

NADABAS

National Account Data Base System

Version 2.1

September 2006

Developers Guide

NADABAS is developed by

Søren Netterstrøm
Statistics Denmark
sne@dst.dk

in cooperation with

Jan Redeby
Statistics Sweden
redeby@iLesotho.com

and

Instituto Nacional de Estatística
Maputo
MoÁambique

Contents

Contents	3
National Account Data Base System	4
Basic concepts.....	4
The database.....	4
The Excel workbooks.....	5
Linking to the database	6
Define tables and columns in database	7
Putting it together.....	9
Functions for creating links.....	10
Register Workbook	11
More options for table definitions.....	12
More about GetDB.....	14
Ignore	15
Create new KeyFamily.....	17
Appendix 1. Database structure.....	18
Base tables.....	18
Key Families	18
Optional tables	19

National Account Data Base System

This is a short description of an Excel add-in that allows for a flexible interaction between a series of Excel Spreadsheets and a Microsoft Access database. It is designed with a system for National Accounts in mind.

The intension is that all links between spreadsheets should be avoided and replaced by links through the database. Any sheet may publish data (by saving them in the database) that can then be obtained by other sheets.

Basic concepts

The system is based on the assumption, that the National Account System can be perceived as a set of time-series, where each time-series is identified by a number of classifications.

As an example, there may be a series for Product P120, transaction code P1, with Final Values (F) containing current prices (concept code cp), so in this case the values from four classifications identify fully the series. By adding time (YYYY, YYYYQ or YYYYMM) each individual value of the series is identified.

All series identified with values from the same set of classifications are said to belong to the same key-family. So the series P130, P1, F, cp and P130, P2, F, kp (constant prices) would belong to the same key-family.

The database

The database consists of tables. Each table has rows and columns. Each row in a table represents one specific cell in a source spreadsheet.

There will be one table for each key-family where data will be saved. This table will have one column for each classification, that will hold the identifier of the series and one column to hold the time-dimension value.

The database will further have one column to hold the actual value, one column to hold any comment and one column to hold the formula of the source cell.

Finally there is one column for username, one for timestamp, one for name of the Excel workbook from where data was saved and one for the name of the named area within the sheet containing the cell.

Figure 1 is an example for a Product table.

Products : Table			
	Field Name	Data Type	
🔑	Year	Text	Classification
🔑	ProductCode	Text	Classification
🔑	TransactionsCode	Text	Classification
🔑	Concept	Text	Classification
🔑	FinalOrProv	Text	Classification
	Value	Number	The cell value
	Comment	Memo	Comment from cell
	Formula	Text	Formula in cell
	UserName	Text	Name of user who saved this
	TimeStamp	Date/Time	Time and date when saved
	ExcelFile	Text	The actual file that saved data
	DataArea	Text	The name of the Data Area

Figure 1

Apart from this table, a table named Products_GET and a number of Views are needed. To create proper a proper structure, use the function Create New KeyFamiliy as described later under Administration.

For further details on the Database, look to appendix 1.

The Excel workbooks

For each Excel workbook that should save and/or load data from the database you need to define the relations between cells of a given area and a key-family (or table in the database).

An area from which data is saved or loaded must be given a name, Mark the area and set the name of the area in the box that normally would hold the coordinates (located to the left on the formula toolbar) or use insert name from the menu. Not all cells in the area need to be loaded or saved, but if they are, they should all belong to the same key-family. If not you must divide the area into a number of smaller areas. Each area must be a contiguous range of cells

The area must be defined in such a way, that each row and each column can be connected to a specific value of a classification.

In the Example below, rows are connected to transaction and columns to concept.

