



JOIN RELATIONAL DATABASE

VERSION 6

DECOS
DEFINING TOMORROW

INHOUDSOPGAVE

1.	INHOUDSOPGAVE	1
	JOIN RELATIONAL DATABASE	1
1	DATABASE CONCEPT	4
1.1	Item Table	6
1.2	Item Types	6
1.2.1	Container	6
1.2.2	Folder	8
1.2.3	Document	8
1.2.4	Blob	9
1.2.5	Address	9
1.2.6	Contact	9
1.2.7	Action	10
1.2.8	Activity	11
1.2.9	USRINFO	11
1.2.10	Tablecon	12
1.2.11	Email	12
1.2.12	Mailing code	12
1.2.13	Report	13
1.2.14	Postcode table	14
1.2.15	Workflow	14
1.2.16	Catchword	15
1.2.17	Thumbnail	16
1.3	Field settings and Rights	17
1.3.1	Itemprofile	18
1.3.2	Itemfield	19
1.3.3	Userprofile and Itemprofile_user	19
1.4	Relations between Items	20
1.5	Additional item types and relations	21
2	DATABASE STRUCTURE	22
2.1	Relational Model	22
2.2	Table Descriptions	22
2.2.1	Itemtype	22
2.2.2	Item	23
2.2.3	Itemrel	25
2.2.4	Reltype	26
2.2.5	Decos_user	26
2.2.6	Userprofile	27
2.2.7	Itemprofile	27
2.2.8	Itemprofile_user	28
2.2.9	Itemright	28
2.2.10	Itemfield	29
2.2.11	Decos_string	31
2.2.12	Decos_language	32

2.2.13	String_type	32
2.2.14	App_status	32
2.2.15	User_var	33
2.2.16	Global_var	33
2.2.17	Decos_audit	33
2.2.18	Audit_type	34
2.2.19	Decoscache	35
2.2.20	Book_field_formula	35
2.2.21	Formula	36
2.2.22	Formula_value	36
2.2.23	ItemProfilerel	36
2.2.24	Global_Memo	37
2.2.25	Decos_Lock	37
2.2.26	Catchword	38
2.2.27	Catchwordrel	38
2.2.28	Thumbnail	39
2.2.29	Postcode table (obsolete)	39
2.3	Unique Record IDs	39

1 DATABASE CONCEPT

There are various standard objects designed for handling respective kind of data. For example a company may have many contacts. To maintain contacts a Contact object is defined in JOIN. Similarly for handling Addresses, an Address item has been defined. Thus every object is designed for storing a particular kind of data or information. JOIN defines several items to implement folder and document registering, maintaining contacts, maintaining addresses, assigning activities etc.

To generalize code and database usage all objects are placed in a new table, called the **Item**. The possible types of Items are stored in the ItemType list. The type of **Item** is kept with each Item as a key to the ItemType list.

Possible item types are listed in the table below.

Possible Itemtype_keys	Description
CONTAINER	Container
FOLDER	Folder
DOCUMENT	If itemprofile_key2 == EMAIL Else Document
BLOB	Attachment, Word or WordPerfect document, Scan, text file etcetera
ADDRESS	Address
CONTACT	Contact person
ACTION	Action (what to do)
ACTIVITY	Activity (what to do to what, when and by whom)
USRINFO	User profile
TABLECON	Table value, such as: Regular table value Classification value Mailing code for address, contact person, document or folder
REPORT	Report definition
WORKFLOW	Process (what to do, when and by whom) escalation of work to next level. Auto decision making and automatic distribution of work.
Catchword	Purpose of Catchword is to allow user to select one or more terms that are not exactly the preferred terms, but still very useful terms that is used for searches
Thumbnail	JOIN has a feature of thumbnail, it is has thumbnail view of document and scans same as window's thumbnail

The relations between the **Items**, for instance between an address and the address file, are kept in the **Itemrel** table. Thus to know to which address file the address is linked, the link between the address and the address file is searched. The link indicates the address file of the address, if CRM is enabled then we can link contact individually with out using address/address file.

The possible relations are stored as **Reltypes**.

For example: The documents are grouped in a container. The relation is indicated with a BDOCUMENT.

A complete list of all possible Reltypes can be found in Appendix **Error! Reference source not found..**

Field settings are stored in an **Itemprofile**, which contains **Itemfields**. They contain the representation and input options of data. For instance the way a document is represented on screen: which fields are shown, where and how long they are. Other settings are stored in several other data objects that will be discussed in the next chapter.

User profile and settings are stored in a **Userprofile**. The settings for a specific user are derived from such a profile.

Linking Items

When information from one **Item** is linked to another **Item** this should be done by adding an **Itemrel**,

Within the **Itemrel** one ITEM_KEY is the PARENT_KEY and the other the CHILD_KEY. In principle this defines a many-to-many relation, limitations like 'only one can be linked' are artificially imposed, defined in the RELTYPE table.

In case of contact address relation ship some time person contact is linked with more than one mailing group and he/she want that he received mail by one group in particular scenario. To handle this particular scenario **itemrel_extension_key** field is added in **itemrel** table, where we store the **item_key** of item table whose **itemtype_key= itemrel_extension**. In all other relation it will be null.

Settings

Settings that apply to all users should be stored in **Global_var**.

Settings that are different per user should be stored in **User_var**. **User_var** is a **Global_var** with a userkey.

App_Status contains settings that also are different per user and per item. **App_Status** is a **Global_var** with a userkey and an itemkey.

Audit

Audit is handling by two tables **Audit_type** and **Decos_Audit**. In **Audit_type** table has type of audit, actual data related to audit is stored in Decos_Audit.

Formula

Any formula that is used in JOIN is handled by three tables (**Book_field_formula**, **Formula**, **Formula_value**). **Book_field_formula** having a detail about book and field on which formula is defined. **Formula** is table holding detail about formula. **Formula_Value** hold a value that is used to show a value for a formula.

Additional item types and relations

The combination of **ITEM**, **ITEMREL**, **ITEMTYPE** and **RELTYPE** makes a complete 100% data driven application possible. If customers need to store objects which do not fit in the set of existing item types, the application can be extended with new item types and relation types.

The application will detect the new types immediately and will show the appropriate interface to use the new item type. It allows the new item type to link the item to each possible other item as defined in the database. This makes the application capable of storing any type of object with any type of relations.

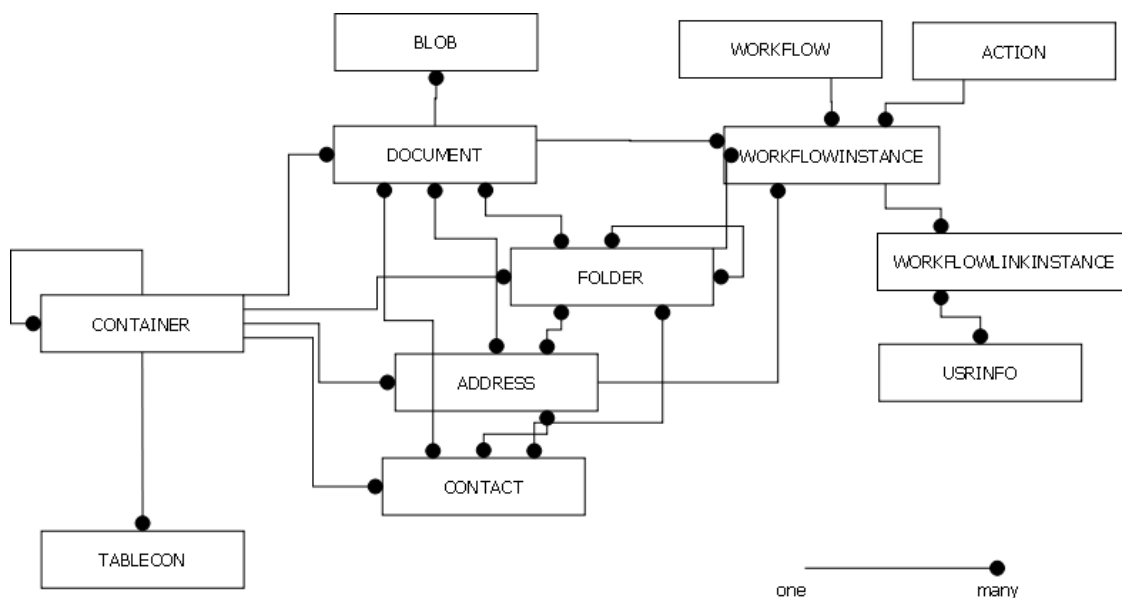
1.1 Item Table

The JOIN database is designed as an object-oriented database. The central object repository is the **Item** table. The item type is defined in the field **itemtype_key**. This field is a reference into the table **Itemtype** that lists all possible item types.

1.2 Item Types

A number of standard item types are always available in a JOIN configuration.

The graph below shows the most standard items and their possible relations.



1.2.1 Container

A container is the highest level container object in the JOIN database. It implements a list of files like Document files, Folder files or Address files as well as a single File (say: one document file). It can also be the container for table values, calculate values, classification values, procedures, and general subfolders.

The highest level container, being the container of a container, is shown in the menu of JOIN.

Possible container children are listed below.

Container child	Description
CONTAINER	All document files
CONTAINER	Document file
DOCUMENT	Document
CONTAINER	All folder files

CONTAINER	Folder file
FOLDER	Folder
CONTAINER	All address files
CONTAINER	Address file
ADDRESS	Address
CONTAINER	All tables
CONTAINER	Table
TABLECON	Table value
CONTAINER	Table for reports
REPORT	Report
CONTAINER	All action files
CONTAINER	Action file
ACTION	Action
CONTAINER	All stdroute files
CONTAINER	Stdroute file
STDROUTE	Stdroute
CONTAINER	All pending files
CONTAINER	Pending files
PENDING ITEM	Pending item
CONTAINER	All dspitem files
CONTAINER	Dspitem file
DSPITEM	DSPITEM
BLOB	All files
BLOB	blob

All these containers hold a name to indicate the children they group. Most of these containers also hold additional data regarding the children. For example: a document file, which is a container of documents, can contain 'n' number of documents. Similarly, the container **'Document Books'** under the **'Maintenance'** menu in JOIN is a container of several such document files.

This is how the container **'Maintenance'** is stored in the **ITEM** table.

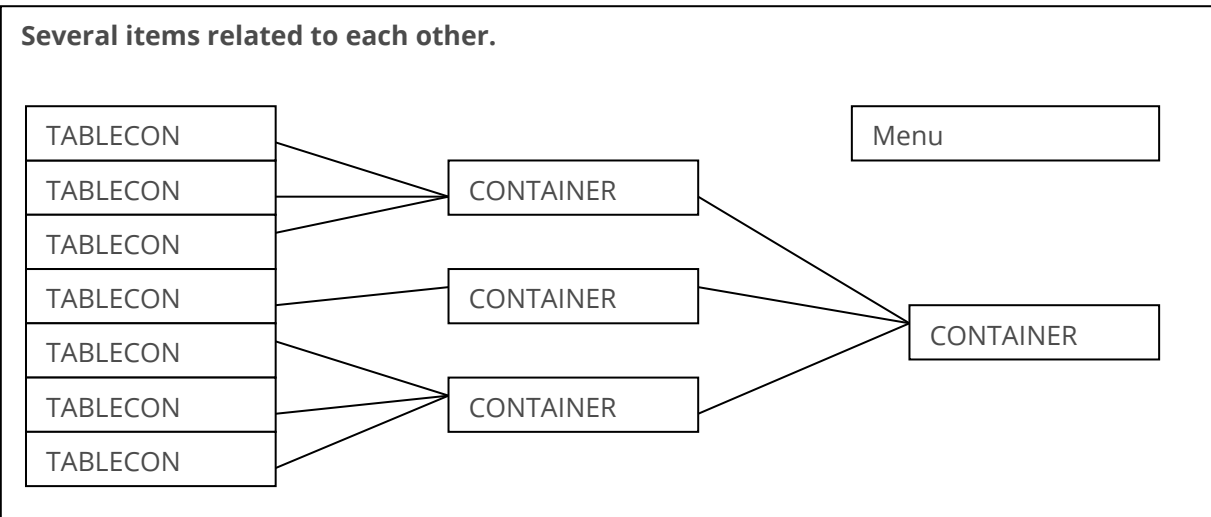
ITEM_KEY	ONDERHOUDMENU
ITEMTYPE_KEY	CONTAINER
ITEMTYPE_KEY2	CONTAINER
STRING_KEY	STR_M_MAINTENANCE

From this information we can deduce that the item is a **'CONTAINER'** item (since the ITEMTYPE_KEY is 'CONTAINER') and it is going to store containers (since the ITEMTYPE_KEY2 is also 'CONTAINER')

This is how a document file, which is a container of documents, is stored in the **ITEM** table:

ITEM_KEY	4CB13E7F8FB8F74C98EE0EF5AF3CDD63
ITEMTYPE_KEY	CONTAINER
ITEMTYPE_KEY2	DOCUMENT
TEXT1	Inkomende Post

This information is enough to indicate that the item is a **'CONTAINER'** item (since the ITEMTYPE_KEY is 'CONTAINER') and it is going to store documents (since the ITEMTYPE_KEY2 is 'DOCUMENT')



1.2.2 Folder

A folder in JOIN is an object that is referred to by a group of related documents. The folder itself can refer to zero or more addresses or contacts.

