Database Reporting Guide

Table of Contents

Introduction

This guide is intended to provide detailed information on the critical tables within the OnBase database schema that are frequently utilized for reporting purposes. The content is intended for experienced SQL database administrators or programmers only.

Use caution when writing external queries to ensure they do not adversely affect OnBase system performance. Custom reports should be tested and baselines should be captured in order to measure performance trends over time.

The information in this guide is current as of the current version of OnBase as of September 8, 2014 in addition to the current versions of SQL Server and Oracle as of said date. Portions of the document may not apply to versions of the software prior to this date, but all information should remain relevant for future versions of the software and schema. The information contained in this guide is subject to change without notice and does not represent a commitment on the part of Hyland Software, Inc. Please contact your first line of support to request any updates to this guide from the Hyland Software Database Support Team.

The Report Services Module

This guide was written for the purpose of explaining the OnBase schema to aide in the writing of custom reports. Custom reports can be developed and implemented using a wide variety of reporting utilities, including OnBase Report Services.

As of OnBase 10.0 and later, an important change was made in the way Report Services connects to and executes reports against SQL Server databases. Report Services connects to the database using the Viewer account, and with this change it now sets an isolation level of READ UNCOMMITTED for this connection by default. This allows Report Services to possibly avoid blocking scenarios and gives it access to as much data as possible when querying the database. This enhancement does not affect product functionality or use.

64-bit vs. 32-bit Schemas

OnBase installations that were created (i.e., database schema creation) prior to OnBase 13 have a 32-bit schema, meaning that in SQL Server installations, the primary data type for integers and enumeration was the **integer** data type, which has a native limit of 2.14 billion unique positive values. The data types mentioned in this guide are based on the 32-bit schema design.

New installations, starting with OnBase 13 have a 64-bit schema, meaning that in SQL Server installations, the primary data type for integers and enumeration is the **bigint** data type. OnBase 13 and higher is capable of storing and retrieving up to 2,147,483,647 unique positive values. If your installation was created on version 13 or higher, please consider that the data types listed in this guide as **int** will be **bigint** in your database. Additionally, OnBase databases using the 64-bit schema have a **filepath** column (HSI.itemdatapage, etc.) with a size of **char(80)** instead of **char(26)**.

(NOTE: For Oracle implementations, the primary data type for integers and enumeration is the **numeric** data type, which was not changed between the 32-bit and 64-bit schema. The **filepath** difference mentioned above is the only data type difference between a 32-bit and 64-bit Oracle schema.)

Standard Database Tables

ITEMDATA

The **ITEMDATA** table is the core of an OnBase system. Every document in the system has one row in the **ITEMDATA** table. The primary key on this table is **itemnum**, which is the unique identifier for every document in the system. The **itemnum** of a document is linked to many other pieces of information about that document. The **ITEMDATA** table is located in DBSpace2.

Column Name	Data Type	Description
itemnum	int	The Document Handle, which is the unique identifier for a document in OnBase.
itemname	char(255)	The Auto-Name string of a document.
batchnum	int	The process batch to which the document belongs.
status	int	The current status of the document in the system. The possible values are: 0 – Indexed (available for retrieval) 1 – Awaiting Index 16 – Deleted
itemtypegroupnum	int	The Document Type Group to which the document belongs.
itemtypenum	int	The Document Type of the document.
itrevnum	int	
itemdate	datetime	The Document Date of the document.
datestored	datetime	The date the document was added to the system.
usernum	int	
deleteusernum	int	
securityvalue	int	
doctracenumber	char(20)	
institution	int	
maxdocrev	int	

The indexes for the **ITEMDATA** table are located in DBSpace2i.

Index Name	Included Columns
itemdata9	itemnum, itemdate (desc), itemtypenum, status
itemdata10	Itemtypenum, itemdate (desc), status
itemdata13	batchnum, itemnum, status

ITEMDATAPAGE

The **ITEMDATAPAGE** table holds records for every document page stored in the OnBase Disk Groups. For example, if a 50-page document is scanned into the system, there are 50 **ITEMDATAPAGE** records in the table. If a 50-page COLD document is processed into the system, however, there is only one **ITEMDATAPAGE** record. A COLD process may process many files, but the pages for any one COLD document only span one file no matter how many pages there are for the document. When scanning in OnBase, each page is stored as a separate file (single-page TIFF). There is at least one **ITEMDATAPAGE** record for every **ITEMDATA** record.

Note: Virtual Electronic Forms have an **ITEMDATAPAGE** record, but the **filepath** is blank.

The **ITEMDATAPAGE** table is located in DBSpace9.

Column Name	Data Type	Description
filetypenum	Int	The file type of the current document (cross-referenced to the **FILETYPE** table).
docrevnum	Int	The number of revisions to the document (zero-based).
itempagenum	Int	The page number of the current document.
itemnum	Int	Links **itemdatapage** to the **ITEMDATA** table.
batchnum	Int	
diskgroupnum	Int	The ID of the Disk Group in which the file exists.
logicalplatternum	Int	The number of the volume in which the file exists.
filepath	char(26)	The file path to the document.
filesize	Int	
compressfile	Int	
numbernotes	Int	
numberpages	Int	
physicalpagenum	Int	
numberlines	Int	
offset	Int	
deleteusernum	Int	
imagetype	Int	
imageoffsettype	Int	
numexceptions	Int	
xdpi	Int	
ydpi	Int	

The indexes for the **ITEMDATAPAGE** table are located in DBSpace9i.