	A	B	K	L	M	N
1	F	Final Values				
2	1997	EQUILIBRIOS DE OFERTA E PROCURA: 1997	Valores K		Valores C	
3	P005	FEIJÃO	Original	Ajuste	Original	Ajuste
4		OFERTA TOTAL (p.c.)	909,953	926,577	904,264	920,415
5	P1	PRODUÇÃO (p.p.)	493,910	532,594	519,100	559,756
6	P11	Produção mercantil	175,891	214,575	184,862	225,518
7	P11a	Empresarial	4,650	4,650	4,887	4,887
8	P11b	Não empresarial (fam-comerc)	171,241	209,925	179,975	220,631
9	P12	Produção para utilização final propria(Fam.não co	318,019	318,019	334,238	334,238
10	P13	Outra produção não mercantil				
11	P13a	Outra produção não mercantil, paga				
12	P13b	Outra produção não mercantil, não paga				
13	P7	IMPORTACOES (cif)	8,298	8,297	9,026	9,026
14	D211	DEREITOS A IMPORTACAO	786	786	855	855
15	D212	IVA				
16	MAR	MARGENS DE DISTRIBUICAO	406,959	384,900	375,283	350,778
17	MARa	Margens de comercio	394,066	369,299	361,711	334,360
18	MARb	Margens de transporte	12,893	15,601	13,572	16,418
19		IQ, IP, %MgDist	2.200	1.721	1.927	1.490
20						
21		PROCURA TOTAL (p.c.)	909,953	926,577	904,264	920,415
22	P2	PROCURA INTERMEDIA (p.c.)	42,124	42,124	44,272	44,272
23	P2a	Autoinsumo e intrainsumo (p.b.)	42,124	42,124	44,272	44,272

Figure 2

The data area starts in C4 and goes to N45 (outside what is shown).

Column A does contain the codes of transaction but there are no codes for concept. See later. Observe that column A is not part of the data area.

The rows 18, 19 and 20 are included in the area, but will not be connected to the database as they have no code associated. Similar rows could be excluded if they only hold calculated values. Columns can be excluded in a similar way, so it is not all cells in the data area that actually is connected with the database.

Linking to the database

To link the data area to a spreadsheet, we use a definition area. The definition area has the same shape as the data area and is like the data area named area. It has one extra column to the left and one extra row at the top. It is normally put on a special sheet that is hidden once the application is ready for production.

The following figure shows the definition area corresponding to the previous data area. The definition area is marked with a yellow background in this example.

DataDef		=			
	A	B	C	D	E
1	Template 1	Indices Q		Indices P	
2		Original	Ajuste	Original	Ajuste
3		C03Qo	C03Qa	C03Po	C03Pa
4	P1	PutDB	PutDB	PutDB	PutDB
5	P11	PutDB	PutDB	PutDB	PutDB
6	P11a	PutDB	PutDB	PutDB	PutDB
7	P11b	PutDB	PutDB	PutDB	PutDB
8	P12	PutDB	PutDB	PutDB	PutDB
9	P13	PutDB	PutDB	PutDB	PutDB
10	P13a	PutDB	PutDB	PutDB	PutDB
11	P13b	PutDB	PutDB	PutDB	PutDB
12	P7	PutDB	PutDB	PutDB	PutDB
13	D211	PutDB	PutDB	PutDB	PutDB
14	D212	PutDB	PutDB	PutDB	PutDB
15	MAR	PutDB	PutDB	PutDB	PutDB
16	MARa	PutDB	PutDB	PutDB	PutDB
17	MARb	PutDB	PutDB	PutDB	PutDB
18					
19					
20					
21	P2	PutDB	PutDB	PutDB	PutDB
22	P2a	PutDB	PutDB	PutDB	PutDB

Figure 3

Observe that column A contains the Transaction code and row 3 the Concept Codes (C03Qo etc). Cell B4 corresponds to the upper left corner of the data area, in this case C4 in the Figure 2.

Those cells in the definition area that has the values PutDB corresponds to the cells that will actually be saved. In case the cells should be loaded, the value should be GetDB. Remark lines 18 to 20, corresponding to the rows with no code have no PutDB.

If the actual workbook contains several data areas with the same structure, for instance one for each year or for several products, they can share the same definition area.