The table below shows an example of a folder record stored in the **Item** table.

ITEM_KEY	32AF72E0067D1C43B44B42BD7B44E6B0
ITEMTYPE_KEY	FOLDER
ITEMTYPE_KEY2	NULL

This information indicates that the item is an **'FOLDER'** item (since the ITEMTYPE_KEY is 'FOLDER' and ITEMTYPE_KEY2 is NULL)

1.2.3 Document

A document is a document registration object. Typically a document can be a purchase order from a client. After the document arrives in the organization, it is registered. I.e. the document is entered into the system or converted into electronic format. The document is then placed in an appropriate container of document files.

This is how a document, is stored in the **ITEM** table:

ITEM_KEY	6D1BA5070B52254E874D7F23717467EB
ITEMTYPE_KEY	DOCUMENT
ITEMTYPE_KEY2	NULL

This information is enough to indicate that the item is an **'DOCUMENT'** item (since the ITEMTYPE_KEY is 'DOCUMENT' and ITEMTYPE_KEY2 is NULL)

A document can also contain zero or more binary documents (office documents or scanned TIFFs). A document may refer to a list of document copies.

An address is used in a document by copying an address to a document and creating a link between the address and the document. This may seem inconsistent with linking an address to a folder, which is done by a link (Itemrel). There are two reasons for it:

- When the document is send it is send to the address of the addressee at that moment; when the address is changed, the document its address should not change.
- When creating a merge file the contents of the Item record are used, not its links.

1.2.4 Blob

A scanned TIFF file can be attached to a document just like attaching a Word document or e-mail. The blob record stores this information. A blob is a reference to scanned image or binary document. BLOB can be stored as separate files or in a compound file.

Binary documents and TIFFs (scanned images) are stored as two different entity types. Scanned TIFF images are stored in a compound file, all other document files as separate files.

The table below shows an example of a Blob record stored in the **Item** table:

ITEM_KEY	209186006208EC49AF6FC171CAEB8F08
ITEMTYPE_KEY	BLOB
ITEMTYPE_KEY2	NULL

This information indicates that the item is an '**BLOB**' item (since the ITEMTYPE_KEY is 'BLOB' and ITEMTYPE_KEY2 is NULL)

1.2.5 Address

An address is an entry in the address database. Normally contains the company address of an addressee, and can contain a group of zero or more contact persons sharing the same address. A company head office address may refer to a list of branch addresses. An address record stores information like company name, city, country, phone, fax, e-mail, building etc.

The table below shows an example of an address record stored in the **Item** table:

ITEM_KEY	64CD52DEE6E2EB4AAC351DFD1BD0C789
ITEMTYPE_KEY	ADDRESS
ITEMTYPE_KEY2	NULL

This information indicates that the item is an '**ADDRESS**' item (since the ITEMTYPE_KEY is 'ADDRESS' and ITEMTYPE_KEY2 is NULL)

1.2.6 Contact

A contact object contains the data of one contact person linked to zero or more addresses. The contact name can be used as the addressee name in a document. It stores information like the contact person name, official contact info & personal contact info.

Note: - that it is possible that instead of the address of the company that is selected with a document, the personal address of a contact person that is selected with the document could be used and thus copied in the document entry fields.

The table below shows an example of a Contact record stored in the **Item** table:

ITEM_KEY	6832E1A77501D411816E00105A747E91CT
ITEMTYPE_KEY	CONTACT
ITEMTYPE_KEY2	NULL

This information indicates that the item is a '**CONTACT**' item (since the ITEMTYPE_KEY is 'CONTACT' and ITEMTYPE_KEY2 is NULL)

We can link contacts with items directly without using addresses. These contacts are stored the same as contacts linked with addresses. But while we are linking contact to items the data entered into the **itemrel** table is different. In **itemrel**, the **reltype_key** of contact is **contactdoc2** where **reltype_key** for contact is **contactdoc1**.

The table below shows an example of a Contact record stored in the **Item** table:

ITEM_KEY	91B805869EDF71478D39F7F5AEB131A5
ITEMTYPE_KEY	CONTACT
ITEMTYPE_KEY2	NULL

This information indicates that the item is a '**CONTACT**' item (since the ITEMTYPE_KEY is 'CONTACT' and ITEMTYPE_KEY2 is NULL)

1.2.7 Action

An action is a to-do without person or date. It describes what to do. Actions define lookup tables with standard values for the possible activities that can be assigned to a particular kind of item object.

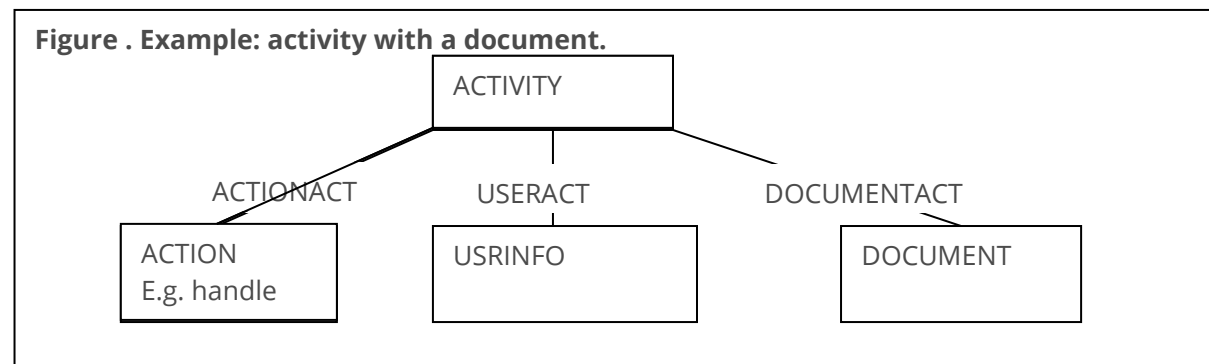
The table below shows an example of an 'Action' record stored in the **Item** table:

ITEM_KEY	PAB
ITEMTYPE_KEY	ACTION
ITEMTYPE_KEY2	DOCUMENT
SUBJECT1	Process

This information indicates that the item is an '**ACTION**' item that defines action for a document.

1.2.8 Activity

An activity is a single task of a certain action type. An example of an activity is the action to send a



receipt of a complaint that should be handled by action holder Graham Barnes before a certain date. An action is linked to exactly one user that has to handle the activity and exactly one item type (document, address, and folder). There also is a link between the action item that defines an activity type and each activity item of that type. In the present system a user can delegate pending activities to other fellow users of the system.

The order in which activities are handled is determined by their sequence number in relation to the parent DOCUMENT/FOLDER/ADDRESS item and by the 'workflow option' setting of the linked ACTION (sequential or parallel). An activity is not yet active if there is a sequential activity that has a lower sequence number and is not handled (i.e., has a NULL handle date).

The table below shows an example of an Activity record stored in the **Item** table:

ITEM_KEY	BHJD6BRMD2H2
ITEMTYPE_KEY	ACTIVITY
ITEMTYPE_KEY2	NULL

This information indicates that the item is an '**ACTIVITY**' item (since the ITEMTYPE_KEY is 'ACTIVITY' and ITEMTYPE_KEY2 is NULL)

1.2.9 USRINFO

An UsrInfo record holds the user data as used within JOIN. This data is mostly related to the user properties like name, password, contact information etc. The application normally only uses the USRINFO record for the current user. No implicit relations to other item object types. An USRINFO record is similar to a contact record, with the difference that it refers to a JOIN user rather than to an external contact.

There are several types of users, such as view clients versus regular users, non-active versus active users, etcetera.

The table below shows an example of a Usrinfo record stored in the **Item** table:

ITEM_KEY	ADMINISTRATOR
ITEMTYPE_KEY	USRINFO
ITEMTYPE_KEY2	CONTAINER
TEXT7	ADMINISTRATOR

This information indicates that the item is an '**USRINFO**' item (since the ITEMTYPE_KEY is 'USRINFO' and ITEMTYPE_KEY2 is 'CONTAINER')
Here the 'Text7' stores the username of the application user.

1.2.10 Tablecon

A table value (or table content) is stored as a TABLECON. This item appears as a choice in the table that pops up when the field is of combo/drop-down type. All the records that appear in a dropdown list have their ITEMTYPE='TABLECON'.

The table below shows an example of a Tablecon record stored in the **Item** table:

ITEM_KEY	6E0430F615E9F845A34A3F3D97516FE7
ITEMTYPE_KEY	TABLECON
ITEMTYPE_KEY2	NULL

1.2.11 Email

An Email is a document registration of an email message. There is no separate EMAIL type in JOIN. Instead, an Email registration is a Document registration with a special item profile. This is because an email is a document.

The Emails are grouped in containers presented as email boxes with an email address as name. A mailbox has at least one action holder. Each email address can have an Inbox, Sent items, etc. The emails are presented as documents with field settings that are more appropriate for emails.

1.2.12 Mailing code

To allow selections on addresses, contact persons, which are often used for exporting selections to make mailings, attaching mailing codes to addresses and or contact persons is possible. There are codes for addresses, contact persons and documents.

Codes are attached to the records through adding a link, the **Itemrel**.

This is possible by adding new column in itemrel table and also adds a new relation type to use contacts directly in mailing codes

For this we add new column to **ITEMREL** table name **ITEMREL_EXTENSION_KEY** and one new relation type name contactmail2. The key that might be inserted into the **ITEMREL** record should refer to an item of **itemtype_key ITEMREL_EXTENSION_KEY**. We can store all kinds of data in this item, so we are capable of creating more extensions for **ITEMREL** records in future.

Following data entry make it clearer.

Data of itemrel table:-

ITEMREL_KEY	PARENT_TYPE	CHILD_TYPE	PARENT_KEY	CHILD_KEY	RELTYPE_KEY	ITEMREL_EXTENSION_KEY
KEY1	ADDRESS	TABLECON	ADDRESS DSYS	MAILINGKEY1	ADDRESSMAIL2	
KEY2	CONTACT	TABLECON	CONTACT PAUL	MAILINGKEY1	CONTACTMAIL1	EXTENSION_KEY1
KEY3	ADDRESS	TABLECON	ADDRESS DSD	MAILINGKEY1	ADDRESSMAIL2	
KEY4	CONTACT	TABLECON	CONTACT KEES	MAILINGKEY1	CONTACTMAIL1	EXTENSION_KEY2

KEY5	ADDRESS	TABLECON	ADDRESSDSE	MAILINGKEY1	ADDRESSMAIL2	
KEY6	CONTACT	TABLECON	CONTACTBEREND	MAILINGKEY1	CONTACTMAIL2	
KEY7	CONTACT	TABLECON	CONTACTGJ	MAILINGKEY	CONTACTMAIL1	EXTENSION_KEY3

The extensions inside the item table should include a reference to the parent address for this instance in this mailing code:

ITEM_KEY	ITEMTYPE_KEY	IT_PARENT_KEY
EXTENSION_KEY1	ITEMREL_EXTENSION	ADDRESSDSD
EXTENSION_KEY2	ITEMREL_EXTENSION	ADDRESSDSE
EXTENSION_KEY3	ITEMREL_EXTENSION	ADDRESSDSE

Note that Berend (as single contact person) is linked using a new RELTYPE named **CONTACTMAIL2**, so we can distinguish between single contact persons and contacts linked to an address item. Add this reltype to the database.