Index Name	Included Columns
itemdatapage1	itemnum, itempagenum, docrevnum, filetypenum
itemdatapage2	batchnum
itemdatapage4	diskgroupnum, logicalplatternum, filepath

DOCTYPE

The **DOCTYPE** table stores a record for every configured OnBase Document Type. This parent table can be joined with the **ITEMDATA** table to get the name of the Document Type (**itemtypename**) for a particular document. The **DOCTYPE** table is located in DBSpace8.

Column Name	Data Type	Description
itemtypenum	int	The unique identifier of the Document Type.
itemtypename	char(66)	The name of the Document Type.
itrevnum	int	
itemtypegroupnum	int	
filetypenum	int	
compressfile	int	
autonamestring	char(150)	
inuse	int	
diskgroupnum	int	
displaythumbs	int	
numrows	int	
isdocrevisionable	int	
docsourceflag	int	
imagewindowflags	int	
uiflags	int	
itemtypeflags	int	
revisablebyinst	int	

The indexes for the **DOCTYPE** table are located in DBSpace8.

Index Name	Included Columns
doctype1	itemtypenum
doctype2	itemtypename

ITEMTYPEGROUP

The **ITEMTYPEGROUP** table stores a record for every configured OnBase Document Type Group. This table can be joined with the **ITEMDATA** table to get the name of the Document Type Group (**itemtypegroupname**) for a particular document. The **ITEMTYPEGROUP** table is located in DBSpace8.

Column Name	Data Type	Description
itemtypegroupnum	int	The unique identifier of the Document Type Group.
itemtypegroupname	char(66)	The name of the Document Type Group.
inuse	int	
itemtypegroupused	int	
numrows	int	
docsourceflag	int	
dmasourcename	char(100)	
dmaconnectflag	int	
dmausername	char(30)	
dmauserpassword	char(30)	
dmasystemname	char(128)	
diskgroupnum	int	
flags	int	

The index for the **ITEMTYPEGROUP** table is located in DBSpace8.

Index Name	Included Columns
itemtypegroup1	itemtypegroupnum

DISKGROUP

The **DISKGROUP** table stores a record for every configured OnBase Disk Group. The table can be joined with the **ITEMDATAPAGE** table to get the name (**diskgroupname**) of the Disk Group for a particular document. The **ITEMDATAPAGE** table is located in DBSpace10.

Column Name	Data Type	Description
diskgroupnum	int	The unique identifier of the Disk Group.
diskgroupname	char(21)	The name of the Disk Group.
currentdirectory	int	
diskgrouptype	int	
diskthreshold	int	
filesindirectory	int	
filesperdirectory	int	
lifespan	int	
numberofbackups	int	
ucautopromotespace	int	
autopromotespace	int	
lastlogicalplatter	int	The current volume for the Disk Group.
cachepath	char(70)	
lpnumsyscache	int	
cachelpnum	int	
lpcachethreshold	int	
formatnum	int	
committedlp	int	
numberofexports	int	
prepathed	int	
exportmgrnum	int	
adminusernum	int	
retentionyears	int	
flags	int	

The index for the **DISKGROUP** table is located in DBSpace10.

Index Name	Included Columns
diskgroup1	diskgroupnum

PHYSICALPLATTER

The **PHYSICALPLATTER** table stores a record for every volume and copy in a Disk Group and allows the user to get the first portion of a document's full file path (**lastuseddrive**). The information in this table, combined with **filepath** in the **ITEMDATAPAGE** table, can provide the complete file path for a document. The **PHYSICALPLATTER** table is located in DBSpace10.

Column Name	Data Type	Description
physicalplatternum	int	The Disk Group copy.
logicalplatternum	int	The Disk Group volume, which links **physicalplatter** to the **itemdatapage** table.
diskgroupnum	int	Links **physicalplatter** to the **diskgroup** and **itemdatapage** tables.
plattertype	int	
diskidalias	char(30)	
diskidfilename	char(60)	
diskidflag	int	
diskidsize	int	
lastuseddrive	char(255)	The first portion of a path to a file in an OnBase Disk Group.
spacefree	int	
spaceused	int	
disksearchorder	int	
blocksize	int	
maxcacheplatters	int	
platterdeleted	int	
onbackupqueue	int	
maxlogicalplatter	int	
minlogicalplatter	int	
dbnum	int	
plattercreated	int	
ondeletequeue	int	
plattertype2	int	

The indexes for the **PHYSICALPLATTER** table are located in DBSpace10.