Define tables and columns in database

The next step is to create a link to the database table representing the key-family. This is done in a separate named area, the table definition area.

Figure 4 shows a table definition area linking the area about to the product table. The first column contains the names of database elements (table and columns), the second column contains information on how each column participates. The first row should contain the name of the table and should be followed by one row for each column in that table.

49	Products	Table	---
50	Year	Constant1	---
51	ProductCode	Constant2	---
52	TransactionsCode	RowID	---
53	Concept	ColID	---
54	FinalOrProv	Constant3	---
55	Value	Value	---
56	Comment	Comment	---
57	Formula	Formula	---
58	UserName	Username	---
59	TimeStamp	Timestamp	---
60	ExcelFile	Origin	---
61	DataArea	DataArea	---
62			---
63			---

Figure 4

As you can see, the columns names correspond to the names in Figure 1.

The following table shows the keywords used in the second column

Product	Table	Product is the table name in the database
Year	Constant1	Associates Year with a constant, see later
ProductCode	Constant2	Associates ProductCode with another constant, see later
TransactionCode	RowID	Row Id (in figure 5) gives the TransactionCode
Concept	ColID	Column ID (in figure 5) give the Concept
Value	Value	The column Value will hold the Cell Value
Comment	Comment	The column Comment will hold any comments
Formula	Formula	The column Formula will hold the Formula if any
UserName	Username	Username has the name of the user that saved the data
Timestamp	Timestamp	Timestamp when the cell was updated
ExcelFile	ExcelFile Origin may be used for compatibility	The Excel file from where the cell was written This does not include the path. For that reason, all sheets associated with the database should have a unique name regardless of the folder where they are located
DataArea	DataArea	Name of the named data area containing the source cell

Putting it together

Now it is time to put the elements together.

¥

To do this, the Datasheet must have an area named DBLinks. This should be the exact name of this area, as that is what tells the Add-In that the sheet is linked to the database.

	A	B	C	D	E	F	G
1	Name of Data Area	Name of TabelAreaDefinitic	Name of DbDefinition	Constant1	Constant2	Constant3	
2	Data1996	DataDef	TableDef	1996 P005	F	PutDb	
3	Data1997	DataDef	TableDef	1997 P005	F	PutDb	
4	Data1998	DataDef	TableDef	1998 P005	F	PutDb	
5	Data1999	DataDef	TableDef	1999 P005	F	PutDb	
6	Data2000	DataDef	TableDef	2000 P005	F	PutDb	
7	Data2001	DataDef	TableDef	2001 P005	F	PutDb	
8	Data2002	DataDef	TableDef	2002 P005	F	PutDb	

Figure 5

DBlinks may have one or many rows, one for each data area in the workbook.

Be careful that DBLinks cover all the rows (data areas) used. To change the scope of DBLinks (or any other named area), use Insert, Name from the Excel Menu

Column 1 is the name of the DataArea (Figure 2),

Column 2 is the name of the Definition Area (Figure 3) and

Column 3 is the name of the area defining table and columns in the database (Figure 4).

Column 4, 5 and 6 are the constants referred to in the table definition area.

There must be at least 1 Constant Column and can be up to 9 in the DBlinks area.

The last column (Column 7) indicates that the Definition area only contains PutDB. Acceptable values are

PutDB, æGetDB, Mixed and GetTemp.

PutDB and GetDB indicate that the area contains only PutDB or GetDB cells respectively.

Mixed is used for areas containing both types.

GetTemp is similar to GetDB, but when data are loaded into such an area, this is not recorded in the database. This may be used for test purposes or for temporary sheets loading data.

This column is used to inform the add-in which lines to further examine for each operation.

In the example above, the sheet contains 7 identically structured data areas, where each data area is associated with a specific year. All areas are for the same Product and Contains final values. The constants are used to reflect this. Refer to Figure 5.

With DBLinks, the linking of the spreadsheet to the database has been completed.

Functions for creating links

Show templates in the menu shows those worksheets that contain the DBLinks area and the definitions of tables and data area structures. This option is only available to administrators.