Following figure show how we select contact number from different contact person mailing code

1.2.13 Report

Itemtype_key	REPORT
Text1	Name of the report
Memo	Reportdefinition in XML-format
Text5	Item key of the main report item
Itemtype_key2	BookType

The 'booktype' is the item key (thus not the item type key!) of the parent container that contains the possible main items in this report.

Possible values for BookType are the menu keys

Menu item	Key
Root	M_ROOT
Maintenance	M_FILESLIST
Document files	M_ADDRESSFILES
Folder files	M_FOLDERFILES
Address files	M_DOCUMENTFILES
Application menu	M_APPLICATION
Usermenu	M_USER
Tablemenu	M_TABLES
Classification	M_TABLECLASSIFICATION
Handled	M_TABLEHANDLED
Systemmenu	M_SYSTEM
Itemtypes	M_ITEMTYPES
Actions	M_ACTIONS
Document actions	M_ACTIONDOCUMENT
Folder actions	M_ACTIONFOLDER
Address actions	M_ACTIONADDRESS
Templates	M_TEMPLATES
Templates for documents	M_TEMPLATESDOCUMENT
Templates for folders	M_TEMPLATESFOLDER

Templates for addresses	M_TEMPLATESADDRESS
Document publication codes	M_CODESDOCUMENT
Folder codes	M_CODESFOLDER
Address mailing codes	M_CODESADDRESS
Contact mailing codes	M_CODESCONTACT
Pending items	M_PENDING
Email	M_EMAIL
JOIN email box	M_EMAILDECOS
Reports	M_REPORTS

1.2.14 Postcode table

The Dutch postcode table holds the information to retrieve the street and city based on the postcode and house number.

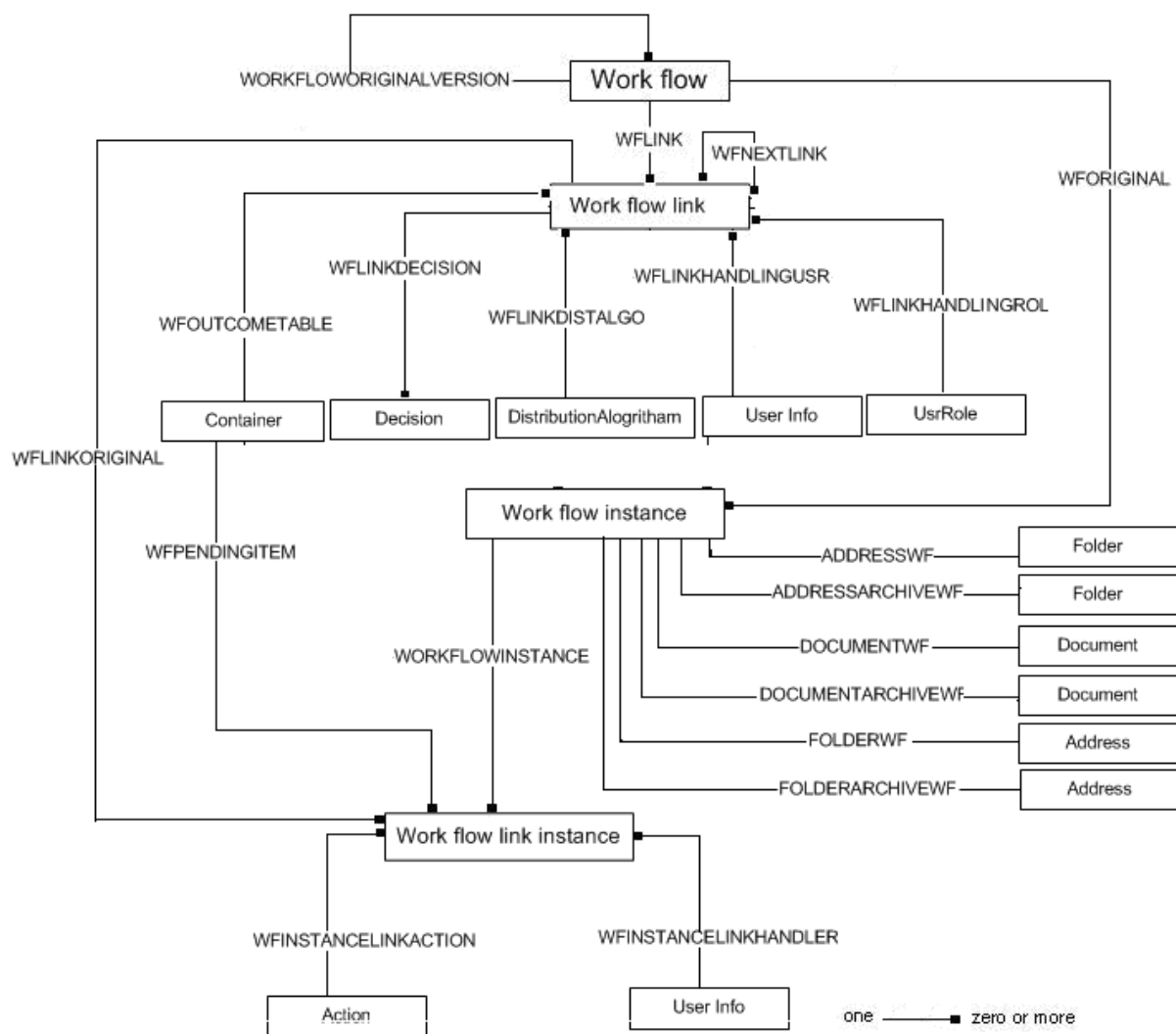
1.2.15 Workflow

A workflow is an automated system to perform certain action. It is designed to handle all activities of a process from start to end.

Workflow is applied to all item types that have possible relations with WORKFLOWINSTANCE items, such as ADDRESS, DOCUMENT and FOLDER..

Workflow are not directly linked with. A relation is generated by a workflowinstance (it is an instance of an original workflow).

Following figure shows the relations of workflow with other items.



The table below shows an example of a Workflow record stored in the **Item** table:

ITEM_KEY	510B0391B07C35488A6F2290E0AAFA7
ITEMTYPE_KEY	WORKFLOW
ITEMTYPE_KEY2	NULL
SUBJECT1	TEST-WORKFLOW
IT_PARENT_KEY	FOLDERWORKFLOWS

This information indicates that the item is an **'WORKFLOW'** item (since the ITEMTYPE_KEY is 'WORKFLOW' and ITEMTYPE_KEY2 is 'NULL')

Here the 'SUBJECT1' stores the name of the workflow and **it_parent_key** used to store whether workflow is on document book, folder book and address book.

1.2.16 Catchword

Purpose of Catchwords is to allow user to select one or more terms that are not exactly the preferred terms, but still very useful terms that is used for searches. In other words, we can assign some words to particular items and if we search by using that assigned word then all the items which having these terms are shown as search result.

The table below shows an example of a **Catchword** record stored in the **Item** table:

ITEM_KEY	C05AC9420AB98C42888984E05C2F8F16
ITEMTYPE_KEY	NULL
ITEMTYPE_KEY2	NULL
IT_PRAENT_KEY	CATCHWORD

Catchword Table important field:-

S.no	Field	Description	Data Store
1	ITEM_KEY	Unique key	- C05AC9420AB98C42888984E05C2F8F16
2	CTW_PATH	Path of the catchword. if it is root then only item_key is stored in CTW_PATH, If catchword is sub value then the previous item item_key is also stored and separated by pipe separator	- C05AC9420AB98C42888984E05C2F8F16 - C05AC9420AB98C42888984E05C2F8F16 7173C1C2E83BAB4AB1610A881DC3E4B3 63705924DA15F045A8128E466336FF3C
3	CTW_DESC	Value that is treated as catch word	- DSE
4	CTW_PATH_DESC	It is like CTW_PATH what only difference is it store description of the path. And value store here is memo field.	Contains friendly descriptions of CTW_PATH
5	CTW_LEVEL	Here level of particular item is stored.	
6	CTW_PREFERRED_DESC	Description related to preference	

1.2.17 Thumbnail

JOIN has a feature of thumbnail, it is has thumbnail view of document and scans same as window's thumbnail. Now user is able to archiving their photos or other images. Size of the images is different in list view and form view

Following table hold all information related to thumbnail.

BLOB_KEY	3BCE45F5804D234882578A88A66EDD19
DOCUMENT_KEY	B283279AD42D344E82E4192E4C8A440D
BLOB_SEQUENCE	1
THUMB_TYPE	2
THUMB_TIMESTAMP	20070117132158
FILE_PATH	d:\decos\data\letters\dsdlogo.gif
THUMB_DATA	BLOB ITEM

BLOB_KEY is the key reference to ITEM_KEY of blob item of item table. DOCUMENT_KEY is the key reference to ITEM_KEY of document item of item table. THUMB_TYPE is used for storing different size of thumbnail, because the thumbnail size within the list of a document book is different as compare to thumbnail used in the form view of an item, we should be able to store several instances of the same thumbnail, but with a different size.

THUMB_TIMESTAMP is used to store latest modified date of the actual file, if it is greater than the modified date inside the thumbnail table, and then the thumbnail is outdated and should be

refreshed. Same should be done with the **ITEM_TIMESTAP** of a BLOB item when the thumbnail of a SCAN item is generated. **FILE_PATH** is the actual path of the file that is used for thumbnail.

The table below shows an example of a BRS Address record stored in the **Item** table:

ITEM_KEY	BRSA0000011641-5161
ITEMTYPE_KEY	Address
ITEMTYPE_KEY2	NULL
IT_PARENT_KEY	A43E14AFFA32B2439F5FD2C4A65611FC

This information indicates that the item is a '**BRS Address**' item (since the ITEMTYPE_KEY is 'Address' and ITEMTYPE_KEY2 is 'NULL')

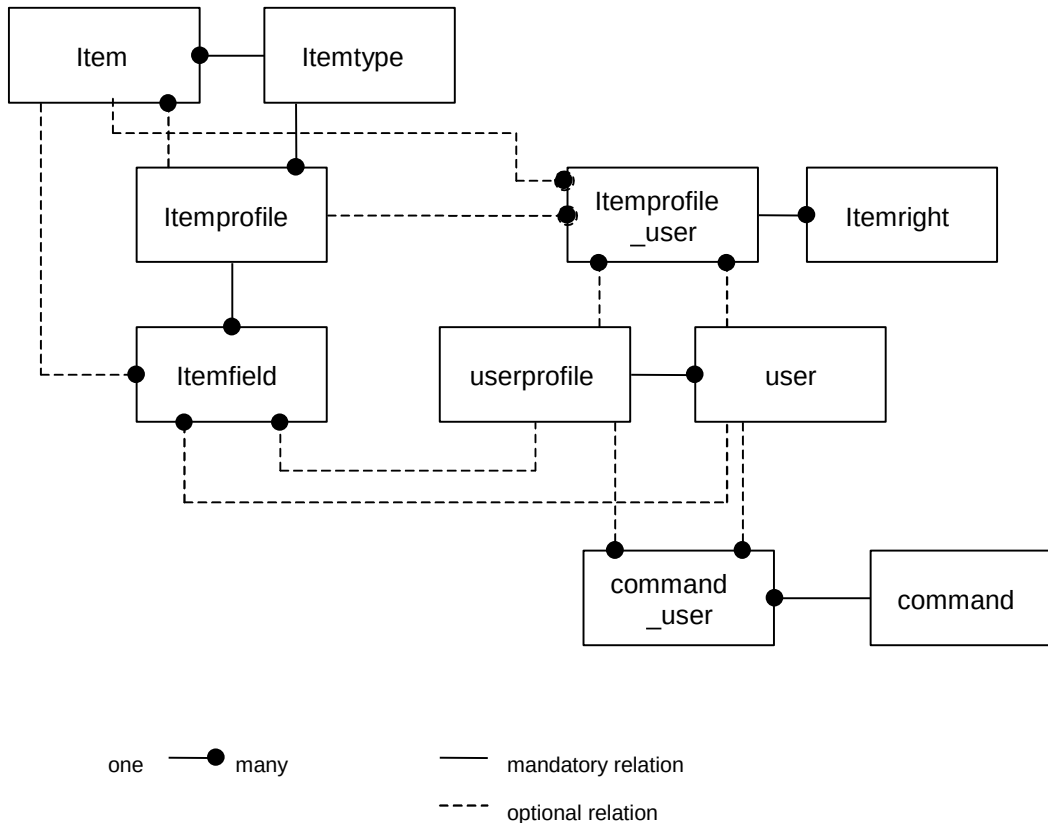
it_parent_key used to store itemkey of the address book container.