Index Name	Included Columns
physicalplatter1	diskgroupnum, logicalplatternum, physicalplatternum
physicalplatter2	diskgroupnum, physicalplatternum

Index Name	Included Columns
physicalplatter3	onbackupqueue
physicalplatter4	ondeletequeue

USERACCOUNT

The **USERACCOUNT** table contains a record for every OnBase user and can be used to obtain the user name for a transaction logged in any of the logging tables (e.g., **TRANSACTIONXLOG**, **SCANNINGLOG**). The **USERACCOUNT** table is located in DBSpace8.

Column Name	Data Type	Description
usernum	int	The unique identifier for each OnBase user.
username	char(75)	The user's name (text).
defaultdate	int	
disablelogin	int	
institution	int	
mainframeupdate	int	
networkid	char(13)	
obrefresh	int	
usercode	char(5)	
userpassword	char(20)	
userpref1	int	
userpref2	int	
autodisplaywin	int	
helpwindowtype	int	
helpwindowloc	int	
badlogincount	int	
encryptedpassword	char(40)	
lastlogon	datetime	
lastpwchange	datetime	
defprintformatnum	int	
realname	char(40)	
licenseflag	int	
longusercode	char(20)	
longpassword	char(20)	
primaryusergroup	int	
userpref3	int	
defprocessdate	int	
qapercent	int	
emailaddress	char(255)	
phonenumber	char(32)	
cellnumber	char(15)	

Column Name	Data Type	Description
lockouttime	datetime	
lockoutreason	int	
usertype	int	
pinhash	char(40)	
pinlastchanged	datetime	
pinlastentered	datetime	
badpincount	int	

The indexes for the **USERACCOUNT** table are located in DBSpace8.

Index Name	Included Columns
useraccount1	username
useraccount2	usernum

USERGROUP

The **USERGROUP** table stores a record for every configured OnBase user group. The **USERGROUP** table is located in DBSpace8.

Column Name	Data Type	Description
usergroupnum	int	The unique identifier for each OnBase user group.
usergroupname	char(128)	User group name.
grouptoemulate	int	
mfaccessflag	int	
userprivilege0	int	
userprivilege1	int	
userprivilege2	int	
userprivilege3	int	
timeout	int	
userprivilege4	int	
passwordflags	int	
passwordexpires	int	
passworddllpath	char(255)	
passwordhistdays	int	
configrights	int	
timeouttype	int	
logviewprivs	int	
logdeleteprivs	int	
configrights2	int	
userprivilege5	int	
userprivilege6	int	
userprivilege7	int	
pswdpolicynum	int	
pinpolicynum	int	

The index for the **USERGROUP** table is located in DBSpace8.

Index Name	Included Columns
usergroup1	usergroupnum

USERXUSERGROUP

The **USERXUSERGROUP** table stores the relationship between an OnBase user and the user groups to which that user belongs. A user can belong to multiple user groups, and any user group can have multiple users. The **USERXUSERGROUP** table is located in DBSpace8.

Column Name	Data Type	Description
usernum	int	The unique identifier for each OnBase user (cross-referenced to **USERACCOUNT**).
usergroupnum	int	The unique identifier for each OnBase user group (cross-referenced to **USERGROUP**).
cfgrightdefault	int	

The index for the **USERXUSERGROUP** table is located in DBSpace8.

Index Name	Included Columns
userxusergroup1	usernum, usergroupnum

REGISTEREDUSERS

The **REGISTEREDUSERS** table lists every workstation that has logged on to OnBase via the OnBase Client (not the Web Client). The **registernum** is also stored in a registry key on each workstation. The **REGISTEREDUSERS** table is located in DBSpace8.

Column Name	Data Type	Description
registernum	int	The unique identifier for each OnBase Client workstation.
registername	char(80)	The workstation name.
dateregistered	datetime	
networkid	char(13)	
pcserialnum	int	
usernum	int	
stationdesc	char(100)	
wkstationgrpnum	int	
cachenum	int	
macaddress	char(12)	
lastlogon	datetime	
badlogincount	int	
machineid	char(50)	
platformtype	int	

The indexes for the **REGISTEREDUSERS** table are located in DBSpace1.

Index Name	Included Columns
registeredusers1	registernum
registeredusers2	pcserialnum

LOGGEDUSER

The **LOGGEDUSER** table lists all users logged on to the OnBase system at the current time. The **LOGGEDUSER** table is located in DBSpace1.

Column Name	Data Type	Description
usernum	int	The unique identifier for each OnBase user (cross-referenced to **USERACCOUNT**).
producttype	int	The OnBase module used to access the system.
registernum	int	The unique identifier for each Client workstation (cross-referenced to **REGISTEREDUSERS**).
checkin	int	
heartbeat	int	
numlocks	int	
sessionid	int	The unique identifier for a session.
sessionguid	char(32)	
terminalsessionid	int	
flags	int	

The indexes for the **LOGGEDUSER** table are located in DBSpace10.

Index Name	Included Columns
loggeduser1	usernum, registernum, producttype
loggeduser2	registernum

Keyword Tables

KEYTYPETABLE

The **KEYTYPETABLE** has one entry for every Keyword that exists in the system. This table is not typically one of the main, driving tables in a query, but rather one that can be joined against to obtain Keyword Type information for a document or Document Type. The **KEYTYPETABLE** table is located in DBSpace8.