If these are visible, they can be hidden using **Hide Template**

If the basic template sheet (the sheet with DBLinks) is visible, the menu has the **Design** item added containing a submenu with the functions needed when designing a workbook for Nadabas.

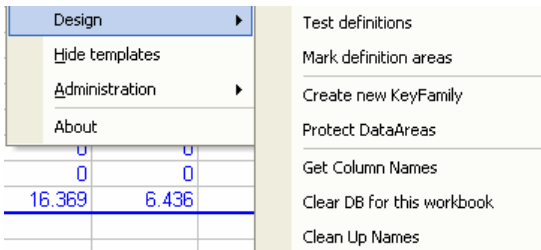


Figure 6

Test Definitions will test your definitions and issue a message describing any errors.

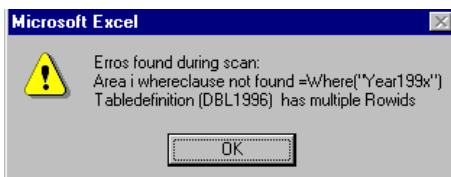


Figure 7

Press Ok and correct any Errors.

If no errors are found you get the message box



Figure 8

It should be noted, that this test is executed any time you load, reload, save or refresh data, so if there are errors you get the first message box and operations are halted.

GetColNames is a help to build the DB definitions.

Enter the name of the table in the first line and select the Cell. Then press GetColNames and all the Column names from the Database will appear on the following lines. All you have to do now is fill in the second Column of the Db Definitions. Note that you must be pointing to the correct database. For those fields that are not Dimensions, even the second column is prefilled.

Products	Table
Year	
ProductCode	
TransactionsCode	
Concept	
FinalOrProv	
Value	Value
Comment	Comment
Formula	Formula
UserName	UserName
ExcelFile	ExcelFile
TimeStamp	TimeStamp
DataArea	DataArea

Figure 9

Mark definition areas will simply colour the areas holding DbLinks, all data definition and database definition areas in order to verify that the named area are correctly defined.

Protect Data Areas locks all cells in data areas that either has a GetDB reference or includes a formula. If the sheet is unprotected at the beginning of the operation all other cells of the sheet will be unlocked. If the sheet is already locked, those cells already locked will remain locked.

After completing locking of cells, the sheet is protected.

You may of course unprotect the sheet and do further protection as needed (headers etc).

Clear DB can be used to erase all cells in the database, that at any point of time was saved from this sheet. It will go through the tables of all the key-families referenced by any table definition area found in DBLinks and erase any cell, where Excel file matches the name of the current sheet. This function should be used with great caution, but during the design of a larger set of spreadsheet, you may realize that the sheet is saving too many cells or values that should be saved from another sheet. Clear DB save gives you the option to clean the database. Be aware, that if you intend to completely remove reference to a key-family, Clear Db Save should be executed before the change is made.

It will also clean up information about any cells loaded from this workbook

Register Workbook

When a workbook is ready for production it should be included into the menu of the system. To do so, use **Register Workbook** (found under **Administration** if the workbook is not registered).

Figure 10

Fill in the group name. Existing names are found in the drop down box, but you can enter a new name if you need.

Fill title to give a more descriptive name of the sheet.

One registered the sheet is available through the menu system.

More options for table definitions

In Figure 5, the links to the classification was done using RowID, ColID or a Constant (in the DBLinks table).

As mentioned before, DBLinks may have up to 9 columns with constant, and the constant are the referenced as Constant1, Constant2, ...Constant9. The number of Columns in DBLinks determines the highest number that can be used.

Cross-classified columns or rows

In the examples so far all rows was identified by a single classification and similar for the columns. It is however possible to use cross classifications.