1.3 Field settings and Rights

In order to present the Item records such as DOCUMENT, FOLDER etcetera on screen, the layout of these records is set per container of these records. Also permission is set to the container. In other words: the layout of each record in a container and the types of user access are the same for each record in that container. These settings are set through the JOIN Admin utility.

The rights system has been designed to meet the following criteria:

- It must be possible to configure standard properties for each item type
- It must be possible to set specific properties for a particular item or for its children
- It must be possible to configure special field properties (e.g., read-only flag, default value) for a user group or for a specific user and to hide fields or display extra fields.
- It must be possible to grant/revoke user rights to generic control functions (e.g., windows menu options, toolbar buttons, etc.)



1.3.1 Itemprofile

An **Itemprofile** logically groups the fields that are shown for an item or item type. The item profile defines the external representation of the item. An item profile is linked to exactly one item type. For example every JOIN item like document, folder, contact address etc, have their own profile.

It is also possible to create new Itemprofile for existing JOIN items in the JOIN Admin utility of the application. The user can use default Itemprofile for a JOIN item or he can define his own customized Itemprofile for an item.

The table below shows the record for the default Itemprofile of a document.

ITEMPROFILE_KEY	DOCUMENTDEF
ITEMTYPE_KEY	DOCUMENT
ITEM_KEY	NULL
PARENT_KEY	NULL
PROFILE_NAME	NULL
STRING_KEY	STR_DOCDEF
FIXED	PROFILE_DEFAULT ('1')

Here the **ITEMTYPE_KEY='DOCUMENT'** indicates that the profile is for a document. There is not much need of the fields **ITEM_KEY** and **PARENT_KEY** as the Itemprofile can be handled in the **ITEM** table itself (using **ITEMPROFILE_KEY** & **ITEMPROFILE_KEY2**). The fields are result of conversion process from the old system to the new system and may be optionally used to refer a particular item record in the **ITEM** table.

Also it should be noted that the field **Fixed** equals PROFILE_DEFAULT ('1') which indicates that the field is the default profile defined by the system.

Each item optionally points to an item profile for its own settings (itemprofile_key field) and/or to an item profile defining settings for its child items (itemprofile_key2 field). This makes it possible to set field properties for specific items or to define non-standard field properties for the children of a particular item. The latter is used for container (book) items so that a group of books can share a specific non-standard item profile for their items.

1.3.2 Itemfield

An Itemfield record describes the properties (attributes) of a field in an Itemprofile such as readonly, used_len, and etcetera. See also paragraph 2.2.7. Each item profile has a list of item fields associated that define the default field properties for the item or item type.

When the fields for an item must be accessed or displayed, the software selects an itemprofile for that particular item. If not found, the default profile for the item type is used instead.

For each field that must be displayed, the software selects all itemfield records associated with the itemprofile where the userprofile_key or user_key field is either Null or equal to the current userprofile or user.

The table below indicates the field list for the default document Itemprofile 'DOCUMENTDEF'. Also against each field we have some of the properties like Field Name, which is the actual field name in the ITEM table, 'USED_LEN' which indicates the length allowed to a user for entering the data.

ITEMFIELD_KEY	ITEMPROFILE_KEY	FIELD	USED_LEN	DISPLAY_WIDTH_LIST	READ ONLY	PASSWORD_FIELD
A000	DOCUMENTDEF	SEQUENCE	6	6	1	0
A001	DOCUMENTDEF	MARK	25	25	0	0
A002	DOCUMENTDEF	SUBJECT1	200	200	0	0

From the above table we can see how an Itemprofile decides the field settings (Field names + their properties).

1.3.3 Userprofile and Itemprofile_user

Users are grouped in **user profiles**. The **itemprofile_user** table key links user profiles or users to item profiles or items. In this context it is possible to define access rights per user or per user group for each item profile or item (usually a container).

Given that:

- The profile (field settings) of a file, used to present a list of records or present an entry screen, is not stored with the current user nor with the current file.

- The **itemprofile_user** can have references to the **Decos_user** or to a **Userprofile**, and to an **Itemprofile** or an **Item**.

There are several ways to retrieve the profile.

The profile to use is determined according to the following schedule:

1. The **Decos_user** record gives the user key. The current file stored in a CONTAINER **Item** gives the item key. Inside the **Itemprofile_user** table a match is searched between the item_key and user_key.
2. If not found: the **Decos_user** is linked (through the userprofile_key field of Decos_user) to a **Userprofile**. Inside the **Itemprofile_user** table a match is searched between the item_key and userprofile_key.
3. If not found: the CONTAINER **Item** is linked to an **Itemprofile** (through the Itemprofile_key field of Item). Inside the **Itemprofile_user** table a match is searched between the itemprofile_key and user_key.
4. If not found: inside the **Itemprofile_user** table a match is searched between the itemprofile_key and userprofile_key.
5. If not found: the default field settings for the item are used.

Thus it is possible to create a set of field settings for, say, a document file, that are used for certain users only.

1.4 Relations between Items

As said before, it is possible to relate one item to another. For example we can associate an address/contact to a document. The possible relationships between these items are stored in the **RelType** table.

The actual relationship between one record to another is maintained in the **ItemRel** table. In the **ItemRel** table, the column PARENT_KEY indicates the ITEM_KEY of the parent item in the **Item** table. The CHILD_KEY is the ITEM_KEY of the child item in the **Item** table. The RELTYPE_KEY indicates what relationship exists between the parent item and the child item.

It should be noted that just mentioning the RELTYPE_KEY would have been enough in the **ItemRel** table, instead of including the PARENT_TYPE and CHILD_TYPE field. These fields are deliberately added for the sake of performance improvement when it comes to retrieving the related records.

Chapter **Error! Reference source not found.** shows all possible relations in JOIN.

Now consider the following **Reltype** record

RELTYPE_KEY	PARENT_TYPE	CHILD_TYPE	STRING_KEY	PARENT_QTY	CHILD_QTY
BDOCUMENT	CONTAINER	DOCUMENT	STR_BDOCUMENT	1	MANY

Where the RELTYPE_KEY is 'BDOCUMENT'. With the PARENTTYPE_KEY = 'CONTAINER' and CHILDTYPE_KEY = 'DOCUMENT', it means that the record is defining the relationship between a *Container* object and a *Document* object.

With PARENT_QTY=1 and CHILD_QTY=MANY, it means that one container of documents can store many document items.

If we see a record for any Documents File in the **ItemRel** table, we will get the RELTYPE_KEY to be 'BDOCUMENT'. Indicating that it is the container of documents.

TABLE: **Item**

ITEM_KEY	ITEMTYPE_KEY	ITEMTYPE_KEY2	TEXT1
3A7F23FE87AAF9 4B88151DF55592 FF82	CONTAINER	DOCUMENT	Inkomende Post

TABLE: **ItemRel**

ITEMREL_KEY	PARENT_KEY	CHILD_KEY	PARENT_TYPE	CHILD_TYPE	RELTYPE_KEY	itemrel_extension_key
500F7596CD409D4F 94072EF11BB6FB5E	3A7F23FE87AAF94B 88151DF55592FF82	A7FA3EFEA3FB9F 47ACCA88834BB 599B8	CONTAINER	DOCUMENT	BDOCUMENT	NULL

For a document file '*Inkomende Post*', the RELTYPE_KEY is 'BDOCUMENT' in the **ItemRel** table.

Thus the **RelType** table defines standard relationships and these relationships are used in the **ItemRel** table to indicate the relationship between the records entered in the system.

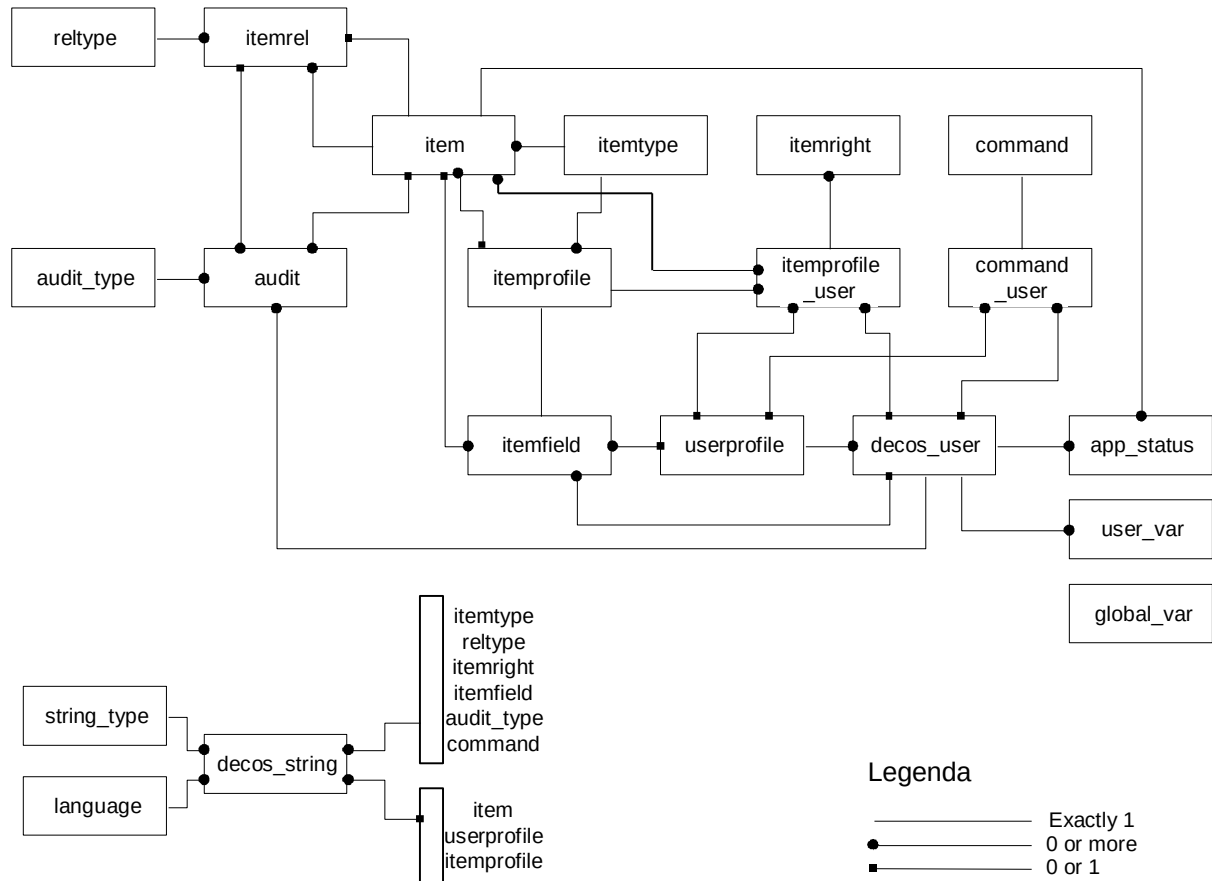
1.5 Additional item types and relations

The combination of ITEM, ITEMREL, ITEMTYPE and RELTYPE ensures a complete 100% data driven application. If customers need to store objects which do not fit in the set of existing item types, the application can be extended with new item types and relation types.