Column Name	Data Type	Description
keytypenum	int	The unique identifier for each Keyword Type.
keytype	char(51)	The unique name for each Keyword Type.
keytypemask	char(51)	
keytypeflags	int	
datatype	int	The data type of the Keyword Type (e.g., char, datetime).
keytypelen	int	Length of an Alphanumeric Keyword Type (will be 0 for non-Alphanumeric data types).
compositeflag	int	
badprimarykey	int	
numrows	int	
numrefs	int	
disttype	int	
staticstring	char(151)	
dropdownset	int	
keywordsetnum	int	
lockkeys	int	
keytypeflags2	int	
columnwidth	int	
dateformat	int	
dateseparator	int	
currencyformatnum	int	The currency format applied to any currency Keyword Type (cross-referenced to **CURRENCYFORMAT** table).
mulkeysetablenum	int	
securitymask	char(51)	

The indexes for the **KEYTYPETABLE** table are located in DBSpace8.

Index Name	Included Columns

Index Name	Included Columns
keytypetable1	keytypenum, keytype
keytypetable2	keytype

Individual Keyword Types

The following section details the OnBase Keyword Type table structure and naming conventions, which are based on the data type of the Keyword Type.

Note: If a document does not contain an entry for a particular Keyword value, there is no record in the associated Keyword table.

For all but one Keyword Type, a single table is created in which the Keyword values are stored. This table is named **KEYITEM###**, where **###** is the Keyword Type ID that can be found by querying the **KEYTYPETABLE**. The **KEYITEM###** table is cross-referenced between the OnBase document and its Keyword Type value. Any document can have more than one record with the same **itemnum** in this table.

For the Dual Table Alphanumeric Keyword Type, there are two tables created in which Keyword information is stored: **KEYTABLE###** and **KEYXITEM###**, where **###** is the Keyword Type ID that can be found by querying the **KEYTYPETABLE**.

There is only one occurrence of a specific alphanumeric string in **KEYTABLE###**. The **keywordnum** is used to associate the alphanumeric string with the **KEYXITEM###** table.

The **KEYXITEM###** table is cross-referenced between the OnBase document and its Keyword Type value (**keyvaluechar** in **KEYTABLE###**). Any document can have more than one record with the same **itemnum** in this table.

Keyword Type (OnBase Configuration)	**Table Design**	**datatype in KEYTYPETABLE**
Numeric (Up to 9 Digits)	**Table Name: KEYITEM###** **Location: DBSpace6**	6

Column	Data Type	Description
itemnum	int	The Document Handle.
keyvaluesmall	int	The 9-digit (maximum) numeric Keyword value for the document.
keysetnum	int	

Keyword Type (OnBase Configuration)	**Table Design**	**datatype in KEYTYPETABLE**
Numeric (Up to 20 Digits)	**Table Name: KEYITEM###** **Location: DBSpace6**	1

Column	Data Type	Description
itemnum	int	The Document Handle.
keyvaluebig	decimal (20,0)	The 20-digit (maximum) numeric Keyword value for the document.
keysetnum	int	

Keyword Type (OnBase Configuration)	**Table Design**	**datatype in KEYTYPETABLE**
Date	**Table Name: KEYITEM###** **Location: DBSpace6**	4

Column	Data Type	Description
itemnum	int	The Document Handle.
keyvaluedate	datetime	The date Keyword value for the document.
keysetnum	int	

Keyword Type (OnBase Configuration)	Table Design	datatype in KEYTYPETABLE
Date & Time	**Table Name: KEYITEM###** **Location: DBSpace6**	9

Column	Data Type	Description
itemnum	int	The Document Handle.
keyvaluetod	datetime	The date/time Keyword value for the document.
keysetnum	int	

Keyword Type (OnBase Configuration)	Table Design	datatype in KEYTYPETABLE
Currency	**Table Name: KEYITEM###** **Location: DBSpace6**	3

Column	Data Type	Description
itemnum	int	The Document Handle.
keyvaluecurr	decimal(20,2)	The currency Keyword value for the document.
keysetnum	int	

Keyword Type (OnBase Configuration)	Table Design	datatype in KEYTYPETABLE
Specific Currency	**Table Name: KEYITEM###** **Location: DBSpace6**	11

Column	Data Type	Description
itemnum	int	The Document Handle.
keyvaluecurr	decimal(20,2)	The currency Keyword value for the document.
keysetnum	int	
currencyformatnum	int	The currency format for the specific Keyword value cross-referenced to the **CURRENCYFORMAT** table.