		E1	E1	E2	E2
		F1	F2	F1	F2
C1	D1	PutDB	PutDB	PutDB	PutDB
C1	D2	PutDB	PutDB	PutDB	PutDB
C1	D3	PutDB	PutDB	PutDB	PutDB
C2	D1	PutDB	PutDB	PutDB	PutDB
C2	D2	PutDB	PutDB	PutDB	PutDB
C2	D2	PutDB	PutDB	PutDB	PutDB

Figure 11

In Figure 11, two classifications are used to identify each row, and to classifications are used identify each column. So there would be 4 classification columns for this in the database. In the table definition this would look like

Cclass	Rowld1
Dclass	Rowld2
Eclass	Colld1
Fclass	Colld2

Figure 12

There can be up to nine cross-classifications, but probably more than 2 would indicate a much too complex data area.

Alternatives to constants

For those classifications in the database that are not linked to a RowID or a ColID Constant is one option as descried above.

Where

Another option is to link such a classification with the value of a single named Cell.

49	Products	Table
50	Year	Constant1
51	ProductCode	Constant2
52	TransactionsCode	RowID
53	Concept	ColID
54	FinalOrProv	Where("FinalProv")
55	Value	Value

Figure 13

In this example, the value for FinalOrProvisional will be taken from a Cell named FinalProv. This option should be used with care. If the user enters a wrong value in the named field, data will not be saved or loaded correctly.

However, if you would produce a large number of workbooks having a similar structure, i.e. one workbook for each product, you could provide a template for such workbooks. On the first sheet, the Product Code should then be entered when customizing for a specific product (in a named cell). In this case, you don't need to go to the DBLinks area and change a Constant for the product code.

WhereLocal

Another option for working with multiple similar data areas is to use the WhereLocal function. This requires that each area is located on a separate sheet.

For Year this would look like

Year	WhereLocal(Year)
------	------------------

Figure 14

Year should be a data area defined on the sheet containing data for 1997. This cell should have the value 1997. Say that this is the cell A2. The sheet containing 1998 should have 1998 in cell A2; the sheet for 1999 should have 1999 in A2 and so on.

Where local could be used as Where, when a template is created, but the user could also change say range of years (the year associated with each data sheet).

Get Cellinfo

Get CellInfo can be used for a cell with GetDB and PutDB to obtain information about those cells actually links with this definition.

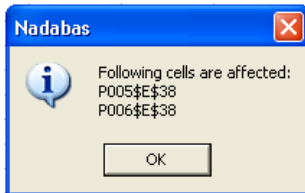


Figure 15

More about GetDB

As earlier stated the function of GetDB is to mark cells that should be loaded from the database. When the user asks to Load Data, NADABAS scans DBLINKS for rows marked as GetDB or Mixed. For each row it scans the corresponding Data Definition Area and locates cells marked as GetDB.

For each such cell, NADABAS tries to obtain the corresponding value in the database.

If a value is found, the corresponding cell in the data area is located. The value is then inserted into that cell together with any comments from the original cell. Before the value is inserted, NADABAS test if the target cell contains a formula. The first time a formula is found, this message box is shown

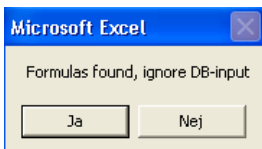


Figure 16

Press Yes to ignore input from the database and keep the formulas or press No to accept input from the database (and destroy the formula).

If more formulas are found they are handled the same way (without showing the message box).

If no value is found in the database, the cell is cleared except if it contains a formula.

When all data areas has been scanned, the following message box shows information about the operation