The application will detect the new types immediately and will show the appropriate interface to use the new item type. It allows the new item type to link the item to each possible other item as defined in the database. This makes the application capable of storing any type of object with any type of relations.

2 DATABASE STRUCTURE

2.1 Relational Model



2.2 Table Descriptions

This section lists all table definitions in Oracle format (first three columns) with explanations (last column). Fields that make up the unique key for each table are shown in **bold**.

Sybase uses different table definitions; for instance the "Long raw" is called a "Text" in Sybase.

Where Oracle can have at most one "Long raw" per record, in Sybase there can be more.

2.2.1 Itemtype

This table contains all the types of items that can be made within the database. This table is read-only. Users of the application cannot adjust the values.

Examples are container, document, address, folder, table-value, and action.

Table definition

itemtype_key	Character(32)	NOT NULL,	Object type
string_key	Character(32)	NOT NULL	Reference to decos_string for object name

Example:

Address files container

Itemtype_key ADDRESS

String_key key to Decos_String STR_ADDRESS being "Address"

The ITEMTYPE_KEY indicates the JOIN object item and the STRING_KEY when put in the DECOS_STRING table gives the description of the object item.

2.2.2 Item

This table contains record of the types that are defined in the **Itemtype** table. No other types are allowed. The table contains the document files, address files, folder files, documents etc. that can be changed with the application.

Sometimes **itemtype_key** referred to parent, sometimes to child. Therefore it was split. Itemtype_key2 indicates child in case **itemtype_key** referred to CONTAINER. **Itemprofile_key** is hardly used. Itemprofile_key2 is used for permissions.

Table definition

item_key	Character(32)	NOT NULL	
itemtype_key	Character(32)	NOT NULL	Object type
itemtype_key2	Character(32)		Related object type (optional)
Itemprofile_key	Character(32)		Optionally refers to a itemprofile for this object
Itemprofile_key2	Character(32)		Optionally refers to a itemprofile for the children of this object
String_key	Character(32)		Reference to decos_string for standard name (optional)
Item_deleted	Date		If not null the record is logically deleted
Mark	Varchar(250)		
subject1	Varchar(250)		
subject2	Varchar(250)		
Document_date	Date		
Received_date	Date		
Processed	Character(1)		PROCESSED ('J' or 'Y') or NOTPROCESSED ('N')
Company	Varchar(250)		
Mailaddress	Varchar(250)		
Zipcode	Varchar(10)		
City	Varchar(250)		
Country	Varchar(20)		
Phone1	Varchar(20)		
Phone2	Varchar(20)		
Phone3	Varchar(20)		
Fax1	Varchar(20)		
Fax2	Varchar(20)		
Url	Varchar(250)		
Salutation	Varchar(60)		
Initials	Varchar(50)		
Surname	Varchar(250)		
Firstname	Varchar(50)		
Prefix	Varchar(50)		
Title	Varchar(250)		
Function	Varchar(250)		
Department	Varchar(250)		
Sex	Varchar(8)		
Email1	Varchar(250)		
Email2	Varchar(250)		
Email3	Varchar(250)		
Text1	Varchar(250)		

Text2	Varchar(250)		
Text3	Varchar(250)		
Text4	Varchar(250)		
Text5	Varchar(250)		
Text6	Varchar(250)		
Text7	Varchar(250)		
Text8	Varchar(250)		
Text9	Varchar(250)		
Date1	Date		
Date2	Date		
Date3	Date		
Date4	Date		
Date5	Date		
Date6	Date		
Date7	Date		
Date8	Date		
Num1	Numeric(12,2)		
Num2	Numeric(12,2)		
Num3	Numeric(12,2)		
Num4	Numeric(12,2)		
Num5	Numeric(12,2)		
Num6	Numeric(12,2)		
bol1	Character(1)		
bol2	Character(1)		
bol3	Character(1)		
bol4	Character(1)		
bol5	Character(1)		
bol6	Character(1)		
bol7	Character(1)		
bol8	Character(1)		
bol9	Character(1)		
bol10	Character(1)		
Memo	Long raw		
It_parent_key	Character(32)		Holds the key of the parent (container) that has the primary relation with this item record
It_sequence	Numeric		Holds the sequence number this record has within the primary relation.
Item_timestamp	Varchar(14)		
Item_created	Datetime		
Item_created_by	Varchar(32)		
Item_applies_to	Varchar(250)		
Barcode	Varchar(20)		
Archived	Char(1)		

ITEM_CREATED and ITEM_CREATED_BY are used for ITEM records which are not yet successfully saved. Untill then, these records should only be visible for its creator.

1. If a user creates a new item record, ITEM_CREATED_BY is set to his user key
2. When the record passes the validation routine, also ITEM_CREATED is set to the current date and time. When validation is unsuccessful, ITEM_CREATED is not filled.

Except for Administrators, users are only able to see item records that meet this condition:
ITEM_CREATED IS NOT NULL OR ITEM_CREATED_BY IS NULL OR ITEM_CREATED_BY='<USERKEY>' where <USERKEY> is the key of the logged-in user.

This has the effect that users can only see their own invalidated records and records where ITEM_CREATED_BY is null (standard items or items inserted by the conversion program).

Example:

The table value "Public" which is part of the table for category, is used as follows:

Field	Description
item_key	Unique code (generated by the application)
itemtype_key	Always contains the value TABLECON
subject1	Table value: "Public"
string_key	Standard (translatable) expansion
Text1	Table value, same as Subject1: "Public"

All other fields are unused.

2.2.3 Itemrel

This table contains the relations that exist between records in the Item table. This table makes it possible for a record in the Item to reference another record in the Item table. For example, each document references the document file it belongs to.

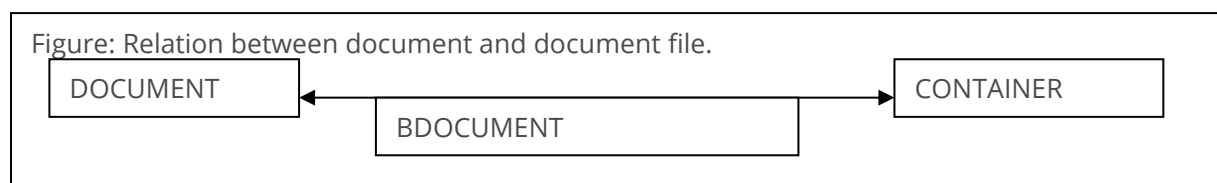
The field reltype_key in this table references the Reltype table. This makes it possible to use named relationships. This can be useful when, for example, a document is linked to multiple addresses. The reltype can be used to specify for example that the relation is the post address or an address where a copy of the document should be sent to. When a letter is made in the application with the document registration, the user has the choice to use the post address or the other addresses that might be linked to this document.

The parent - child relationship that can be made this way between Item records makes it possible to show all related records of a certain parent record.

The field's parent_type and child_type reference the Itemtype table. The sequence field gives a sequence number that is unique within the relation a parent record has with records in the Item table of a certain Itemtype type.

Possible relation types are described in Appendix **Error! Reference source not found..**

An example is the relation between document and document file that indicates that the



document is part of the document file.

Table definition

itemrel_key	Character(32)	NOT NULL	Unique record id
parent_key	Character(32)	NOT NULL	Item_key of parent item
child_key	Character(32)	NOT NULL	Item_key of child item
parent_type	Character(32)	NOT NULL	Itemtype_key (item type) of parent item
child_type	Character(32)	NOT NULL	Itemtype_key (item type) of child item
Sequence	Numeric(9,0)	NOT NULL	Sequence number of child in list.
reltype_key	Character(32)	NOT NULL	Relation type (reference into reltype)
primary_rel	Character(1)	NOT NULL	1: This is the primary link to the child (rather than just another reference). If the primary

			relation is deleted, the child itself and all other relations it has must also be deleted.
itemrel_deleted	Date		If not null the record is logically deleted
itemrel_extension_key	Varchar(32)		Item_key of the item that contains extra meta data about the relation

2.2.4 Reltype

Reltype is the table with information about all the possible relation types that can be made between records in the Item table. The values of this table are used in the Itemrel table to specify the relation. A reltype record has a string_key that can be used to name the relation type.

Each reltype also defines the kind of relation that can be made between the two object types (parent_qty and child_qty can both be '1' or 'MANY'). Note that a CONTAINER can have relations to all other object types, but each container can only contain a particular child type (stored in itemtype_key2 field) or another container.

Possible relation types are described in Appendix A.

The relation between a document and a document file is called the BDOCUMENT.

Table definition

reltype_key	Character(32)	NOT NULL	Relation type
parent_type	Character(32)	NOT NULL	Itemtype_key for parent object type
child_type	Character(32)	NOT NULL	Itemtype_key for child object type
string_key	Character(32)	NOT NULL	Reference to decos_string naming relation type
parent_qty	Varchar(4)	NOT NULL	'1' or 'MANY'
child_qty	Varchar(4)	NOT NULL	'1' or 'MANY'

Folder is part of a folder file which is a container:

reltype_key	BFOLDER
parent_type	CONTAINER
child_type	FOLDER
string_key	"Folder file"
parent_qty	MANY
child_qty	MANY

2.2.5 Decos_user

This table contains the users that can use the application. The access_code field is used to store the user's password. The userprofile_key determines the rights and settings a user gets in the application. The item_key field references a USRINFO record in the Item table, used to store personal information about the user.

User ADMINISTRATOR has access_code "QI6\$#" and has access to all data.

Table definition

user_key	Character(32)	NOT NULL	Encrypted user logon name, must be unique.
user_kcv	Character(16)		Key check value for encrypted user logon name. Used to locate user record.
access_code	Character(32)		Encrypted password (32 hex characters).
userprofile_key	Character(32)	NOT NULL	Reference to user profile defining rights for this user
item_key	Character(32)	NOT NULL	Reference to 'USRINFO' item for this user
item_kcv	Character(16)		Key check value for encrypted item_key. Used to locate user record.

2.2.6 Userprofile

A User profile defines a group of users. Itemfield settings can be dependant on a user profile or be specific for an individual user.

Table definition

Userprofile_key	Character(32)	NOT NULL	
Language_code	Character(2)		Default user language for this profile
string_key	Character(32)		Reference to decos_string for standard profile name
profile_name	Varchar(50)		User-defined profile name (optional)

Example:

Userprofile_key	67YTR45WROMNCGHSAJKHS8MNBDRQM JU
Language_code	'EN'
string_key	NULL
profile_name	"Sales department users"

2.2.7 Itemprofile

This table links Itemfield settings to an item type and optionally to specific items or their children (an item record may refer to an item profile for itself or for its children).

Table definition

Itemprofile_key	Varchar (32)	NOT NULL	Unique id
itemtype_key	Varchar (32)	NOT NULL	Refers to the item type for this profile
item_key	Varchar (32)		Optionally refers to a specific item (obsolete)
parent_key	Varchar (32)		Optionally refers to a specific parent container (obsolete)
string_key	Varchar (32)		Reference to decos_string for standard profile name
profile_name	Varchar(250)		User-defined profile name (optional)
Fixed	Character (1)	NOT NULL	PROFILE_USERDEF ('0'): user-defined itemprofile PROFILE_DEFAULT ('1'): standard itemprofile for the item type PROFILE_PREDEF ('2'): other pre-defined item profile
Addprofile_Key	Varchar(32)		UNKNOWN
Docprofile_key	Varchar(32)		UNKNOWN
Fldprofile_key	Varchar(32)		UNKNOWN
Itemprof_timestamp	Varchar(14)		UNKNOWN
Dspprofile_key	Varchar(32)		UNKNOWN

Example:

Itemprofile_key	'DOCUMENTDEF'
itemtype_key	'DOCUMENT'
item_key	NULL
parent_key	NULL
string_key	'STR_DOCDEF'
profile_name	NULL
Fixed	PROFILE_DEFAULT

2.2.8 Itemprofile_user

The itemprofile_user table links user profiles or users to item profiles. In this context it is possible to define access rights per user or per user group for each item profile.