Keyword Type (OnBase Configuration)	Table Design	datatype in KEYTYPETABLE
Floating Point	**Table Name: KEYITEM###** **Location: DBSpace6** **Column** | **Data Type** | **Description** itemnum | int | The Document Handle. keyvaluefloat | float | The floating point Keyword value for the document. keysetnum | int |	5
Alphanumeric Single Table	**Table Name: KEYITEM###** **Location: DBSpace6** **Column** | **Data Type** | **Description** itemnum | int | The Document Handle. keyvaluechar | char(*length*) | The alphanumeric Keyword value for the document, where *length* is the length of the Keyword, up to 251. keysetnum | int |	10

Keyword Type (OnBase Configuration)	Table Design	datatype in KEYTYPETABLE
Alphanumeric Dual Table	**Table Name: KEYTABLE###** **Location: DBSpace6** **Column** | **Data Type** | **Description** keywordnum | int | The unique identifier for the alphanumeric string. keyvaluechar | char(*length*) | The alphanumeric Keyword value for the document, where *length* is the length of the Keyword, up to 251. **Table Name: KEYXITEM###** **Location: DBSpace3** **Column** | **Data Type** | **Description** itemnum | int | The Document Handle. keywordnum | int | The unique identifier of the **KEYTABLE###** record associated with the document. keysetnum | int |	2

The indexes for the **KEYITEM###** tables are located in DBSpace6i.

Index Name	**Included Columns**
keyitem###1	itemnum, keyvalue*<datatype>*
keyitem###2	keyvalue*<datatype>*,itemnum, currencyformatnum†

†Column only exists for Specific Currency Keyword Type index.

The indexes for the **KEYTABLE###** tables are located in DBSpace6i.

Index Name	**Included Columns**
keytable###1	Keywordnum
keytable###2	keyvaluechar, keywordnum

The indexes for the **KEYXITEM###** tables are located in DBSpace3i.

Index Name	**Included Columns**
keyxitem###1	itemnum, keywordnum
keyxitem###2	keywordnum, itemnum

Mixed Case Keywords

As of OnBase version 6.2, Single Table and Dual Table Alphanumeric Keywords can store alphanumeric values in mixed case rather than all uppercase. This is accomplished by adding an additional column to the **KEYITEM###** and **KEYTABLE###** tables named **keyvaluecharcs**. The text value is still stored in uppercase in the **keyvaluechar** column for searching purposes.

For a Single Table Alphanumeric keyword, the **datatype=13** in **KEYTYPETABLE**.

For a Dual Table Alphanumeric keyword, the **datatype=12** in **KEYTYPETABLE**.

The following section details the Autofill Keyword Set, Keyword Type Group, and Multi-Instance Keyword Type Group table structure and naming conventions.

Note: If a document does not contain an entry for a particular Keyword value, there is no record in the associated Keyword Type Group table.

KEYWORDSET

The **KEYWORDSET** table has one entry for every Autofill Keyword Set, Keyword Type Group, or Multi-Instance Keyword Type Group in the OnBase system. Like the **KEYTYPETABLE**, this table is not typically one of the main, driving tables in a query, but rather one that can be joined against to obtain additional information. The **KEYWORDSET** table is located in DBSpace8.

Column Name	Data Type	Description
keysettablenum	int	The unique identifier for the Keyword Set or Group.
keysetname	char(80)	
tablecreated	int	
updatekeyset	int	
updatekeysetdocs	int	
removeunusedkeyset	int	
iskeytypegroup	int	0 = Autofill Keyword Set, 1 = Keyword Type Group, 2 = Multi-Instance Keyword Type Group.
flags	int	
autonamestring	char(150)	
vbscriptnum	int	The ID of the VB Script used by an External Autofill Keyword Set.
selectstring	char(16)	The SQL string used to retrieve Keyword values by an External Autofill Keyword Set.
connectstring	char(255)	The ODBC source, user name, and password used by an External Autofill Keyword Set.

The index for the **KEYWORDSET** table is located in DBSpace8.

Index Name	Included Columns
keywordset1	Keysettablenum

KEYSETDATA***

Each Autofill Keyword Set in the system has a **KEYSETDATA*** ** table, where ******* is the unique identifier of the Keyword Set (**keysettablenum** in **KEYWORDSET**). There can be multiple Keyword Types assigned to an Autofill Keyword Set. Each Keyword Type has its own **ks###** column in the table. All **ks###** columns are a char data type, regardless of the data type specified for the Keyword Type in the **KEYTYPETABLE** table. The **KEYSETDATA*** ** table is located in DBSpace6.

Column Name	Data Type	Description
keysetnum	int	The unique identifier for each row in the table.
ks###	char(*length*)	The Keyword Type assigned to the Autofill Keyword Set, where **###** is the unique identifier for the Keyword Type (cross-referenced to **KEYTYPETABLE**).
useagecount	int	

The indexes for the **KEYSETDATA*** ** table are located in DBSpace6i.

Index Name	Included Columns
Akeysetdata* **	ks### (the primary Keyword Type in the set)
keysetdata*_2**	Keysetnum

KEYGROUPDATA***

Each Keyword Type Group in the system has a **KEYGROUPDATA*** ** table, where ***** ** is the unique identifier of the Keyword Type Group (**keysettablenum** in **KEYWORDSET**). There can be multiple Keyword Types assigned to a Keyword Type Group. Each Keyword Type will have its own **kg###** column in the table. If the Keyword Type Group is configured to include Document Date, the **itemdate** column is included in the table. There can only be one row for each itemnum in the table. The **KEYGROUPDATA*** ** table is located in DBSpace6.

