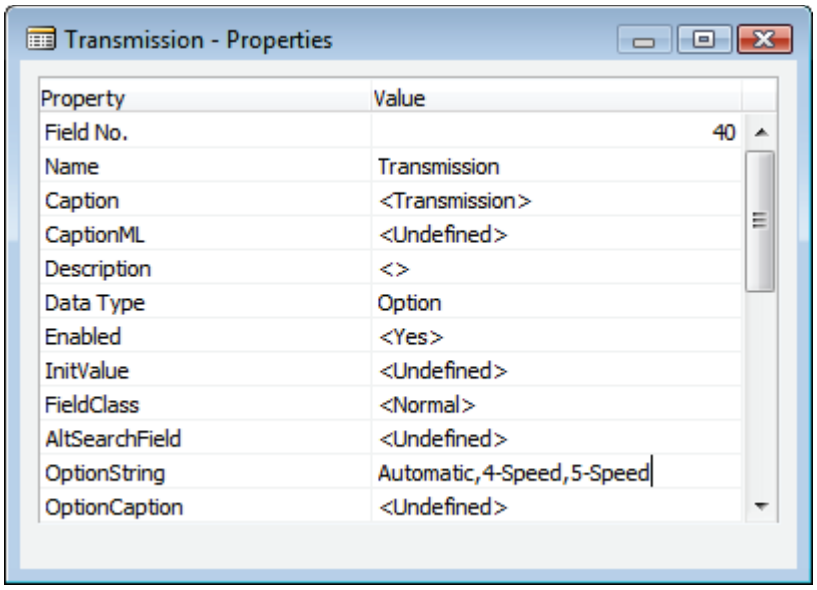

FIGURE 2.10: TRANSMISSION FIELD PROPERTIES WINDOW

Note: *The options set in the* ***Transmission*** *field are displayed in a drop-down list in the table. If a space is added between the options, or any symbols such as the greater-than and less-than symbols are added in the option string, those symbols are displayed in the drop-down list.*

d. Close the **Properties** window.

e. Compile and save the table by selecting **File > Save As**. The **Save As** dialog box appears.

f. Type "90000" in the **ID** field and "Vehicle" in the **Name** field**,** ensuring that the **Compiled** check box is selected, and then click **OK**. This compiles and saves the table.

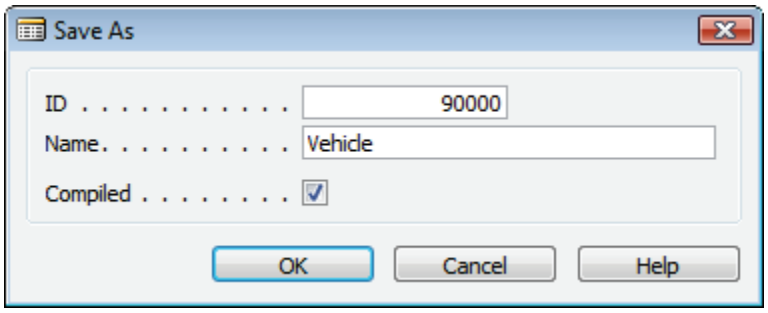

FIGURE 2.11: SAVE AS WINDOW

g. Close the table by closing the Table Designer.

2. Set the primary key.

Note: *Microsoft Dynamics NAV 2013 does not allow for unkeyed tables. When you create a new table and save it without a key, the first field is defaulted as the first key. Therefore, it becomes the primary key. With this configuration, there cannot be any two records with the same Model. This is not a good choice for the primary key. The following steps show how to set a correct primary key.*

a. Design table **90000**, **Vehicle** from the Object Designer.
b. Click **View > Keys** to open the **Keys** window.
   Notice that Model is defaulted as the primary key.

Note: *With the current configuration, there can be only one unique vehicle model in the **Vehicle** table.*

c. Replace the current primary key with the following:
   - **Key**: Model,Serial No.

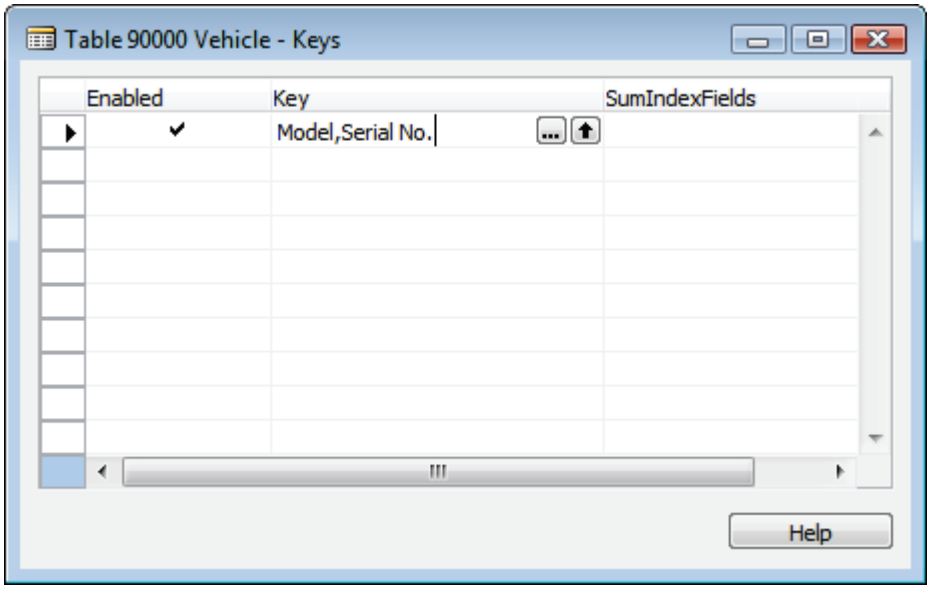

FIGURE 2.12: VEHICLE TABLE WITH THE PRIMARY KEY

Note: *Instead of typing the value directly into the **Key** field, click the **AssistEdit** button on the **Key** field to open the **Field List** window. Add the fields that will be created as a key in the **Field List** window and then click **OK**.*

d. Close the **Keys** window.
e. Compile, save, and close the table.

Note: *This enables multiple models of the vehicle in the* ***Vehicle*** *table, providing that they have different serial numbers.*

3. Add data to the table.

Note: *Generally, you add data to a table through pages. The following steps show how to manually add data directly to the table.*

a. In the **Object Designer's Table** list, select table **90000**, Vehicle and then click **Run**.
The **Vehicle** table opens in a tabular form. This form is auto-created by the Microsoft Dynamics NAV Development Environment. Notice that the table is currently empty.

b. Type the following information in the **Vehicle** table:

Model	Serial No.	Description	Transmission	List Price	Date of Manufacturing
FORD	5000	Red, Mustang	Automatic	18,000	01/15/10
FORD	2000	Blue, F150	5-Speed	26,000	01/15/10
TOYOTA	1000	Gold, Camry	Automatic	23,000	02/01/11
FORD	3000	Black, Explorer	Automatic	30,000	01/15/10
TOYOTA	3000	Black, Tacoma	5-Speed	20,000	12/15/09
TOYOTA	2000	Gray, Camry	Automatic	22,000	01/15/10

Note: *Every new record is sorted in the order of the primary key fields: alphabetically by* ***Model*** *and then numerically by* ***Serial No.***

c. Close the table.

4. Create a secondary key.

Note: *For some users, the date of manufacturing and the transmission are more important than the model and serial number. Those users might want to change the order of the records on the page to be sorted by* ***Date of Manufacturing*** *and* ***Transmission****. To do this, a secondary key must be created. The following steps show how to create a secondary key.*

a. Design table **90000**, **Vehicle** from the Object Designer.

b. Select **View > Keys**. The **Vehicle - Keys** window opens.

c. Click the **AssistEdit** button on the **Key** field of the first blank line (second line). The **Field List** window opens.

d. Type the following information in the **Field List** window:

Field
Date of Manufacturing
Transmission

Note: *Instead of typing the field name in the* ***Field List*** *window, use the* ***Lookup*** *button to select the fields from the* ***Vehicle*** *table. The order of these fields is important. In the order described, the records are sorted by* ***Date of Manufacturing*** *first and then by* ***Transmission****, if the dates are the same.*

e. Click **OK** to close the **Field List** window.
The secondary key of **Date of Manufacturing,Transmission** is created in the **Vehicle - Keys** window.