Table definition

Itemprofile_user_key	Character(32)	NOT NULL	Unique id
Item_key	Character(32)		Item profile the record refers to. Must be filled if itemprofile_key is NULL.
Itemprofile_key	Character(32)		Item profile the record refers to; it can also be other than NOT NULL (thus: it can be NULL). Must be filled if item_key is NULL.
Itemright_key	Character(32)	NOT NULL	Defines set of rights for this record
Userprofile_key	Character(32)		If not null: user profile this record applies to
User_key	Character(32)		If not null: specific user this record applies to

Example:

Itemprofile_user_key	HGQWNEOWYGDSHGDHSBYFW6543HGDS76T
Item_key	POZXHBAGSFSLN2566NBZCFASDJSDGTF7
Itemprofile_key	NULL
Itemright_key	UYT65VMNSADU85HGAS54KJBDS485875E
Userprofile_key	UHDSIYDG87587ADGSDF86587HDSGASVD
User_key	NULL

Note the following possible relations:

Field	Relations with
Itemprofile_user_key	None
Item_key	Item CONTAINER, field Item_key; Itemfield, field Item_key
Itemprofile_key	Itemprofile, field Itemprofile_key; Itemfield, field Itemprofile_key
Itemright_key	Itemright, field Itemright_key
Userprofile_key	Userprofile, field Userprofile_key; Decos_user, field Userprofile_key; Itemfield, field Userprofile_key
User_key	Decos_user, field User_key; Itemfield, field User_key

2.2.9 Itemright

These values define the operations that a user can do on a certain type of record. The contents of this table are used in the itemprofile_user table.

A right (ITEM_RIGHT_KEY value) can be a combination of 'C' (change), 'A' (add) and 'D' (delete) if modification rights are granted. If modification is not allowed, the right is set to 'R' (read) or 'N' (no access). Read right is implied if a user has change, add or delete rights.

Table definition

Itemright_key	Character(32)	NOT NULL	
string_key	Character(32)	NOT NULL	Reference to decos_string for display

Examples:

Itemright_key	string_key
'N'	'STR_NOACCESS'
'R'	'STR_READ'
'C'	'STR_CHANGE'
'CA'	'STR_CHANGEADD'
'CAD'	'STR_CHANGEADDDDEL'

2.2.10 Itemfield

The field settings for a record in the Itemprofile table are stored here. Fields can be visible in the list and/or form views of the application and a user may or may not have the right to modify a field.

Table definition

Itemfield_key	Character(32)	NOT NULL	Unique record id
Itemprofile_key	Character(32)	NOT NULL	Reference to item profile
Field	Varchar(25)	NOT NULL	Field name in item table
Item_key	Character(32)		Optionally refers to specific item
Userprofile_key	Character(32)		Optionally refers to specific user profile
User_key	Character(32)		Optionally refers to specific user
String_key	Character(32)	NOT NULL	Reference to decos_string for field name
Mandatory	Character(1)	NOT NULL	BOL_TRUE ('1'): mandatory field BOL_FALSE ('0'): not mandatory field
Table_key	Character(32)		If not null: reference to 'CONTAINER' for 'TABLECON' list
Tablevalues	Character(1)		BOL_TRUE ('1'): if only values from selection table are allowed
Used_len	Numeric(3,0)		Maximum allowable length for this object variant
Display_width_form	Numeric(3,0)		Field width displayed in form view (characters)
display_width_list	Numeric(3,0)		Field width displayed in list view (characters)
show_in_list	Character(1)		BOL_FALSE ('0'): if a field may not be shown in list view
show_in_form	Character(1)		BOL_FALSE ('0'): if a field may not be shown in form view
form_sequence	Numeric(3,0)		Defines display order of fields within form view
list_sequence	Numeric(3,0)		Defines display order of fields within list view
Defaultvalue	Varchar(250)		Default value
default_is_macro	Character(1)		BOL_TRUE ('1'): if default value is macro string, e.g., "%USER%"
Autonum	Character(1)		BOL_TRUE ('1'): if auto increment number field
Uniquevalue	Character(1)		BOL_TRUE ('1'): if value must be unique within the parent container
Readonly	Character(1)		BOL_TRUE ('1'): if display only field in this profile
Password_field	Character(1)		BOL_TRUE ('1'): if display as a password field
Max_one_tablevalue	Character(1)		BOL_TRUE ('1'): if only one value can be selected from table: table_key
Calc_key	Character(32)		Key to calculate formula
Picture	Character(32)		Input mask, e.g. all characters must be digits
Usenewtable	Character(1)		

Limited_Entry	Character(1)		To restrict field only for user who has right to see limited entry field
Table_Level	Character(1)		Table level used when one table value is depending upon other value.
Tooltiptext	Varchar(250)		To show tool tip for item field

Note: if any of the attribute field values is NULL for a UserProfile-specific or user-specific ItemField record, the value should be looked up in the generic ItemField record for the same Itemtype/field.

Example:

Itemfield_key	YTREQWPO98FV34DE34JKLOIUYBVCVC37
Itemprofile_key	IUY67TGVVCAS4326KBGVVDPMKNH9804GB
Field	Company
Item_key	NULL
Userprofile_key	NULL
User_key	NULL
String_key	STR_COMPANY
Mandatory	BOL_TRUE
Table_key	NULL
Tablevalues	BOL_FALSE
Used_len	200
Display_width_form	200
display_width_list	40
show_in_list	BOL_TRUE
show_in_form	BOL_TRUE
form_sequence	1
list_sequence	1
Defaultvalue	NULL
default_is_macro	BOL_FALSE
Autonum	BOL_FALSE
Uniquevalue	BOL_FALSE
Readonly	BOL_FALSE
Password_field	BOL_FALSE
Max_one_tablevalue	BOL_FALSE
Calc_key	NULL
Picture	NULL
Usenewtable	0
Limited_Entry	1
Table_Level	2
Tooltiptext	NULL

2.2.11 Decos_string

This table makes it possible to use different languages in the application. String_key, language_code and string_type form the unique identifier for this table. The language code the user has chosen can then be used to find the string that is needed in the application. If a language code is not found in the table the default language that Dutch (language_code: NL) is used. String_type can be used to select context-specific strings, e.g., singular or plural object name.

Table definition

string_key	Character(32)	NOT NULL	Used to identify string in object
Language_code	Character(2)	NOT NULL	Language code (user preference or default from user profile)
string_type	Character(32)	NOT NULL	
Description	Varchar(250)		
Fixed	Character(1)	NOT NULL	STRING_PREDEF ('1'): pre-defined string STRING_USER ('0'): user-modifiable

Example:

Field	Dutch string	English string
String_key	'STR_ADRFOLD'	'STR_ADRFOLD'

Language_code	'NL'	'EN'
String_type	'SINGULAR'	'SINGULAR'
Description	'geadresseerde'	'addressee'
Fixed	STRING_PREDEF	STRING_PREDEF

2.2.12 Decos_language

All the languages that are supported in the application are mentioned in this table.

Table definition

Language_code	Character(2)	NOT NULL	Possible values: see examples
string_key	Character(32)		Reference to name of language

Examples:

Language_code	String_key
'NL'	'STR_DUTCH'
'EN'	'STR_ENGLISH'

2.2.13 String_type

This table describes the type of a string. The types plural and singular are the most used. The type button or title can be used in the application to define special text on screens.

Table definition

string_type	Character(32)	NOT NULL	Possible values: see examples
string_key	Character(32)		Reference to name of string type

Examples:

String_type	String_key
'PLURAL'	'STR_PLURAL'
'SINGULAR'	'STR_SINGULAR'
'TITLE'	'STR_TITLE'
'BUTTON1'	'STR_BUTTON1'
'BUTTON2'	'STR_BUTTON2'

2.2.14 App_status

This table keeps session status for the internet application per menu. The information is used to backtrack through the screens when the user confirms or cancels sub screens.

Table definition

User_key	Character(32)	NOT NULL	
Menu_key	Character(32)	NOT NULL	Identifies current window in intranet application
Stack_level	Numeric(2,0)	NOT NULL	Used to navigate through nested windows This is unused and cannot be used at all. It was meant as a future extension. Note that it is sometimes emptied (which should not be required).
Session_variable	Varchar(100)	NOT NULL	Variable name
Session_value	Varchar(250)		Variable value

Examples:

USER_KEY	145.18.104.21	192.0.0.111
MENU_KEY	ADDFILEGlobal	DOCFILEIN2001

STACK_LEVEL	1 (since unused)	1 (since unused)
SESSION_VARIABLE	RELKEY_ACTIVITY	TAB_CONTACT
Session_value	ADDRESSACT	5

2.2.15 User_var

This table keeps session status and persistent user preferences per user. It is a general-purpose per-user information store, comparable to the Windows HKEY_CURRENT_USER registry hive.

User_var replaces the table Menu_status in the prototype.

Table definition

user_key	Character(32)	NOT NULL	
Session_variable	Varchar(50)	NOT NULL	Variable name
session_value	Varchar(250)		Variable value

Example:

User_key	ADMINISTRATOR
Session_variable	LANGUAGE
Session_value	NL

2.2.16 Global_var

Global_var keeps persistent application settings that are applicable to all users.

Table definition

session_variable	Varchar(50)	NOT NULL	Variable name
session_value	Varchar(250)		Variable value

Example:

SESSION_VARIABLE	'OCRREINDEXSTARTED'
SESSION_VALUE	STARTED'

2.2.17 Decos_audit

Decos_audit keeps an audit trail of all added, changed and modified Items or item relations per user and date/time.

Table definition:

audit_key	Character(32)	NOT NULL	
user_key	Character(32)	NOT NULL	Key of the user that performed action to be audited
Table_name	Character(32)		Name of the record type affected
Key_ref	Character(32)		Key of affected record
audit_date	Date	NOT NULL	Date/timestamp of action
audit_type	Character(32)	NOT NULL	Type of action, reference to audit type table
Audit_details_memo	Long		Details of action

Audit_details_memo

For AUDIT_DELETE the Audit_details_memo field stays empty. For AUDIT_MODIFY the modified fields are stored. Whether this feature can be implemented and is not causing any performance loss must be decided on implementation.

For AUDIT_ADD the contents of Audit_details_memo are:

Record type	Contents of audit_details_memo field
Itemtype	Do not audit this record type
Item	If not empty: Mark, Subject1, Subject2, Document_date, Company, Mailaddress, City, Text1
Reltype	Do not audit this record type
Itemrel	Sequence
Decos_user	All fields
Userprofile	All fields
Itemprofile	All fields
Itemprofile_user	All fields
Itemright	Do not audit this record type
Itemfield	Field, String_key, Table_key, Used_len, Show_in_list, Show_in_form, Form_sequence, List_sequence, Defaultvalue, Calc_key, Readonly
Command	Define usage as soon Command is used.
Command_user	Define usage as soon Command is used.
Decos_string	Do not audit this record type
Decos_language	Do not audit this record type
String_type	Do not audit this record type
App_status	All fields
User_var	All fields
Global_var	All fields
Decos_audit	Do not audit this record type (itself is the audit record!)
Audit_type	Do not audit this record type
Postcode table	Do not audit this record type

Format of fields in the Audit_details_memo field is:

<fieldname>:<space><contents>CRLF

Example of one field that is modified in Decos_user:

Userprofile_key: 67YTR45WROMNCGHSAJKHS8MNBDRQMJU

Example of Decos_audit:

audit_key	EW87HT8HFWT87HJDK987TQPBV47YT4E3
user_key	YE37TRBHNUQLKJUI87YUTGBREDS39TRB
Table_name	DOCUMENT
Key_ref	UI76YTHSKA891HFSE23GHGG422KJGYLO
audit_date	2001-08-16, 14:05:23
audit_type	AUDIT_DELETE
Audit_details_memo	

2.2.18 Audit_type

Audit_type lists all possible modification types ("Added", "Changed", "Removed") with string_keys for display purposes.

The **Audit_types** can be:

Audit_type	Explanation
AUDIT_ADD	Create or add a record, record link
AUDIT_MODIFY	Modify the contents of a record or recordlink
AUDIT_DELETE	Delete a record, record link
AUDIT_LOGON	Login to Decos Web

Table definition

audit_type	Character(32)	NOT NULL	Auditable action type
string_key	Character(32)	NOT NULL	Reference to name of action type in audit log

Example:

Audit_type AUDIT_ADD
String_key STR_AUDITADD

2.2.19 Decoscache

Decos cache table is used for storing data temporary for JOIN. So when next time logged in, same information is show to user.