Column Name	Data Type	Description
itemnum	int	The Document Handle.
itemdate	int	This column exists only if Document Date is included in the group.
kg###	determined by Keyword Type Data Type	The Keyword Type assigned to the Keyword Type Group, where **###** is the unique identifier for the Keyword Type (cross-referenced to **KEYTYPETABLE**).
kgcs###	keyvaluechar (*length*)	This column exists only for Mixed Case Alphanumeric Keyword Types (in addition to the **kg###** column).
cf###	int	This column exists only for Specific Currency Keyword Types (in addition to the **kg###** column).

The indexes for the **KEYGROUPDATA*** ** table are located in DBSpace6i.

Index Name	Included Columns
Akeygroupdata*_###**	kg###, cf###†, itemnum
Akeygroupdata*_###**	kg###, itemdate‡, itemnum
keygroupdata*_2**	Itemnum

†Column only exists for Specific Currency Keyword Type index.

‡Column only exists for Keyword Type Groups that include Document Date.

KEYRECORDDATA***

Each Multi-Instance Keyword Type Group in the system has a **KEYRECORDDATA*** ** table, where ***** is the unique identifier of the Multi-Instance Keyword Type Group (**keysettablenum** in **KEYWORDSET**). There can be multiple Keyword Types assigned to a Multi-Instance Keyword Type Group. Each Keyword Type has its own **kg###** column in the table. If the Keyword Type Group is configured to include Document Date, the **itemdate** column is included in the table. There can be multiple rows for each itemnum in the table, each having a different **recordnum**. The **KEYRECORDDATA*** ** table is located in DBSpace6.

Column Name	Data Type	Description
itemnum	int	The Document Handle.
recordnum	int	The unique identifier for each row in the table.
itemdate	int	This column exists only if Document Date is included in the group.
kg###	determined by Keyword Type Data Type	The Keyword Type assigned to the Keyword Type Group, where **###** is the unique identifier for the Keyword Type (cross-referenced to **KEYTYPETABLE**).
kgcs###	keyvaluechar (*length*)	This column exists only for Mixed Case Alphanumeric Keyword Types (in addition to the **kg###** column).
cf###	int	This column exists only for Specific Currency Keyword Types (in addition to the **kg###** column).

The indexes for the **KEYRECORDDATA*** ** table are located in DBSpace6i.

Index Name	Included Columns
Akeygroupdata*_###**	kg###, cf###†, itemnum
Akeygroupdata*_###**	kg###, itemdate‡, itemnum
keyrecorddata*_2**	itemnum, recordnum

†Column only exists for Specific Currency Keyword Type index.

‡Column only exists for Keyword Type Groups that include Document Date.

Batch Tables

Documents are processed into the OnBase system in batches. These batches are routed to various queues where work can be performed.

ARCHIVEDQUEUE

The **ARCHIVEDQUEUE** table stores a record for every scanned batch. The **ARCHIVEDQUEUE** table is located in DBSpace10.

Field	Data Type	Description
batchnum	int	Unique batch number.
queuenum	int	Unique identifier of the scan queue that processed the batch.
queuename	char(25)	Name of scan queue that processed the batch.
batchname	char(200)	Name of the batch.
status	int	See Processing Table Status in Appendix A.
tmpdiskgroupnum	int	
tmplogicalplttrnum	int	
diskgroupnum	int	Unique Disk Group ID in which the batch is stored.
logicalplatternum	int	Volume number of the Disk Group in which the batch is stored.
usernum	int	User who scanned the batch.
datestarted	datetime	Date and time the batch was created.
dateended	datetime	Date and time the batch completed processing (batch transitions from queue to queue).
numberdocuments	int	Number of documents originally belonging to the batch.
archiveflags	int	
bitmapnum	int	
iconnum	int	
lastusedplatter	int	
totalpages	int	
printeddate	datetime	
totaldocuments	int	
batchflags	int	
registernum	int	

The indexes for the **ARCHIVEDQUEUE** table are located in DBSpace10.

Index Name	Included Columns
archivedqueue1	status, batchnum
archivedqueue2	batchnum, status
archivedqueue3	usernum
archivedqueue4	queuenum, status
archivedqueue5	queuenum, registernum

PARSEDQUEUE

The **PARSEDQUEUE** table stores a record for every DIP or COLD batch. The **PARSEDQUEUE** table is located in DBSpace10.

Field	Data Type	Description
batchnum	int	Unique batch number.
batchfilename	char(25)	
parsefilename	char(25)	Name of DIP or COLD process that processed the batch.
parsefilenum	int	Unique identifier of the DIP or COLD process that processed the batch.
datestarted	datetime	Date and time that the batch was created.
dateended	datetime	Date and time that the batch completed processing (batch transitions from queue to queue).
archiveflags	int	
parsingmethod	int	
itemdate	datetime	
filepath	char(25)	
diskgroupnum	int	Unique Disk Group ID in which the batch is stored.
logicalplatternum	int	Volume number of the Disk Group in which the batch is stored.
numberdocuments	int	
tmpdiskgroupnum	int	
tmplogicalplttrnum	int	
status	int	See Processing Table Status in Appendix A.
usernum	int	
verifyitemnum	int	
lastusedplatter	int	
printeddate	datetime	
processflag	int	
parserclass	int	

The indexes for the **PARSEDQUEUE** table are located in DBSpace10.