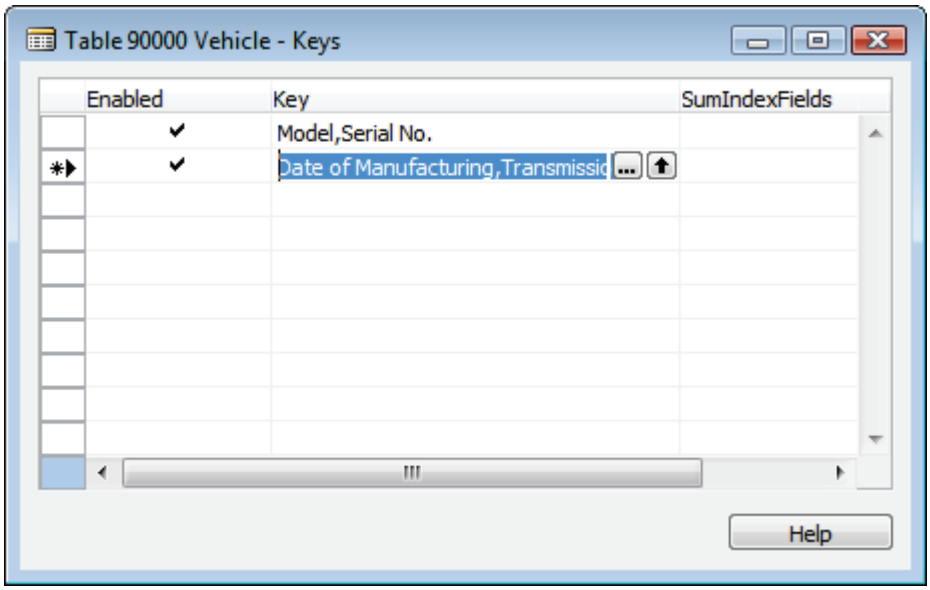

FIGURE 2.13: VEHICLE TABLE WITH A SECONDARY KEY

f. Close the **Vehicle - Keys** window.

g. Compile, save, and close the table.

5. Use the secondary key.
   a. Run table **90000**, **Vehicle** from the Object Designer.
      The **Vehicle** table opens and it contains the data previously entered. The records are sorted in the primary key order.
   b. Select the **Sort** button on the Toolbar. The **Sort** window opens.
   c. Select the **Date of Manufacturing,Transmission** key and then click **OK**.
      The records immediately re-sort themselves by their **Date of Manufacturing** and **Transmission**.

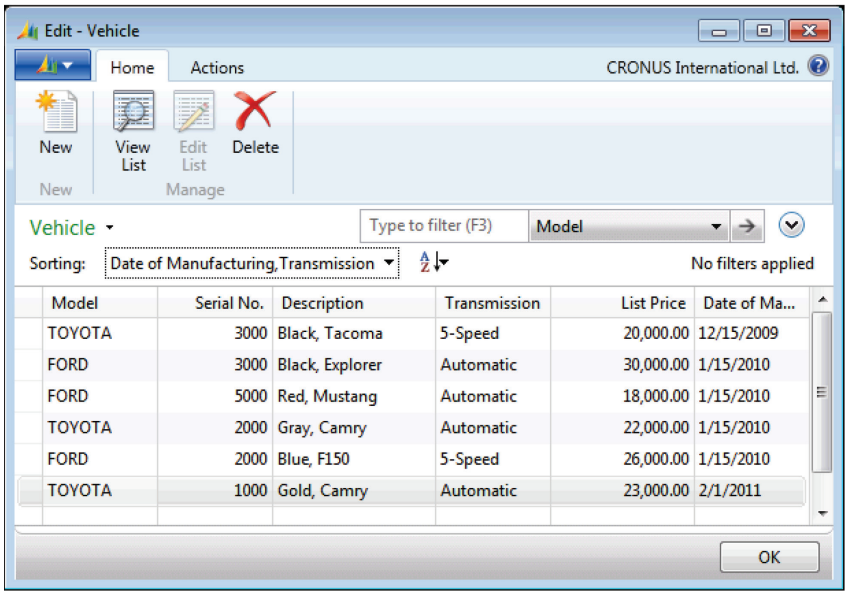

FIGURE 2.14: VEHICLE TABLE WINDOW SORTED BY THE SECONDARY KEY

Table Relationships

The three kinds of relationships between tables in relational database design are as follows:

- One-to-Many
- Many-to-Many
- One-to-One

The one-to-many relationship is the most common. If a database contains tables with related data, developers can define a relationship between them. Developers relate tables by specifying one or more fields that contain the same value in related records. These matching fields frequently have the same name in each table. The fields can use relationships to do the following:

- Validate data entries
- Perform lookup in other tables
- Automatically propagate changes from one table to other tables

Table Relation Property

Table relations are defined by using the **TableRelation** property. This property lets developers define simple and advanced table relations. A simple table relation consists of only a table ID and an optional field ID, whereas advanced table relations are typically prefixed with a conditional statement and include filters.

When developers create a table relation, they specify which field to look up in another table. If the optional field ID is left blank, the first field in the primary key is set to **Relate to**. To make a relation to the second or third field in a primary key, the field ID must be specified.

Note: *The field ID that is specified in the* ***TableRelation*** *property must be in the primary key of the table that is specified by the table ID in the property. If the specified field is not the first field in the primary key, the other fields that are listed before it in the key must be filtered to one value. This is not a true system requirement, but makes sense in a real application to avoid user confusion.*

An example of a table relation is shown in the **Sales Header** table. The **Sales Header** table has a **Salesperson Code** field that relates to the **Salesperson/Purchaser** table. When the users look up the **Salesperson Code** field, the **Lookup** page of the **Salesperson/Purchaser** table opens.

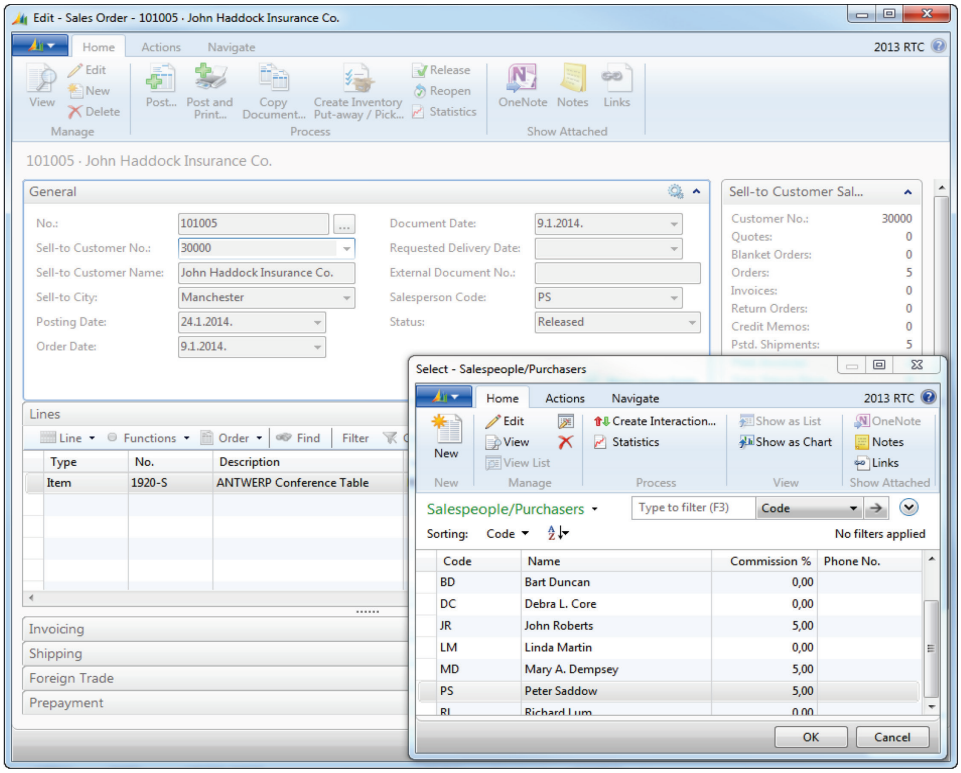

FIGURE 2.15: TABLE RELATION IN SALES ORDER'S SALESPERSON CODE

Filter Table Relation

Advanced table relations can involve filtering within the table relation. An example of filtering in a table relation is shown in the **Sales Line** table. The **Sales Line** table has a **Location Code** field that relates to the **Location** table.