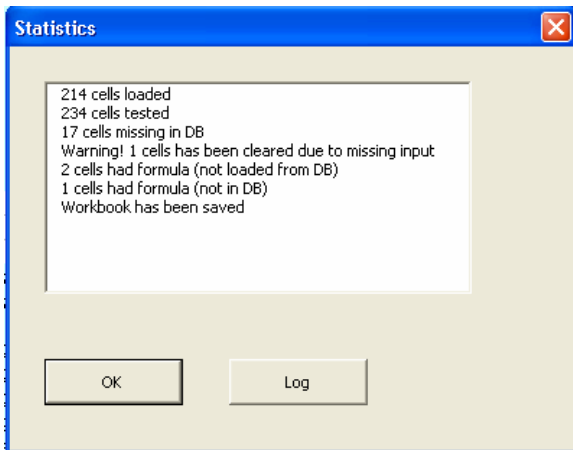


Figure 17

If you press log, you will get more details

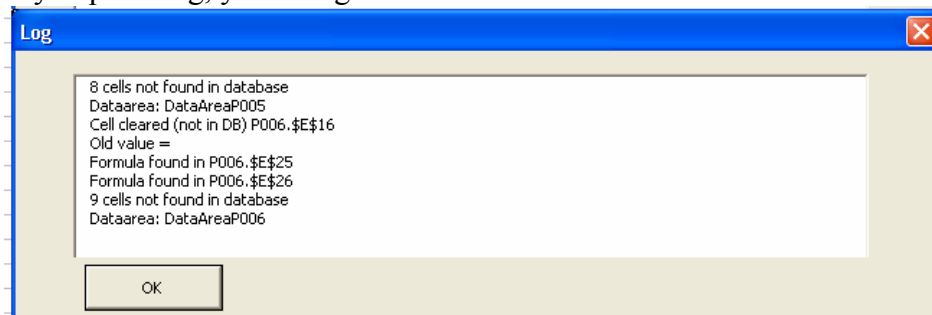


Figure 18

Ignore

Workbooks are often based on a standard template, i.e. all workbooks related to products have the same structure and are based on the same template with the main difference that each workbook refers to a specific product.

Within a single workbook there may one sheet for each year having the same structure and using the same data definition area.

In such cases, by design, not all cells marked with a GetDB will be loaded from the database because the sources are not in a form suitable for this. It is more convenient to enter the data directly to the sheet.

Rather than trying to eliminate unused GetDB it is recommended to input data in a special input area of the workbook (it may be on the same sheet), and then put a formula into the cell that was intended to load from the DB. Due to the behaviour of GetDB, as discussed above, the cells that contain formulas will not be affected in case there is no data in the DB for the cell.

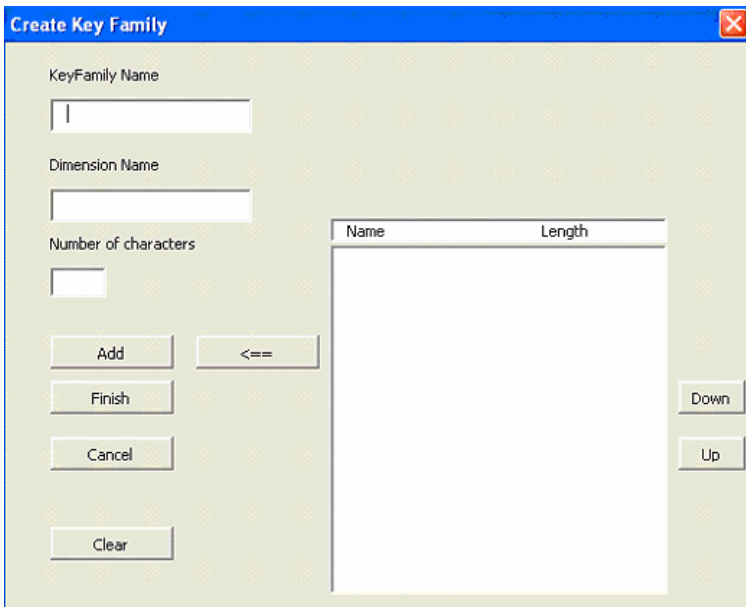
If there should be data in the DB, you must select to ignore such data or override the formulas.

The designer can select the behaviour of the system by adding an extra constant column to DBLinks. If this constant contains the constant IGNORE, the system will ignore all DB-input to any cell holding a formula without asking.

Create new KeyFamily

Create new KeyFamily is used to create the necessary tables and views in the database for a new key family.