Index Name	Included Columns
parsedqueue1	status, batchnum
parsedqueue2	batchnum
parsedqueue3	status, parserclass, batchnum

Log Tables

TRANSACTIONXLOG

The OnBase **TRANSACTIONXLOG** table stores information regarding modifications to a document. The **itemnum** of a document (stored in the **num** field) can be used to retrieve all of the logged information about a document. The **TRANSACTIONXLOG** table is located in DBSpace1.

Column Name	Data Type	Description
transactionnum	int	
tmessage	char(200)	Text message describing the change that occurred to the document.
action	int	
logdate	datetime	The date the transaction was logged.
num	int	The Document Handle of the modified document.
usernum	int	The unique identifier of the OnBase user who modified the document (cross-referenced to **USERACCOUNT**).
itemnum	int	
docrevnum	int	
registernum	int	The unique identifier of the Client workstation from which the action was performed.
actionnum	int	
subactionnum	int	
severityflag	int	
tracelvl	int	
extrainfo1	int	
extrainfo2	int	

The indexes for the **TRANSACTIONXLOG** table are located in DBSpace1.

Index Name	Included Columns
transactionxlog1	transactionnum
transactionxlog2	num
transactionxlog3	itemnum

Note: The transactionxlog3 index on itemnum only exists in new OnBase databases created with OnBase 11.0 and higher.

SECURITYLOG

The OnBase **SECURITYLOG** table stores information related to system security (e.g., logging on to or off of the Client, Web Client or Unity Client, modifying Licensing, Creating/Modifying passwords). The **SECURITYLOG** table is located in DBSpace1.

Column Name	Data Type	Description
securitylognum	int	
usernum	int	The unique identifier of the OnBase user who performed the logged action (cross-referenced to **USERACCOUNT**).
registernum	int	The unique identifier of the workstation from which the action was performed (cross-referenced to **REGISTEREDUSERS**).
logdate	datetime	The date the transaction was logged.
messagetext	char(200)	Text message describing the transaction.
actionnum	int	See Log Table Actions in Appendix A.
subactionnum	int	
extrainfo1	int	
severityflag	int	
tracelvl	int	

The indexes for the **SECURITYLOG** table are located in DBSpace1.

Index Name	Included Columns
securitylog1	logdate
securitylog2	usernum, logdate
securitylog3	actionnum, logdate

SCANNINGLOG

The OnBase **SCANNINGLOG** table stores information related to the activity of scanned batches (e.g., batch created, batch indexed, batch committed). The **SCANNINGLOG** table is located in DBSpace1.

Column Name	Data Type	Description
scanninglognum	int	
usernum	int	The unique identifier of the OnBase user who performed the logged action (cross-referenced to **USERACCOUNT**).
registernum	int	The unique identifier of the workstation from which the action was performed (cross-referenced to **REGISTEREDUSERS**).
logdate	datetime	The date the transaction was logged.
messagetext	char(200)	Text message describing the transaction.
actionnum	int	See Scanning Log Actions in Appendix A.
subactionnum	int	
queuenum	int	The unique identifier of the Scan Queue in which the batch exists (cross-referenced to **SCANQUEUE**).
batchnum	int	The unique identifier of the batch (cross-referenced to **ARCHIVEDQUEUE**).
extrainfo1	int	The number of documents originally scanned in the batch (this is not updated if documents are deleted from or added to the batch).
extrainfo2	int	The number of pages originally scanned in the batch (this is not updated if pages are deleted from or added to documents).
itemnum	int	
eventnum	int	
severityflag	int	
tracelvl	int	

The indexes for the **SCANNINGLOG** table are located in DBSpace1.

Index Name	Included Columns
scanninglog1	logdate
scanninglog2	usernum
scanninglog3	batchnum
scanninglog4	queuenum
scanninglog5	eventnum, actionnum

PROCESSINGLOG

The OnBase **PROCESSINGLOG** table stores information related to various import processes and batches (e.g., batch committed, batch purged). The **PROCESSINGLOG** table is located in DBSpace1.

Column Name	Data Type	Description
processinglognum	int	
usernum	int	The unique identifier of the OnBase user who performed the logged action (cross-referenced to **USERACCOUNT**).
registernum	int	The unique identifier of the workstation from which the action was performed (cross-referenced to **REGISTEREDUSERS**).
logdate	datetime	The date the transaction was logged.
messagetext	char(200)	Text message describing the transaction.
actionnum	int	See Log Table Actions in Appendix A.
subactionnum	int	
severityflag	int	
batchnum	int	The unique identifier of the batch (cross-referenced to **ARCHIVEDQUEUE** or **PARSEDQUEUE**).
parsefilenum	int	
verifyitemnum	int	The Document Handle of the Verification Report for the batch (document will not exist if the batch has been purged).
extrainfo1	int	
tracelvl	int	
isacknowledged	int	

The indexes for the **PROCESSINGLOG** table are located in DBSpace1.