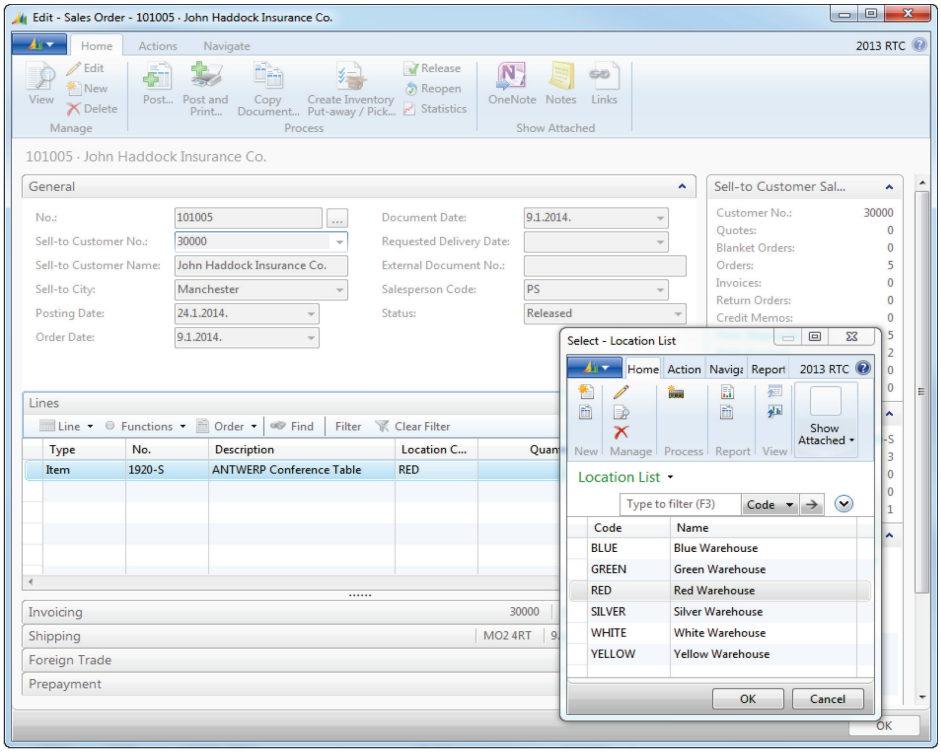

FIGURE 2.16: FILTERED TABLE RELATION IN SALES ORDER LINE'S LOCATION CODE

The **Location** table holds all the records for location. This includes those used as In-Transit locations. When users select a location in the **Sales Line**, for example in the **Sales Order**, they should be unable to select a location that is used as In-Transit locations. Therefore, the table relation must be filtered to show only those locations that are not used as In-Transit locations.

Conditional Table Relation

Advanced table relations can also involve a conditional statement within the table relation. An example of a conditional table relation is also shown in the **Sales Line** table. The table **Sales Line** has a **No.** field that relates to six different tables, depending on the condition of the **Type** field in the **Sales Line**.

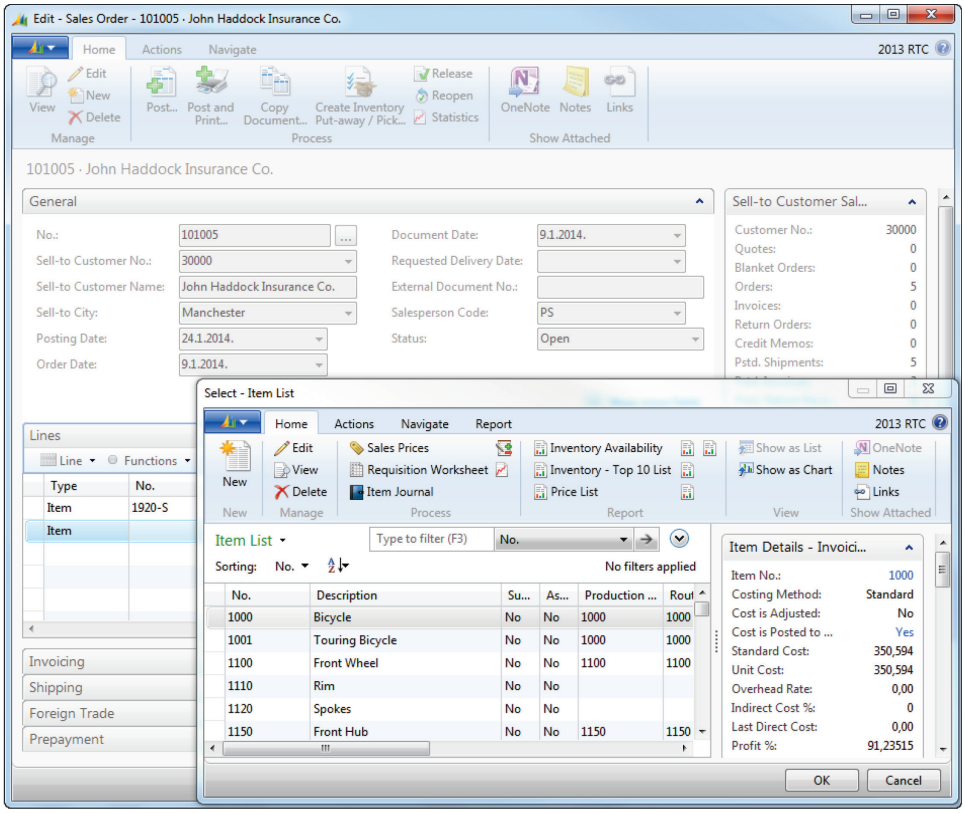

FIGURE 2.17: CONDITIONAL TABLE RELATION IN SALES ORDER LINE'S NO.

If the users select **Item** as the **Type** in the **Sales Line**, for example in the **Sales Order**, then the **No.** field relates to the **Item** table. When the users look up the **No.** field, the **Lookup** page of the **Item** table opens. Or, if the users select **Resource** as the **Type**, the **No.** field relates to the **Resource** table, and when the users look up the **No.** field, the **Lookup** page of the **Resource** table opens.

Demonstration: Set Table Relations

The following demonstration shows how to create a table to record sales transactions and create fields that have a table relation to other tables.

You need to create a new table, named **Sales Transaction**. This table must record the salesperson of a particular transaction. The **Salesperson/Purchaser** table holds the record of all salespersons and purchasers in CRONUS International, Ltd. Therefore, your **Sales Transaction** table sets a table relation for the **Salesperson Code** field in the **Sales Transaction** table to relate to the **Salesperson/Purchaser** table.

The **Sales Transaction** table must also to record sales transactions of Items, Resources, or G/L Accounts. The **Type** field is used to differentiate this. Depending on the **Type** selected, the **No.** field must relate to different tables, the **Item** table, the **Resource** table, the **G/L Account** table, or the **Standard Text** table. The following steps show how to set a conditional table relation on the **No.** field.

Demonstration Steps

1. Create a table with table relations.
   a. In the **Object Designer's Table** list, click **New**. The Table Designer opens.
   b. Type the following in the Table Designer:

Field No.	Field Name	Data Type	Length
10	Line No.	Integer	
20	Salesperson Code	Code	10
30	Type	Option	
40	No.	Code	20
50	Amount	Decimal	

The table will look as shown in the "Table Designer" figure.

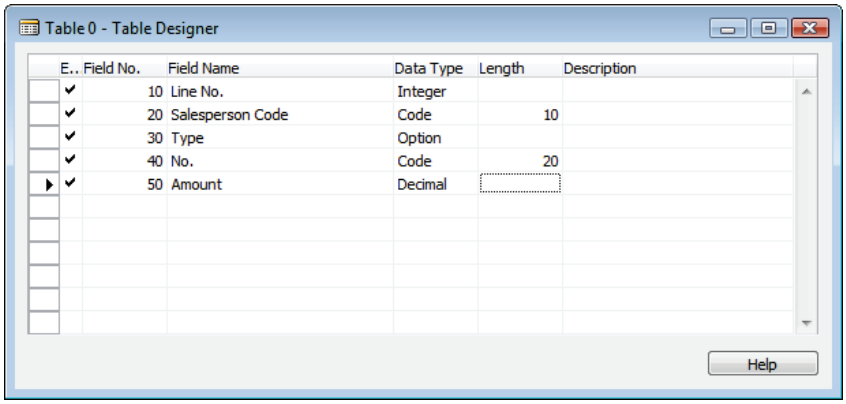

FIGURE 2.18: TABLE DESIGNER

   c. Open the **Properties** window for the **Type** field and set the following property:
      - **OptionString**: G/L Account,Item,Resource

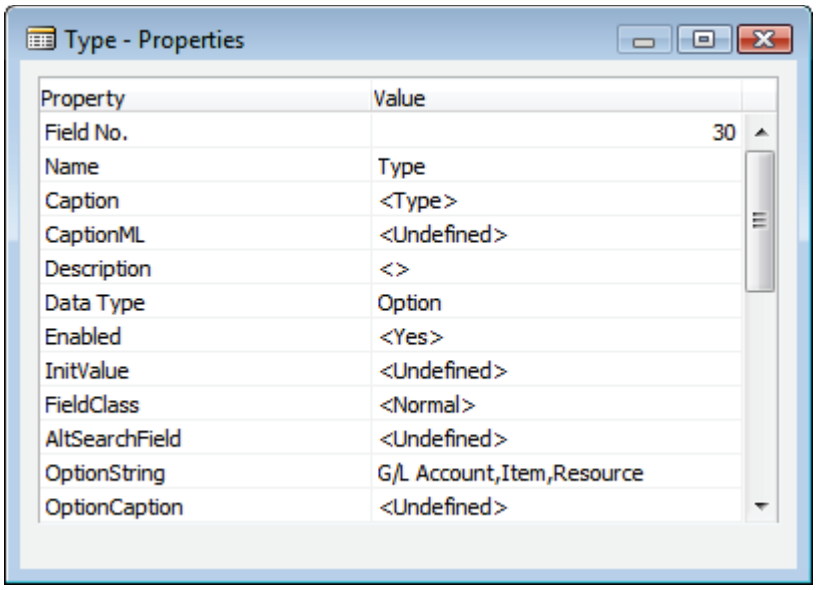

FIGURE 2.19: TYPE – PROPERTIES WINDOW

d. Close the **Properties** window.

e. Compile and save the table with the ID of 90001 and the name of **Sales Transaction**.