The following windows should appear



The screenshot shows a window titled "Create Key Family" with a blue header bar. Inside, there are three text input fields: "KeyFamily Name", "Dimension Name", and "Number of characters". Below the "Number of characters" field are five buttons: "Add", "<==", "Finish", "Cancel", and "Clear". To the right of these fields is a table with two columns, "Name" and "Length". Below the table are two buttons: "Down" and "Up".

Figure 19

In the first textbox, enter the name of the Key family

Then enter the first Dimension Name and the maximum number of characters needed to hold any code from this dimension. If you don't know, enter 50, which should be sufficient in any case. Then press the Add button and name and length are more into the right side table.

Enter the next dimension name and length pair, press Add on so on.

The **←** button can be used to remove an item from the table (first select the item).

The **Down** and **Up** buttons are used to alter the orders of the items if you like.

Once your satisfied with you're dimensions, press **Finish**, and all tables and views needed are created.

Cancel leaves this window without making any new Key Family

Clear clears the whole table window.

Appendix 1. Database structure.

The database is used to hold all data that are shared between two or more workbooks.

Before any key families are created the workbook contains only three tables. As new key families are added, a set of tables and views are created for each key family.

It is strongly recommended no to perform any updates and other maintenance directly to the database using Microsoft Access. The only exception is to compress the database if needed for perform reasons. All other maintenance should be performed using the interface via the Nadabas as described above.

The information in this appendix is only intended only as information.

Base tables

The system has three base tables, Administrators, Keynames and Messages.

Administrators hold the UserId of persons designated as administrators of the system. If the table is empty, any user is considered to be an administrator.

Keynames contains the names of the key families defined for the database. The table is used, when the system needs to browse information in all key families.

Messages holds all messages in the system that has not been deleted. Each message has a unique number as key.

Key Families

When a new key family is created, the system creates one table to hold information that is saved and on table to keep track of all loading of data (successful or not). If the same piece of data (identified by the dimensions) is saved or loaded more than once, only the latest information is saved. The data holding data is given the name the key family, the second has _GET appended, i.e. Products and Products_GET for the key family is Product.

The tables consists of the dimension variables and variables to hold the content of the cells saved and information about when data are saved, the active workbook and the UserID of the user loading or saving data.

The system needs 4 predefined Queries (Views) for each key family. For Products, these would be named

Chains_products
MissUpdates_Products
NotInDb_Products
ToUpdate_Products

Chains_Products gives information about pair of workbooks that are saving (source) and loading (target) the same cell (or set of cells).

MissUpdates_Products hold information about workbooks that needs to be updated because the source workbooks have been updated after data has been loaded.

NotInDb_Products are used to track unsuccessful loads. When a workbook loads cells, information about each cell is saved in Products_GET. If there exists no corresponding cell in Products, we have a missing link that will be reported by NotInDB_Products.

ToUpdate_Product are used to speed up reloading of data. It returns only those cells a given sheet needs in order be reloaded, i.e. those cells that have been saved (changed) after they have been loaded last.

Optional tables

Documents are created whenever the first document is registered. It holds information about the document, name and path, level of association and name of group and file as appropriate.

Permissions hold information about permissions. It holds the name of a workbook and the name of a user in each row.

If **Permissions** exists then all workbooks are protected by default.

If **Password** exists, it puts protection on change database by requiring a password to perform the operation. Password must have one column named Password that holds the actual password (Text. 8 characters).

Year is created if set years are used to delimit the years handled during load and save operations.

Version may hold the Version number of NADABAS AddIn currently in use. If present, the system at startup will check the version of NABAS actually used with the version number stored in the table. If not the same, a warning message will be issued.

BasePath is used to simplify the maintenance of the system. If BasePath exists, then all information of Path for documents and excel files (in Documents and Workbooks respectively) may start with a !. This is then automatically replaced with the content of BasePath.

BasePath should hold only one row and one column (BasePath). Its automatically created when using the command Set Base Folder (under Administration).