Table definition

USER_KEY	Varchar(32)	NOT NULL	User key is stored in the database
Cache_variable	Varchar(32)	NOT NULL	
Cache_date	Datetime		Date of the entry made in table
Cache_depends	Varchar(40)		On which cache is created
Cache_memo	Mediumtext		All data for that session is store in this field

Example:-

USER_KEY	ADMINISTRATOR
Cache_variable	WEBFM_RPTPERSONAL_EN
Cache_date	2007-01-30 18:36:27
Cache_depends	RPTGLOBALBOOK
Cache_memo	All data for that session

2.2.20 Book_field_formula

This table is used for storing data related to book, field, and name of the formulas.

Table definition

Book_Field_Formula_Key	Char(32)	NOT NULL	
Book_key	Char(32)		Name of the folder book on which formula is applicable
Field_key	Char(50)		Key of the field on which formula is applicable
Formula1	Char(32)		Name of the first formula
Formula2	Char(32)		Name of the second formula
Seprator_value	Char(32)		Separator between two formula.

Example:-

Book_Field_Formula_Key	ECD144F25264EF4FA0FC55294BFF31D5
Book_key	32AF72E0067D1C43B44B42BD7B44E6B0
Field_key	TEXT7
Formula1	1Action700
Formula2	
Seprator_value	

2.2.21 Formula

Information related to formula is stored in this table. In this table we have full description of formula type, formula name, book key, formula_expr.

Table definition

Formula_Key	Varchar(32)	NOT NULL	
Formula_Name	Varchar(50)		Name of the formula
Formula_Type	Varchar(50)		Type of book on which formula is applicable
Book_key	Varchar(32)		Key of the book
Formula_Expr	Varchar(50)		Field on which formula generated value is shown
Formula_Increment	decimal(12,2)		Increment value for formula
Couptodate	Char(1)		If a formula is date based then entry is inserted into this field

Example:-

Formula_Key	A5DA879EEFE1F445AA467C3AC2AC0407
Formula_Name	action700
Formula_Type	FOLDER
Book_key	0C8ED03BD3DED641B4EE07039CEBBB1F
Formula_Expr	1_TEXT2
Formula_Increment	0
Couptodate	0

2.2.22 Formula_value

The formula value is table that is used to store actual value and condition on which formula is generated.

Formula_Value_Key	Varchar(32)	NOT NULL	
Formula_Key	Varchar(32)		Key of the formula reference taken from formula table
Valuefor	Varchar(50)		Value on which cur value is generated
Cur_Value	decimal(12,2)		This value is generated on the basis of value for
Date_From	Datetime		
Date_to	Datetime		

Example:-

Formula_Value_Key	AB31E2AF9A3C1A41A64107659401E23B
Formula_Key	2CBD09B16489A74E98475844FEE9F3DF
Valuefor	II
Cur_Value	2
Date_From	
Date_to	

2.2.23 ItemProfilerel

This table is used for storing relation between item profile and templates.

ItemProfilerel_key	character(32)	NOT NULL	
itemprofile_Key	Character(32)		It is key of itemprofile that referred by itemprofile table.

Container_key	Character(32)		Container key like document
Child_key	Character(32)		Key of the template record
Child_type	Character(32)		Type of the template
Reltype_key	Character(32)		Reltype whose relation is stored.

Example:-

ItemProfileRel_key	127E7C302D4CB14EB06202AE21E9FC8C
itemprofile_Key	0797A1EC1D8F1A49B6CDC67FA1C605C7
Container_key	
Child_key	6560D5D6B5284F4DA13D0D24F21EDF32
Child_type	BLOB
Reltype_key	BBLOB

2.2.24 Global_Memo

Global_Memo keeps persistent application settings that are applicable to all users like global. But only difference it store large value in session value field.

Table definition

session_variable	Varchar(50)	NOT NULL	Variable name
session_Memo	Longtext		Variable value

Example:

SESSION_VARIABLE	127E7C302D4CB14EB06202AE21E9FC8C
SESSION_MEMO	(large text value is stored in this field)

2.2.25 Decos_Lock

Decos_Lock enables the system to quickly lock CONTAINER items to generate a guaranteed unique new sequence number. Since this row uses a locking mechanism, it also keeps track of the amount of records that are created. This limits the amount of count actions done on the database, which improves performance.

Table definition

Item_KEY	Varchar(32)	NOT NULL	
Child_type	Varchar(32)	NOT NULL	Type of item on which value is incremented
Max_sequence	Decimal(12,2)		Highest number that is used in item table for that record
RECORDCOUNT	Decimal(12,2)		Total number of record available in data base related to this item
LOCKED_BY	Varchar(32)		The user who generate new value or record
Lock_Timestamp	Varchar(14)		Time for which record is locked
Reltype_Key	Varchar(32)		Relation type

Example:

Item_KEY	26141BF4E521FC4B9811D38F205E3344
Child_type	WORKFLOWLINK
Max_sequence	12
RECORDCOUNT	12
LOCKED_BY	

Lock_Timestamp	
Reltype_Key	

2.2.26 Catchword

Catchword is to allow user to select one or more terms that is used for searching. This table is use to store information related to that catchword.

Table definition

Item_KEY	Varchar(32)	NOT NULL	
Ctw_Path	Varchar(255)		Path of the catchword.(key are shown)
Ctw_Desc	Varchar(255)		Description of the catchword(instead key values are shown)
Ctw_path_Desc	Mediumtext		Description of the catchword path(instead key values are shown)
Ctw_Level	Decimal(12,2)		Sub Level of the catchword
Ctw_Preferreddesc	Varchar(255)		Catchword description whose used as preferred term
Ctw_Obsolete	char(1)		
Ctw_deleted	Datetime		Datetime of catchword deletion
Ctw_scope_note	Varchar(250)		Test which explain scope of the catchword

Example:

Item_KEY	8308DD08ED9D8040BB716D6F981D7EC8
Ctw_Path	17F129A59EA93449891EC409248C58C5 220E93659448A94F8AB39B3B14F743B0 8308DD08ED9D8040BB716D6F981D7EC8
Ctw_Desc	Test1.2
Ctw_path_Desc	Test -> Test1(memo value)
Ctw_Level	2
Ctw_Preferreddesc	Test2.2
Ctw_Obsolete	
Ctw_deleted	2007-02-01 12:30:10
Ctw_scope_note	this word is used to search text value

2.2.27 Catchwordrel

This table is used to store information related to relation between catchwords.

Table:

Catchwordrel_key	character(32)	NOT NULL	
Ctw_key	character(32)		Key of the catchword
Ctw_rel_key	character(32)		Key of the catchword who has relation with the ctw_key
Ctw_preferred	Decimal(12,2)		No of Preferred term

Example:

Catchwordrel_key	0DD87B72D2FD204FA005F53886B1F0AC
Ctw_key	E667B57668216C4C9C9AC6582D41F6D3
Ctw_rel_key	CED0CF11C8EDD34CB1082503749DFCAE
Ctw_preferred	1

2.2.28 Thumbnail

JOIN has a feature of thumbnail, it is has thumbnail view of document and scans same as window's thumbnail. This table is used to store information related to thumbnail.

Table:

BLOB_KEY	Varchar(32)	NOT NULL	Is the reference to item_key of blob item
DOCUMENT_KEY	Varchar(32)		Reference to document item of item table
BLOB_SEQUENCE	Decimal(12,2)		Sequence of the file store for thumbnail
THUMB_TYPE	Decimal(1,0)	NOT NULL	Thumbnail used for list view or form view.
THUM_TIMESTAMP	Varchar(14)		To use latest modified date of actual file
FILE_PATH	Varchar(255)	NOT NULL	Actual path of the file used as thumbnail.
THUMB_DATA	Blob		Item store in a binary format

Example

BLOB_KEY	3BCE45F5804D234882578A88A66EDD19
DOCUMENT_KEY	B283279AD42D344E82E4192E4C8A440D
BLOB_SEQUENCE	1
THUMB_TYPE	2
THUM_TIMESTAMP	20070117132158
FILE_PATH	d:\decos\data\letters\dsdlogo.gif
THUMB_DATA	BLOB ITEM

2.2.29 Postcode table (obsolete)

Table definition

Field	Type and size		Description
PEV	character(12)	PRIMARY KEY	Postcode, Even and Van fields
TOT	character(5)	NOT NULL	NOT NULL
WOONPL	Varchar(24)	NULL	NULL
STRAAT	Varchar(43)	NULL	NULL

Example:

PEV	2201XA100002
TOT	00026
WOONPL	NOORDWIJK ZH
STRAAT	Bonnikleplein

2.3 Unique Record IDs

All automatically generated record IDs (key fields) will be created using the Windows operating system function UuidCreate. This function is guaranteed to generate truly unique results on any Windows machine that contains an ethernet card. UuidCreate returns a 16 byte binary ID that will be stored as 32 uppercase hexadecimal characters in the key field.

Record IDs are stored as strings rather than in binary format to avoid conversion or implementation problems on different database engines. Also this allows storing pre-defined IDs (e.g., string keys, command IDs) in the same fields as automatically generated keys. We only have to ensure that predefined IDs are unique and are NOT represented.

APPENDIX A: ITEMRELS TYPE

Table of which is using which type of connection.

The meaning of a record in insert.sql can be retrieved by checking the STRING_KEY, which can be found in strnl.sql (Dutch) or stren.sql (English).

Itemrel_type	Parent record type	Child record type	String	Relation	Relation	remarks
ACTIONACT	ACTION	ACTIVITY	STR_ACTIONACT	1	MANY	Action with activity
ADDRBRANCH	ADDRESS	ADDRESS	STR_ADDBRANCH	1	MANY	Obsolete
ADDRESSACT	ADDRESS	ACTIVITY	STR_ADDRESSACT	1	MANY	Action with address
ADDRESSARCHIVEWF	ADDRESS	WORKFLOWINSTANCE	STR_WFDOCUMENTWF	1	1	To unlink address item from workflow
ADDRESSDOC1	ADDRESS	DOCUMENT	STR_ADRDOC1	MANY	MANY	Addressee
ADDRESSDOC2	ADDRESS	DOCUMENT	STR_ADRDOC2	MANY	MANY	Copy to
ADDRESSFOLD	ADDRESS	FOLDER	STR_ADRDOC1	MANY	MANY	Address in folder
ADDRESSMAIL1	ADDRESS	TABLECON	STR_ADRMAILCC1	MANY	MANY	Address mailing code
ADDRESSMAIL2	ADDRESS	TABLECON	STR_ADRMAILCC1	MANY	MANY	Address, contact mailing code
ADDRESSWF	ADDRESS	WORKFLOWINSTANCE	STR_WFDOCUMENTWF	1	1	Address in workflow
ADRCON1	ADDRESS	CONTACT	STR_ADRCON1	1	MANY	Contact person with address
ADREXTRADRTYPE	ADDRESS	ADDRESSTYPE	STR_ADDRESS_ADDRESSTYPE	MANY	1	Extra address store in a contact details
BACTION	CONTAINER	ACTION	STR_BACTION	1	MANY	Action table
BACTIVITY	CONTAINER	ACTIVITY	STR_BACTIVITY	1	MANY	Pending item containers(active, handled, inactive)
BADDRESS	CONTAINER	ADDRESS	STR_ADDRESS	MANY	MANY	Address file
BADMINLEVEL	CONTAINER	ADMINILEVEL	STR_ADMINILEVEL	1	MANY	Level of admin(super admin, standard admin)
BADMINLEVELRIGHT	ADMINILEVEL	ADMINLEVELRIGHT	STYR_USERADMINILEVEL	1	MANY	Admin level with admin right