2. Set a table relation with a filter.

   a. Open the **Properties** window for the **Salesperson Code** field.

   b. Click the **AssistEdit** button on the **TableRelation** property. The **Table Relation** window opens.

   c. Click **Lookup** on the Table column. The **Table List** window opens.

   d. Select **Salesperson/Purchaser** from the **Table List** window, and then click **OK**.

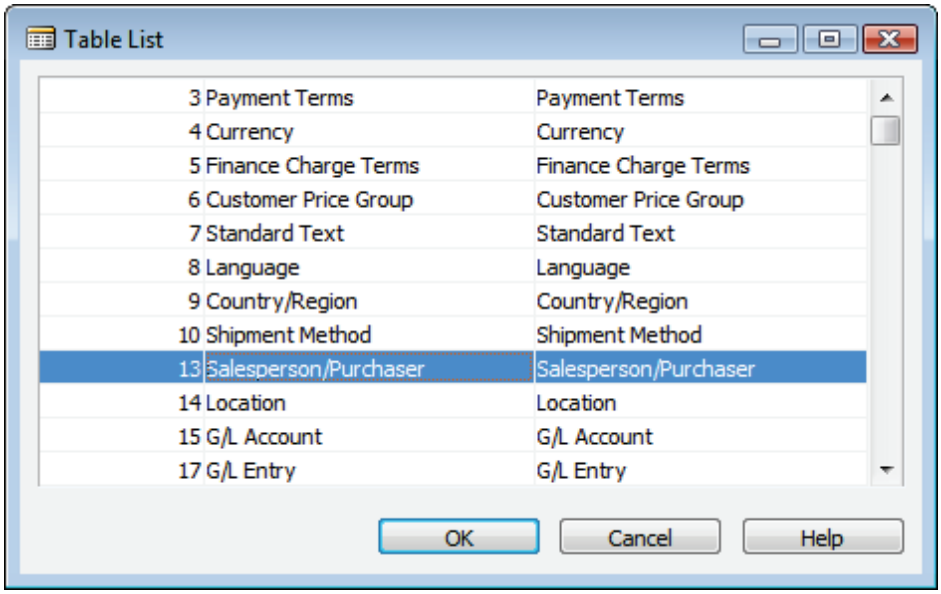

FIGURE 2.20: TABLE LIST WINDOW

Note: *Instead of selecting the table, typing the table ID or the table name directly in the Table column also achieves the same result.*

e. Click the **AssistEdit** button on the **Table Filter** column. The **Table Filter** window opens.
In this window, specify filters on as many fields in the related table as needed. A filter can be added to the list in three ways:
   - By a constant value
   - By using a filter expression
   - By a field in the current table

In this case, use a filter expression.

f. Type the following in the **Table Filter** window:
   - **Field:** Commission %
   - **Type:** Filter
   - **Value**: >0

The **Table Filter** window will look as shown in the following figure.

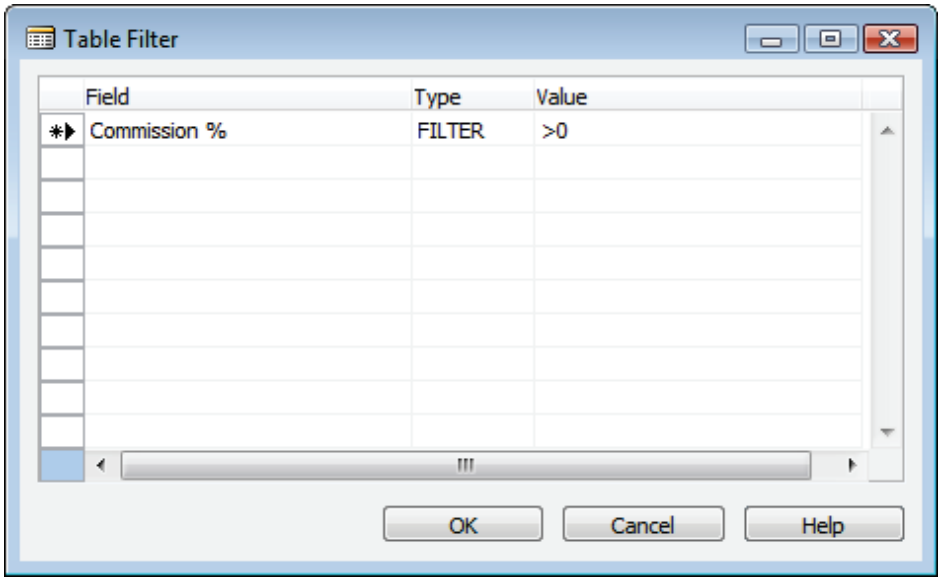

FIGURE 2.21: TABLE FILTER WINDOW

Note: *Selecting **Filter** in the Type column enables users to filter instances where **Commission %** is greater than zero. For this scenario, only the **Filter** option can be used. **Const** and **Field** options locate exact matches only by using the **equal to** operator.*

g. Click **OK** to close the **Table Filter** window. Notice that the Table Filter column in the **Table Relation** window is set to **Commission %=FILTER(>0)**.

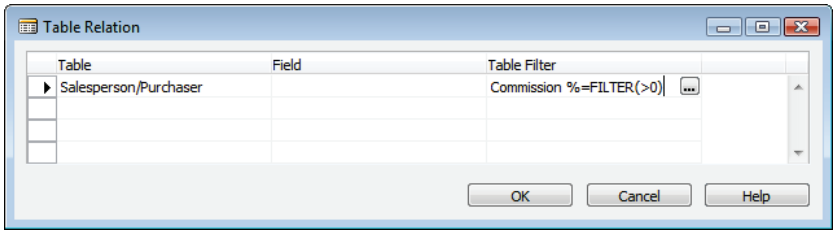

FIGURE 2.22: TABLE RELATION WINDOW

h. Click **OK** to close the **Table Relation** window. Notice that the **TableRelation** property is set to **Salesperson/Purchaser WHERE (Commission %=FILTER(>0))**.

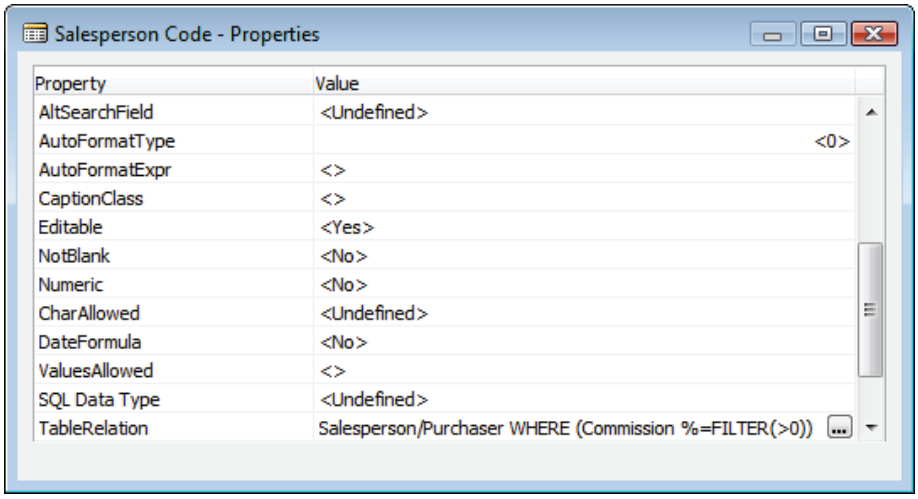

FIGURE 2.23: SALESPERSON CODE – PROPERTIES WINDOW

i. Close the **Salesperson Code - Properties** window.

j. Compile and save the table.

3. Set a conditional table relation.

   a. Open the **Properties** window for the **No.** field.

   b. Click the **AssistEdit** button on the **TableRelation** property. The **Table Relation** window opens.

   c. Type the following in the **Table Relation** window:

Condition	Table
Type=CONST(G/L Account)	G/L Account
Type=CONST(Item)	Item
Type=CONST(Resource)	Resource
	Standard Text

Note: *The empty condition means* else*, and is applied if none of other conditions are satisfied.*

The **Table Relation** window will look as shown in the following figure.

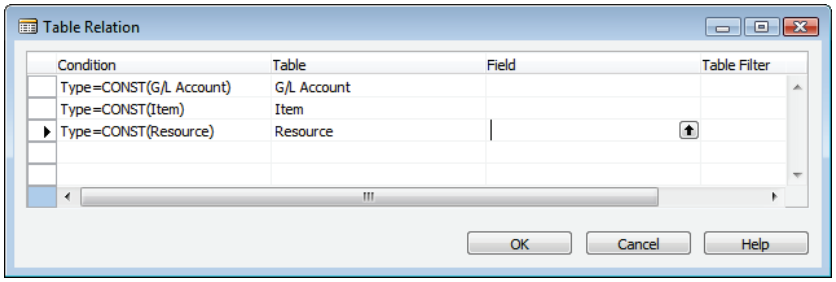

FIGURE 2.24: TABLE RELATION WINDOW

Note: *In the* ***Table Relation*** *window, instead of typing the value, use the* ***AssistEdit*** *button on the Condition column to fill up each condition and use the* ***Lookup*** *button on the Table column to open the table list and select each table. Scroll quickly through the table list by pressing the first letter of the table name until the intended table is located.*

d. Click **OK** to close the **Table Relation** window. Notice that the **TableRelation** property is set to **IF (Type=CONST(G/L Account)) "G/L Account" ELSE IF (Type=CONST(Item)) Item ELSE IF (Type=CONST(Resource)) Resource ELSE "Standard Text"**.

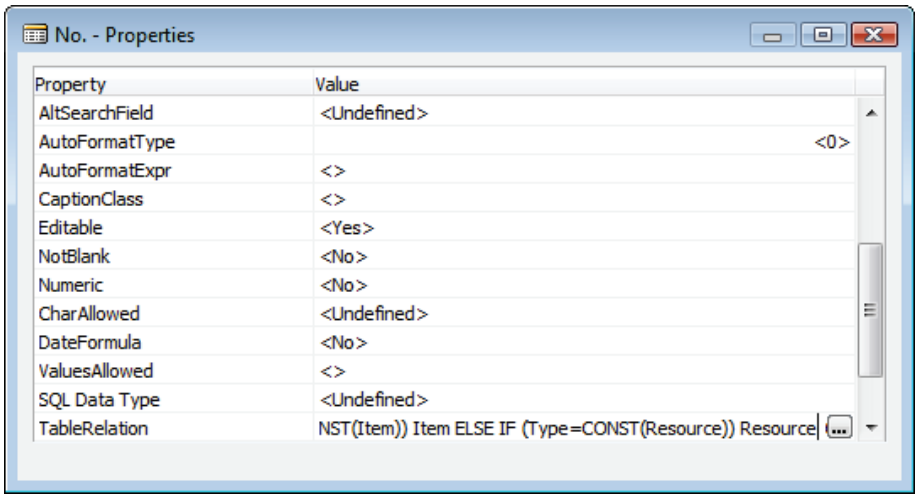

FIGURE 2.25: NO. – PROPERTIES WINDOW

e. Close the **No. - Properties** window.

f. Compile, save, and close the table.

4. Test a simple table relation.

   a. Run table **13**, **Salesperson/Purchaser** from the Object Designer.

   b. Move to a blank line and type the following:

- **Code:** XX
- **Name:** New Salesperson
- **Commission %:** 5

c. Close table **13**.

d. Run table **90001**, **Sales Transaction**.

e. Type the following in the **Sales Transaction** table:

Line No.	Salesperson Code	Type	No.	Amount
10000	XX	Item	70000	123.56
20000	JR	Item	70000	234.97
30000	XX	Item	70010	90.99

f. Close table **90001**.

g. Run table **13**, **Salesperson/Purchaser**.

h. Select the previously added line and then replace the XX in the **Code** field with YY. Click **Yes** when a message prompts asking for confirmation.

Note: *Changing the primary key values of a record in a table changes the name of the record. This is known as renaming a record. Notice the results of renaming a record that has other records related to it by a table relation.*

i. Close table **13**.

j. Run table **90001**, **Sales Transaction**. Notice that all records with **Salesperson Code** XX automatically changed to YY.

k. Close table **90001**.

5. Test a filter and a conditional table relation.

a. Run table **90001**, **Sales Transaction** from the Object Designer.

b. Delete all the current records in the table.
The easiest way to delete all records is to select the upper-left gray box of the form, which selects all the records, and then press CTRL+DEL.

c. Type "10000" in the **Line No.** field of the first blank line.

d. Select the **Lookup** button on the **Salesperson Code** field to retrieve a value from the **Salesperson/Purchaser** table. Notice that not all records from the **Salesperson/Purchaser** table are shown. Only those records that have the **Commission %** field set to greater than zero are shown.

e. Select **JR, John Roberts**.

Note: *The value JR is added to the* ***Salesperson Code*** *field. Because the table relation is only set up to the table and not to any particular field, the lookup returns the value from the first primary key field of the table. The* ***Salesperson/Purchaser*** *table only has one field in the primary key. Therefore, you do not have to specify a field in the table relation. Only primary key fields can be selected in a table relation.*

f. Select **Item** in the **Type** field.
g. Click **Lookup** on the **No.** field. The lookup page of the **Item** table (the **Item List** page) opens.
h. Close the **Item List** window.
i. Select **Resource** in the **Type** field.
j. Click **Lookup** on the **No.** field. Notice that this time, the lookup page of the **Resource** table (the **Resource List** page) opens.
k. Close the **Resource List** window and then close the table.

Special Table Fields

In addition to the conventional data fields that hold values, three kinds of specialized fields are available for data retrieval:

- SumIndexFields
- FlowFields
- FlowFilter Fields

FlowFields

A FlowField is a virtual field that extends the table data. It is not a permanent part of the table data. A FlowField is a powerful feature of Microsoft Dynamics NAV 2013. It is used to calculate values from another table. The information in the FlowFields exists only at run time.

To update a FlowField, developers use the **CALCFIELDS** function that has the syntax:

Code Example

```
[Ok :=] <Record>.CALCFIELDS(Field1, [Field2],...)
```

If a FlowField is a direct source expression of a control on a page, the FlowField is automatically calculated when the page is displayed. If a FlowField is a direct source expression of a field on a report, XMLport or a query, the FlowField is automatically calculated when the record is retrieved from the database.

The following table describes the seven kinds of FlowFields.

FlowField	Data Type	Description
Sum	Decimal	The sum of a specified set in a column in a table.
Average	Decimal	The average value of a specified set in a column in a table.
Exist	Boolean	Indicates whether any records exist in a specified set in a table.
Count	Integer	The number of records in a specified set in a table.
Min	Any	The minimum value in a column in a specified set in a table.
Max	Any	The maximum value in a column in a specified set in a table.
Lookup	Any	Looks up a value in a column in another table.

Calculation Formulas and the CalcFormula Property

A FlowField is always associated with a calculation formula that determines how the value in the FlowField is calculated. The following example shows a possible value for the **CalcFormula** property:

Code Example

```
Sum("Cust. Ledger Entry".Amount
 WHERE (Customer No.=FIELD(No.),
     Global Dimension 1 Code=FIELD(Department Filter),
     Global Dimension 2 Code=FIELD(Project Filter),
     Posting Date=FIELD(Date Filter),
     Currency Code=FIELD(Currency Filter)))
```

FlowFilter

Users might want to limit calculations so that they include only those values in a column that have some specific properties. For example, the user might want to sum up only the amounts of customer entries that are entered in April. This is possible if the application is designed by using FlowFilter fields for the FlowFields.

The FlowFilter must exist in the same table as the FlowField and will automatically appear on all pages with this table as source table in the **Limit totals to** filter.

SumIndexFields

A SumIndexField is a decimal field that can be attached to a key definition. This is the fundamental feature of the Microsoft Dynamics NAV that constructs the basis for FlowFields. SumIndexFields enable fast calculation of numeric columns in tables, even in tables with thousands of records. This is because SumIndexFields are maintained when the database record is updated.

SumIndexFields enable fast calculation, such as sums of columns to be displayed by using FlowFields. For example, in a typical database application, if a user wants the sum of all the values in the **Amount** field, Microsoft SQL Server is forced to access every record and add each value in the **Amount** field. This is a time-consuming operation in a database that has thousands of records. With Microsoft Dynamics NAV, as few as two accesses (if the best key is used) are used to add the amount for these records.

This special index structure, a SumIndexField, is associated with a key. Each key can have at most 20 SumIndexFields. During database design, a decimal field can be associated with a key as a SumIndexField. This tells Microsoft SQL Server to create and maintain a structure that contains the accumulated sum of values in a column.

Special Table Fields Example

You can find the use of special table fields almost everywhere in the application. One example of where you can find special table field use is in the **Customer** table. There are several special table fields that are implemented in the **Customer** table.

One of these special table fields is the **Balance (LCY)** field, which is a FlowField of type **Sum**. It is not a physical field in the table; instead, it is derived by calculating the **Amount** field from the **Detailed Cust. Ledger Entry** table, filtered by the **Customer No.**, **Dimensions Filter** and **Currency Filter** fields.