BBLOB	CONTAINER	BLOB	STRBBLOB	1	MANY	Blob table
BCONTACT	CONATINER	CONTACT	STR_BCONTACT	MANY	MANY	Contact file
BCRITERIA	CONTAINER	CRITERIA	STR_M_CRITERIA	1	MANY	
BDOCUMENT	CONTAINER	DOCUMENT	STR_BDOCUMENT	1	MANY	Document file
BFOLDER	CONTAINER	FOLDER	STR_BFOLDER	MANY	MANY	Folder file
BLETTERTEMPLATE	CONTAINER	LETTERTEMPLATE	STR_BLETTERTEMPLATE	1	MANY	Unknown
BMGMTQRY	REPORT	MANAGEMENTQUERY	STR_MGMTQRY	1	MANY	Unknown
BOOKDEFAULTWORKFLOW	CONTAINER	DEFAULTWORKFLOWSETTING	STR_DERWFSETTING	1	1	Folder link with default workflow
BREPORT	CONTAINER	REPORT	STR_M_REPROT	1	MANY	Container for reports
BSTDROUTE	CONTAINER	STDROUTE	STR_BSTDROUTE	1	MANY	Relation between standard route and activity
BSTD TABS	CONTAINER	STDROUTE	STR_BSTDROUTE	1	MANY	Folder in standard tab
BTABLECON	CONTAINER	TABLECON	STR_BTABLECON	1	MANY	Table
CONTACTDEFAULTADDRESS	CONTACT	ADDRESSTYPE	STR_ADDRESS_ADDRESSTYPE	MANY	1	Unknown
CONTACTDOC1	CONTACT	DOCUMENT	STR_CONTACTDOC1	MANY	MANY	Contact with a document
CONTACTDOC2	COTNACT	DOCUMENT	STR_CONTACTDOC2	MANY	MAN Y	Contact with a document in case of CRM
CONTACTDSP2	CONTACT	DSPITEM	STR_CONTACTDSP2	MAN Y	MANY	
CONTACTFOLD	CONTACT	FOLDER	STR_CONTACTDOC1	MANY	MANY	Contact in folder
CONTACTFOLD2	CONTACT	FOLDER	STRCONTACTFOLD2	MANY	MANY	Contact in folder in case of CRM
CONTACTMAIL1	CONTACT	TABLECON	STR_CNIMAILCC3	MANY	MANY	Contact mailing code
CONTAINERADDRESSTYPE	CONTAINER	ADDRESSTYPE	STR_ADDRESS_ADDRESSTYPE	1	MANY	Unknown
CONTAINERDEFAULTANSWER	CONTAINER	CONTAINER	STR_DEFAULTANSWERBOOK	1	MANY	Default answer book assign to document
CONTAINERDEFAULTCONTACT	CONTAINER	CONTAINER	STR_CONTAINERDEFAULTCONTACT	1	1	Default contact book assign to address book
CONTEXTRADR	CONTACT	ADDRESS	STR_CONTACT_EXTRADDRESS	1	MANY	Contact and Extra address relation
DECISIONTBLVALUE	DECISION	TABLECON	STR_WFDECISIONTBLVALUE	1	1	UNKNOWN
DISTALGOCONTAINER	CONTAINER	DISTRIBUTIONALALGORITHM	STR_WFCONTAINER	1	MANY	UNKNOWN
DOCCATCHWORD	DOCUMENT	CATCHWORD	STR_CATCHWORD	MANY	MANY	Catchword link to document
DOCOPY	DOCUMENT	DOCUMENT	STR_DOCCOPY	1	MANY	Copy of document

DOCMail1	DOCUMENT	TABLECON	STR_CNTMAILCC3	MANY	MANY	Mail of document(when we send attach document as mail that time new entity is generated which has this relation type)
DOCUMENTACT	DOCUMENT	ACTIVITY	STR_DOCACT	1	MANY	Activity with document
DCOUMENTARCHIVEWF	DOCUMENT	WORKFLOWINSTANCE	STR_WFDOCUMENTWF	1	1	To unlink document item from workflow
DOCUMENTMAIL	DOCUMENT	BLOB	STR_DOCMAIL	1	MANY	UNKNOWN
DOCUMENTFIFF	DOCUMENT	BLOB	STR_DOCTIFFBLOB	1	MANY	Blob attached to document
DOCUMENTWF	DOCUMENT	WORKFLOWINSTANCE	STR_WFDOCUMENTWF	1	1	Document related to workflow instance
FILEOFFILE	CONTAINER	CONTAINER	STR_FILEOFFILE	1	MANY	Container of container, e.g. all tables, all document files
FOLDERCATCHWORD	FOLDER	CATCHWORD	STR_CATCHWORD	MANY	MANY	Catchword link to folder
FOLDERACT	FOLDER	ACTIVITY	STR_FOLDERACT	1	MANY	Activity with folder
FOLDERARCHIVEWF	FOLDER	WORKFLOWINSTANCE	STR_WFDOCUMENTWF	1	1	To unlink folder item from workflow
FOLDERDOC1	FOLDER	DOCUMENT	STR_FOLDERDOC1	MANY	MANY	Folder with document
FOLDERFOLD	FOLDER	FOLDER	STR_FOLD1FOLD1	1	MANY	Folder linked to a folder
FOLDER FOLDEREQU	FOLDER	FOLDER	STR_FOLDER FOLDEREQU	MANY	MANY	Folder having equal relation
FOLDERWF	FOLDER	WORKFLOWINSTANCE	STR_DOCUMENTWF	1	1	Folder link with workflow instance
LETTERTEMPLATEBLOB	LETTERTEMPLATE	BLOB	STR_LETTERTEMPLATEBLOB	1	1	Blob attached to template
LETTERTEMPLATETF	LETTERTEMPLATE	LETTERTEMPLATEFIELD	STR_LETTERTEMPLATE	1	MANY	Not clear
LETTERTEMPLATETP	LETTERTEMPLATE	LETTERTEMPLATEPROFILE	STR_LETTERTEMPLATETP	1	MANY	Not clear
LETTERTEMPLATETS	LETTERTEMPLATE	LETTERTEMPLATESELECTION	STR_LETTERTEMPLATELTS	1	MANY	Not clear
ROLECONTAINER	CONTAINER	USRROLE	STR_WFCONTAINER	1	MANY	Roles of a users
SCANMARKING	SCAN	MARKING	STR_SCANMARKING	1	MANY	To draw mark in scan
SCANPOSTIT	SCAN	STICKNOTE	STR_SCANSTICKNOTE	1	MANY	Sticky note shown in scan
STDROUTEACT	STDROUTE	ACTIVITY	STR_STDROUTEACT	1	MANY	Activity under standard route
STDATABTAB	STDATABS	TABS	STR_STDATABSSTAB	1	MANY	tabs in standard tab
USERACT	USRINFO	ACITIVITY	STR_USERACT	1	MANY	User that is responsible for the activity
USERADMINILEVEL	ADMINILEVEL	USRINFO	STR_USERADMINLEVEL	1	MANY	
USERAGENDA	USRINFO	USRAGENDA	STR_WFUSRAGENDA	1	MANY	

USERFASTMENUBLOCK	USRINFO	FASTMENUBLOCK	STR_DEFWFSETTING	1	MANY	Item show in fast menu as per user
USERROLE	USRROLE	USRINFO	STR_WFUSERROLE	MANY	MNAY	User role with user
USRDATA	USRINFO	CONTAINER	STR_USERDATA	1	MANY	Obsolete
USRINFO	CONTAINER	USRINFO	STR_USERINFO	1	MANY	All user profiles
USRROLETABLE	USRROLE	CONTAINER	STR_WFUSRROLETABLE	1	1	User table
WFCONTAINER	CONTAINER	WORKFLOW	STR_WFCONTAINER	1	MANY	Workflow in document /folder
WFDECISIONLINK	DECISION	WORKFLOWLINK	STR_WFDECISIONLINK	MANY	1	Decision with workflow link
WFFAVOURITE	USRINFO	WORKFLOWINSTANCE	STR_WFFAVOURITE	1	MANY	
WFHANDLEDITEM	CONTAINER	WORKFLOWLINKINSTANCE	STR_WFPENDINGITEM	1	MANY	Handle item in workflow
WFINACTIVEPENDIGNITEM	CONTAINER	WORKFLOWINSTANCE	STR_WFPENDINGITEM	1	MANY	Active activity in a workflow
WFINSTANCELINK	WORKFLOWINSTAN CE	WORKFLOWLINKINSTANCE	STR_WFINSTANCELINK	1	MANY	Instance of workflow link instance
WFINSTANCELINKACTION	WORKFLOWLINKIN STANCE	ACTION	STR_WFINSTANCECLELINKACTION	MANY	1	Workflow link instance for a action
WFINSTANCELINKHANDLER	WORKFLOWLINKIN STANCE	USRINFO	STR_WFINSTANCECLELINKHANDL ER	MANY	1	Handler of workflow link instance
WFLINK	WORKFLOW	WORKFLOW	STR_WFLINK	1	MANY	Relation between to workflow
WFLINKDISTALGO	WORKFLOWLINK	DISTRIBUTIONALGORITHM	STR_WFCONTAINER	MANY	1	Workflow and algorithm used for distributing work
WFLINKHANDLINGROLE	WORKFLOWLINK	USRROLE	STR_WFLINKHANDLINGROLE	MANY	1	Workflow related with user role
WFLINKHANDLINUSR	WORKFLOWLINK	USRINFO	STR_WFLINKHANDLINGUSR	MANY	1	Workflow instance related with user
WFLINKORGINAL	WORKFLOWLINK	WORKFLOWLINKINSTANCE	STR_WFLINKORIGINAL	1	MANY	Workflow instance of workflow
WFNEXTLINK	WORKFLOWLINK	WORKFLOWLINK	STR_WFNEXTLINK	MANY	MANY	Workflow link linked with workflow link
WFORIGINAL	WORKFLOW	WORKFLOWINSTANCE	STR_WFORIGINAL	1	MANY	Workflow having workflow instance
WFORIGINALVERSION	WORKFLOW	WORKFLOW	STR_WFORIGNALVERSION	1	MANY	To maintain copy of workflow if any change in a flow of workflow.

WFOUTCOMETABLE	WORKFLOWLINK	CONTAINER	STR_WFOUTCOMETABLE	MANY	1	Workflow file
WFPENDINGITEM	CONTAINER	WORKFLOWLINKINSTANCE	STR_WFPENDINGITEM	1	MANY	Pending activity in workflow
WFREPORT	WORKFLOWREPORT	WORKFLOW	STR_WFREPORT	1	MANY	
DSPITEM	DSPITEM	DSPITEM	STR_DSPDSP	1	MANY	DSPITEM LINK TO DSPLINKITEM
DSPADDRESS	DSPITEM	ADDRESS	STR_DSPADD	MANY	MANY	Dspitem link to address
DSPFOLDER	DSPITEM	FOLDER	STR_DSPFOLD	MANY	MANY	Dspitem link to folder
DSPDOCUMENT	DSPITEM	DOCUMENT	STR_DSPDOC	MANY	MANY	Dspitem link to document
DSPFILE	DSPITEM	BLOB	STR_DSPFILE	1	MANY	Blob attached to dspitem
CONATACTDSP2	DSPITEM	CONTACT		MANY	MANY	Contact link to dspitem
BDSPITEM	DSPITEM	CONTAINER	STR_BDSPITEM	MANY	1	Dspitem file
BCATALOG	CONTAINER	LBCATALOG	STR_BCATALOG	1	MANY	Library files
ADDRESSLBCAT	ADDRESS	LBCATALOG	STR_ADDRESSLBCAT	MANY	MANY	Address link to library book
CONTACTLBCAT	CONTACT	LBCATALOG	STR_CONTACTLBCAT	MANY	MANY	Contact link to library book
FOLDERLBCA	FOLDER	LBCATALOG	STR_FOLDERLBCAT	MANY	MANY	Folder link to library book
LBCATLOGDOC	DOCUMENT	LBCATALOG	STR_LBCATLOGDOC	MANY	MANY	Document linked to Library book
LBSUBSCRIPTION	LBCATALOG	LBCATALOG		MANY	MANY	
CONTACTMAIL2	CONTACT	TABLECON	STR_CNTMAILCC3	MANY	MANY	Single Contact mailing code