The **Dimensions Filter** (made of the **Global Dimension 1 Filter** and the **Global Dimension 2 Filter**) and the **Currency Filter** fields are FlowFilter fields in the **Customer** table and are used to filter the calculation of the **Balance (LCY)** FlowField.

However, the SumIndexField **Amount** is located in the keys of the **Detailed Cust. Ledger Entry** table. This finishes the implementation of special table fields.

Demonstration: Use Special Table Fields

The following demonstration shows how to create special table fields in the **Salesperson/Purchaser** table for the **Sales Transaction** table—which was created in the previous demonstration—to ease calculations of sales by salespersons.

Demonstration Steps

1. Add records.
   a. Run table **90001**, **Sales Transaction** from the Object Designer.
   b. Delete all the current records in the table.
   c. Type the following in the **Sales Transaction** table:

Line No.	Salesperson Code	Type	No.	Amount
10000	JR	Item	70000	300.00
20000	MD	G/L Account	1120	150.00
30000	JR	G/L Account	1140	200.00
40000	MD	Item	70000	100.00
50000	PS	Item	70000	110.00
60000	JR	Item	70010	50.00
70000	JR	Resource	LIFT	500.00
80000	MD	Resource	LIFT	550.00
90000	PS	Item	70010	75.00

   d. Close the table.

2. Create a FlowField.
   a. Design table **13**, **Salesperson/Purchaser** from the Object Designer.
   b. Add a new field to the table by typing the following:
      - **Field No.:** 50000
      - **Field Name:** Sales
      - **Data Type:** Decimal

The following figure shows how the table will now appear.

Table 13 Salesperson/Purchaser - Table Designer

E..	Field No.	Field Name	Data Type	Length	Description
✔	5076	To-do Status Filter	Option		
✔	5077	Closed To-do Filter	Boolean		
✔	5078	Priority Filter	Option		
✔	5079	Team Filter	Code	10	
✔	5082	Opportunity Entry Exists	Boolean		
✔	5083	To-do Entry Exists	Boolean		
✔	5084	Close Opportunity Filter	Code	10	
✔	5085	Search E-Mail	Code	80	
✔	5086	E-Mail 2	Text	80	
✔	50000	Sales	Decimal		

Help

FIGURE 2.26: SALESPERSON/PURCHASER TABLE

Note: *The field number must be between 50,000 and 99,999 because the* ***Salesperson/Purchaser*** *table is a base application table (table ID is less than 50,000). Customizations to the base application must be made in this range. The data type must be a decimal because a Sum FlowField calculates a decimal value.*

c. Open the **Properties** window for the **Sales** field and then set the **FieldClass** property to FlowField. The **CalcFormula** property is now available.

Sales - Properties

Property	Value
Field No.	50000
Name	Sales
Caption	<Sales>
CaptionML	<Undefined>
Description	<>
Data Type	Decimal
Enabled	<Yes>
InitValue	<Undefined>
FieldClass	FlowField
CalcFormula	<Undefined>
DecimalPlaces	<Undefined>
BlankNumbers	<DontBlank>

FIGURE 2.27: SALES – PROPERTIES WINDOW

d. Click the **AssistEdit** button on the **CalcFormula** property. The **Calculation Formula** window opens. This is the place to set how the FlowField calculates its value.

e. Type the following in the **Calculation Formula** window:
   - **Method:** Sum
   - **Table:** Sales Transaction
   - **Field:** Amount

f. Click the **AssistEdit** button on the **Table Filter** field. The **Table Filter** window opens.

g. Type the following in the **Table Filter** window:
   - **Field:** Salesperson Code
   - **Type:** Field
   - **Value:** Code

The following figure shows how the **Table Filter** window will now appear.

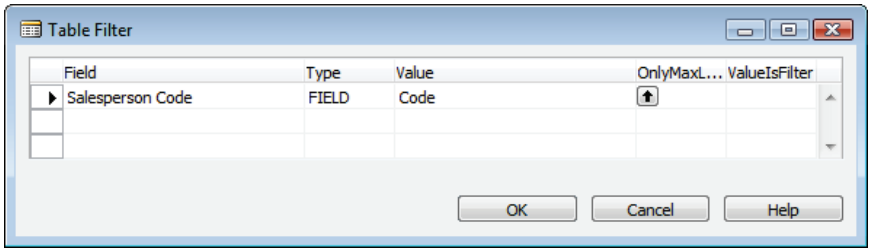

FIGURE 2.28: TABLE FILTER WINDOW

Note: *The previous step makes sure that the* ***Sales*** *FlowField only adds the* ***Amount*** *from the* ***Sales Transaction*** *table that has the same* ***Salesperson Code*** *as the current* ***Code*** *in the* ***Salesperson/Purchaser*** *table.*

h. Click **OK** to close the **Table Filter** window. Notice that the **Table Filter** field in the **Calculation Formula** window is set to **Salesperson Code=FIELD(Code)**.

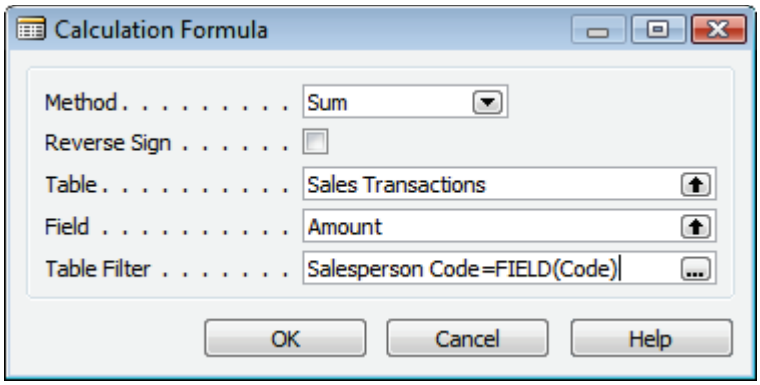

FIGURE 2.29: CALCULATION FORMULA WINDOW

i. Click **OK** to close the **Calculation Formula** window.

Note: *Just closing the window does not save the changes.*

j. Close the **Properties** window.

k. Compile, save, and close the table.

3. Create a SumIndexField.

Note: *The **Sales** FlowField is not yet usable. For the system to calculate the **Sales** FlowField, a SumIndexField must be created in the **Sales Transaction** table with a correct key.*

*The **Sales** FlowField must add the **Amount** fields from **Sales Transaction** table. Therefore, the **Amount** SumIndexField must be created in the **Sales Transaction** table.*

*Because the calculation of the **Sales** FlowField is by **Salesperson Code**, the **Salesperson Code** must be a field in the key that the **Amount** SumIndexField is created in. The order of the fields in the key is not important for this FlowField.*

a. Design table **90001**, **Sales Transaction** from the Object Designer.

b. Click **View > Keys**. The **Keys** window opens.

c. In the **Keys** window, create a secondary key by typing the following:

- **Enabled:** Selected
- **Key:** Salesperson Code
- **SumIndexFields:** Amount

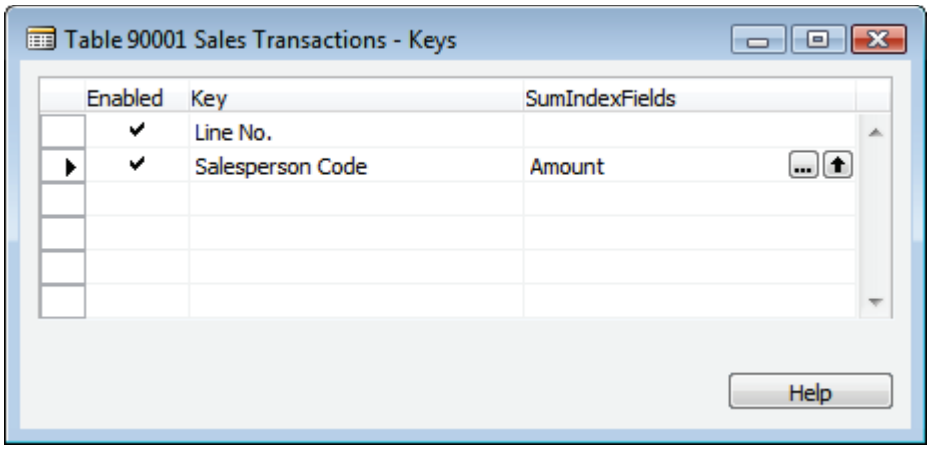

FIGURE 2.30: KEYS WINDOW

d. Close the **Keys** window.

e. Compile, save, and close the table.

4. Test the FlowField.

a. Run table **13**, **Salesperson/Purchaser** from the Object Designer.

b. Go the last column of the table. Notice that the **Sales** field calculates the value from the **Amount** field of the **Sales Transaction** table by **Salesperson Code**.

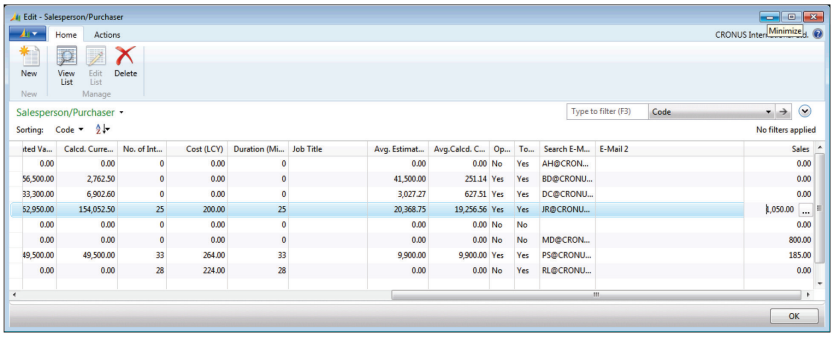

FIGURE 2.31: SALESPERSON/PURCHASER TABLE

Note: *The **Drill-down** button on the **Sales** FlowField is not working. This is because there is no drill-down page defined for the **Sales Transaction** table.*

c. Close the table.

5. Create a FlowFilter.

Note: *A FlowFilter lets users dynamically change the Table Filter part of the FlowField. By creating a FlowFilter, the functionality of the current FlowField can be extended so that users can change the sum based on the* ***Type*** *field in the* ***Sales Transaction*** *table. This lets you use the FlowField for four sums:*

- *Total for G/L Accounts sold*
- *Total for items sold*
- *Total for resources sold*
- *Total of all sales*

Because the calculation will be based not only by ***Salesperson Code****, but also by* ***Type****, the key where the SumIndexField is created must be changed to include* ***Type****.*

a. Design table **90001**, **Sales Transaction** from the Object Designer.
b. Click **View > Keys**. The **Keys** window opens.
c. In the **Keys** window, modify the secondary key to the following:
   - **Enabled:** Selected
   - **Key:** Salesperson Code, Type
   - **SumIndexFields**: Amount

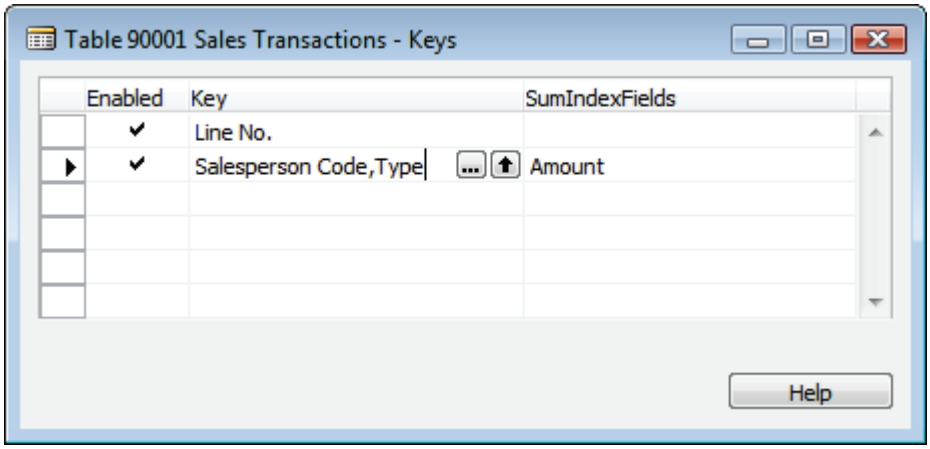

FIGURE 2.32: KEYS WINDOW

d. Close the **Keys** window.
e. Compile, save, and close the table.
f. Design table **13**, **Salesperson/Purchaser** from the Object Designer.

g. Add a new field to the table by typing the following:
   - **Field No.:** 50001
   - **Field Name:** Type Filter
   - **Data Type:** Option

The following figure shows how the table will now appear:

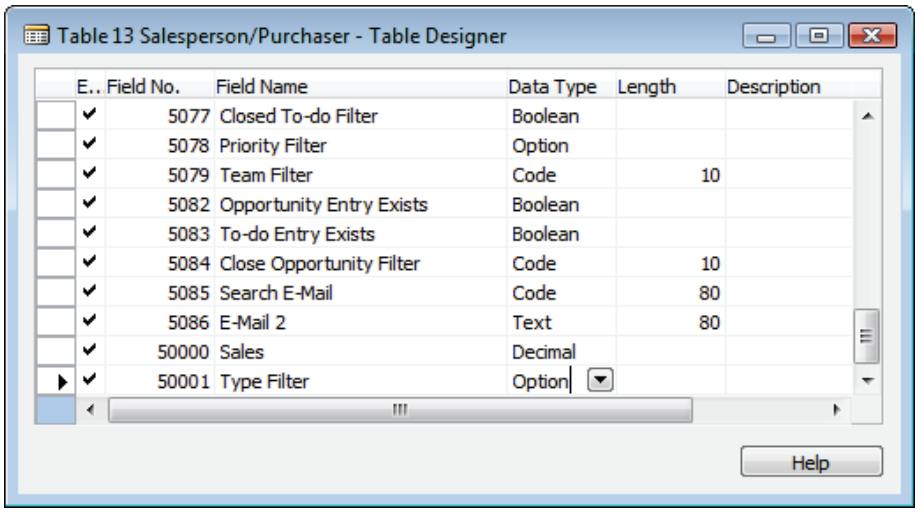

FIGURE 2.33: SALESPERSON/PURCHASER – TABLE DESIGNER WINDOW

h. Open the **Properties** window for the **Type Filter** field and then set the following property:
   - **FieldClass**: FlowFilter
   - **OptionString**: G/L Account,Item,Resource

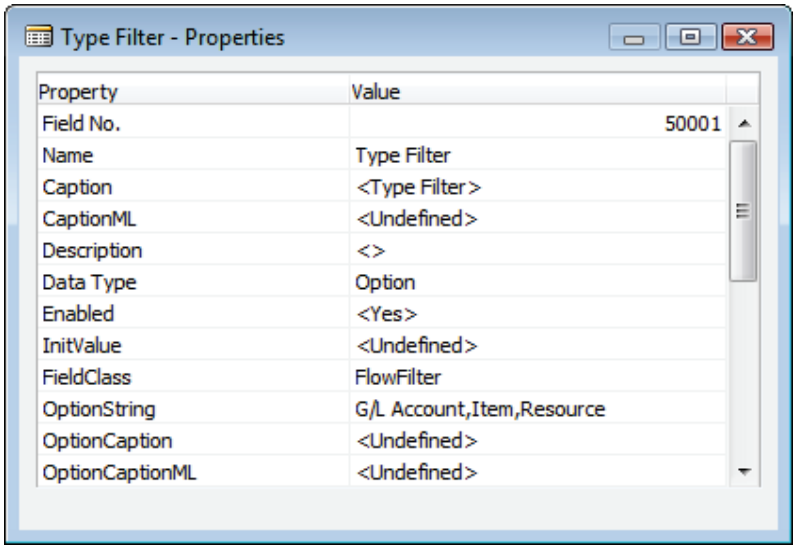

FIGURE 2.34: TYPE FILTER – PROPERTIES WINDOW

i. Close the **Properties** window.

j. Open the **Properties** window for the **Sales** field.

   Click the **AssistEdit** button on the **CalcFormula** property. The **Calculation Formula** window opens.

k. Click the **AssistEdit** button on the **Table Filter** field. The **Table Filter** window opens.

l. Type the following in the **Table Filter** window:

   - **Field**: Type
   - **Type:** Field
   - **Value:** Type Filter

The following figure shows how the **Table Filter** window will now appear:

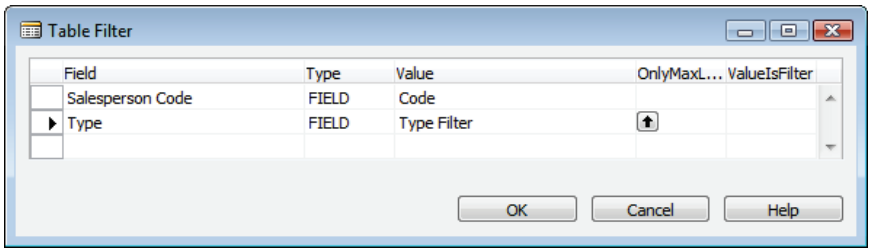

FIGURE 2.35: TABLE FILTER WINDOW

Note: *The previous step makes sure that the* ***Sales*** *FlowField not only adds the* ***Amount*** *from the* ***Sales Transaction*** *table that has the same* ***Salesperson Code*** *as the current* ***Code*** *in the* ***Salesperson/Purchaser*** *table, but that it also has the same* ***Type*** *as the* ***Type Filter*** *field.*

m. Click **OK** to close the **Table Filter** window. Notice that the **Table Filter** field in the **Calculation Formula** window is set to **Salesperson Code=FIELD(Code),Type=FIELD(Type Filter)**.

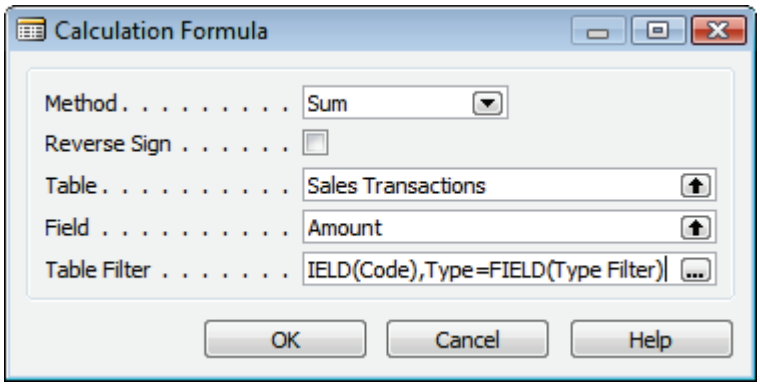

FIGURE 2.36: CALCULATION FORMULA WINDOW

n. Click **OK** to close the **Calculation Formula** window.

o. Close the **Properties** window.

p. Compile, save, and close the table.

6. Test the FlowFilter.

a. Run table **13**, **Salesperson/Purchaser** from the Object Designer.

b. Move to the last column of the table. Notice that the **Sales** field calculates the value from the **Amount** field of the **Sales Transaction** table by **Salesperson Code**.

c. Press CTRL+SHIFT+F3. The **Limit totals to**: filter area shows. This is where you can change the value of a FlowFilter. Remember that changing the value of the FlowFilter might change the value of the FlowField.

d. Click **Add Filter**.

e. In the **Where** field, select **Type Filter**.

f. In the **Select a Value** field select **Item**.

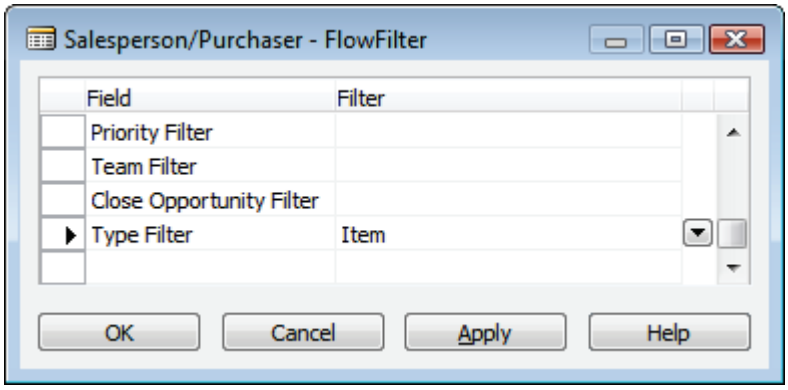

FIGURE 2.37: SALESPERSON/PURCHASER - FLOWFILTER WINDOW

g. Notice that the **Sales** field calculates the value from the **Amount** field of the **Sales Transaction** table by **Salesperson Code** and by **Type** Item.

h. Close the table.

Lab 2.1: Create a Table

Scenario

Simon is a developer working for CRONUS International Ltd. CRONUS International Ltd. has decided to start selling Microsoft Dynamics NAV training courses as its business.

Simon must create a table to record course information and set several keys so that his users have the option for a different sorting sequence for the records in the table.

Objectives

The objectives are:

- Show how to create a table, set basic field properties, and then create primary and secondary keys for the table.

Exercise 1: Create a Table

Exercise Scenario

Simon creates a table to store the information about courses.

Task 1: Create a Table

High Level Steps

1. Create a table to keep the course information. This includes the course code, course name, course description, course duration in days, course type (instructor led, e-learning or remote training), course prices, and whether it is an active course or not.
2. Set the OptionString property for the **Type** field.
3. Compile, save and close the table.
4. Add records to the new table.

Detailed Steps

1. Create a table to keep the course information. This includes the course code, course name, course description, course duration in days, course type (instructor led, e-learning or remote training), course prices, and whether it is an active course or not.
   a. In the **Object Designer's Table** list, click **New**. The Table Designer opens.

b. Type the following in the Table Designer:

Field No.	Field Name	Data Type	Length
10	Code	Code	10
20	Name	Text	30
30	Description	Text	50
40	Type	Option	
50	Duration	Decimal	
60	Price	Decimal	
70	Active	Boolean	
80	Difficulty	Integer	
90	Passing Rate	Integer	

2. Set the OptionString property for the **Type** field.
   a. Open the **Properties** window for the **Type** field and then set the following property:
   b. **OptionString**: Instructor Led,e-Learning,Remote Training
   c. Close the **Properties** window.

3. Compile, save and close the table.
   a. Compile and save the table by clicking **File > Save As**. The **Save As** dialog box opens.
   b. Type "90010" in the **ID** field and "Course" in the **Name** field. Make sure that the **Compiled** check box is selected and then click **OK**.
   c. Close the table by closing the Table Designer.

4. Add records to the new table.
   a. Run table **90010**, **Course** from the Object Designer.
   b. Type the following:

Code	Name	Description	Type	Duration	Price	Active	Difficulty	Passing Rate
80040	Installation & Configuration	Basic knowledge of installation and configuration	Remote Training	2	1,000	Yes	5	75
80041	Finance	Basic knowledge of finance	Instructor Led	3	1,500	Yes	7	80

Code	Name	Description	Type	Duration	Price	Active	Difficulty	Passing Rate
80042	C/SIDE Introduction	Introduction to programming	Instructor Led	5	2,500	Yes	8	80
80043	Introduction	Introduction to Microsoft Dynamics NAV	Remote Training	2	1,000	Yes	4	60
80049	Application Setup	Basic knowledge of application setup	e-Learning	2	1,000	Yes	5	65
80050	Business Intelligence	Basic knowledge of Business Intelligence	e-Learning	1	500	Yes	5	65
80055	C/SIDE Solution Development	Advanced topics in programming	Instructor Led	5	2,500	Yes	10	75

c. Close the table.

Module Review

Module Review and Takeaways

Tables are the most fundamental objects in Microsoft Dynamics NAV and are used to store information in the Microsoft Dynamics NAV database. C/SIDE provides the possibility for developers to create new tables, modify existing tables, properties, and triggers, sort table data, and set relations between tables to look up data from other table fields. Microsoft Dynamics NAV also provides a powerful feature to perform fast calculations for large records with special table fields.

Many standard tables are available in Microsoft Dynamics NAV, and C/SIDE enables developers to extend the application to suit their users' requirements by creating custom tables or modifying existing tables.

Developers are encouraged to work in Microsoft Dynamics NAV, become familiar with the various table types that are found in the application, and practice creating new tables. This helps developers understand the users' needs and become skilled at providing tables that meet those needs and integrate smoothly into the application.

Test Your Knowledge

Test your knowledge with the following questions.

1. What is the option string for the **Reserve** field in the **Customer** table?

_____

_____

_____

_____

2. What is the value of the **TableRelation** property of the **Global Dimension 1 Code** field in the **Customer** table?

_____

_____

_____

_____

3. What class of field is the **Date Filter** field in the **G/L Account** table?

4. What table property defines the default lookup page for the table?

5. What field property is used to force the user to enter a value into a **Primary key** field?

6. What number range can you use for new fields that are added to the **Customer** table?

7. To view customers in order by the city that they live in, what do you need to change in the table description?

8. What do you need to change in the **Salesperson Code** field in the **Customer** table so that the user is only able to look up **Salespeople** that have a **Commission %** larger than zero?

Test Your Knowledge Solutions

Module Review and Takeaways

1. What is the option string for the **Reserve** field in the **Customer** table?

   MODEL ANSWER:

   Never,Optional,Always

2. What is the value of the **TableRelation** property of the **Global Dimension 1 Code** field in the **Customer** table?

   MODEL ANSWER:

   "Dimension Value".Code WHERE (Global Dimension No.=CONST(1))

3. What class of field is the **Date Filter** field in the **G/L Account** table?

   MODEL ANSWER:

   FlowFilter

4. What table property defines the default lookup page for the table?

   MODEL ANSWER:

   LookupPageID

5. What field property is used to force the user to enter a value into a **Primary key** field?

   MODEL ANSWER:

   NotBlank

6. What number range can you use for new fields that are added to the **Customer** table?

   MODEL ANSWER:

   50,000 to 99,999.

7. To view customers in order by the city that they live in, what do you need to change in the table description?

   MODEL ANSWER:

   You must define a key that starts with the City field.

8. What do you need to change in the **Salesperson Code** field in the **Customer** table so that the user is only able to look up **Salespeople** that have a **Commission %** larger than zero?

   MODEL ANSWER:

   You must define the **Table Filter** property for the table relation.