Index Name	Included Columns
processinglog1	logdate
processinglog2	batchnum
processinglog3	parsefilenum, logdate
processinglog4	severityflag, logdate

Medical Records Tables

CHART

The **CHART** table stores one record for every patient visit or chart in a Medical Records Management application or other chart-based implementation of OnBase. The primary key on this table is **chtnum**, which is the unique identifier for every chart in the system. The **chtnum** of a chart is linked to many other pieces of information about the chart. The **CHART** table is located in DBSpace1.

Column Name	Data Type	Description
chtnum	int	The internal unique identifier of a chart object.
chtidnumber	char(20)	The Patient Account Number, Episode Number, or Visit Number.
chttitle	char(100)	The auto-name string of the chart.
mpinumber	char(20)	The Master Patient Index Number for the patient; a unique identifier of the patient across facilities.
medrecnumber	char(20)	The Medical Record Number for the patient; a unique identifier of a patient across multiple visits within one or multiple facilities.
mrnum	int	The internal unique identifier of the patient's Medical Record Number.
facilitynum	int	The internal unique identifier of the medical Facility; a foreign key to the **MEDFACILITY** table.
defchtgroupnum	int	Not implemented.
dptnum	int	Associates chart with a hospital department; a foreign key to the **MEDDEPARTMENT** table.
defunitnum	int	Associates chart with a hospital unit; a foreign key to the **MEDUNIT** table.
nursestationnum	int	The unique identifier of the Nursing Station; foreign key to the **NURSINGSTATION** table.
bed	char(9)	The patient's bed number.
ptfirstname	char(20)	The patient's first name.
ptlastname	char(30)	The patient's last name.
ptdob	datetime	The patient's date of birth.
ptsex	int	The patient's gender: 0 = Undefined, 1 = Male, 2 = Female, 3 = Other, 4 = Ambiguous.
ptdischargestat	int	Indicates whether or not the chart has been discharged: 0 = undefined, 1 = discharged, 2 = not discharged. NOTE: This value does not actually control any logic within Medical Records processing.
admittypenum	int	The internal unique identifier of the chart's admit type; a

Column Name	Data Type	Description
		foreign key to the **ADMITTYPE** table.
daystodelay	int	Not implemented.
admitdate	datetime	Date the patient was admitted.
dischargedate	datetime	Date the patient was discharged (1964-01-01 00:00:00 if not yet discharged).
admitphysnum	int	The internal unique identifier of the chart's admitting physician; a foreign key to the **PHYSICIANINFO** table.
attendphysnum	int	The internal unique identifier of the chart's attending physician; a foreign key to the **PHYSICIANINFO** table.
chtstatus	int	The current status of the chart. (See Appendix for enumeration values.)
chtanalysisreq	int	Internal processing status using sequential coding and analysis.
primarydiagnosis	char(10)	The primary diagnosis for the patient's chart.
lengthofstay	int	The length of the patient's stay in days.
chtlockusernum	int	Not implemented.
availablemrdate	datetime	Date the chart is available for processing by the Medical Records Management processors.
uiprefnum	int	Reference to the chart's tab configuration; a foreign key to the **UIPREF** table.
chtrevnum	int	Not implemented.
vipflag	int	Indicates whether or not the patient is a VIP: 0 = Not VIP, 1 = VIP.
hoursonhold	int	The total number of hours the chart was suspended within Analysis or Reanalysis.
mpinum	int	The internal unique identifier of the patient's Master Patient Index number; a foreign key to the **MSTRPTINDEX** table.
holdusernum	int	The internal unique identifier of the user that suspended the chart within Analysis or Reanalysis; a foreign key to the **USERACCOUNT** table.
holddate	datetime	The date and time that the chart was suspended within Analysis or Reanalysis.
onhold	int	Indicates whether the chart is currently suspended within Analysis or Reanalysis: 0 = not suspended, 1 = suspended.
holdreason	char(250)	The user-entered reason for suspending the chart within Analysis or Reanalysis.
holdhours	int	The user-entered number of hours to suspend the chart within Analysis or Reanalysis.
codingholdusernum	int	The internal unique identifier of the user who suspended the chart within Coding; a foreign key to the

Column Name	Data Type	Description
		USERACCOUNT table.
codingholduserdate	datetime	The date and time that the chart was suspended within Coding.
codingonhold	int	Indicates whether the chart is currently suspended within Coding: 0 = not suspended, 1 = suspended.
codingholdreason	char(250)	The user-entered reason for suspending the chart within Coding.
codingholdhours	int	The user-entered number of hours to suspend the chart within Coding.
needsreviewcode	int	Stores multiple values indicating why chart is in Needs Review Queue. (See Appendix for enumeration values.)
datereanalyzed	datetime	The date and time the chart completed reanalysis.
decisioning	int	Describes the admit type requirements for Coding and Analysis at the time the chart was first processed. (See Appendix for enumeration values.)
altmedrecnumber	char(20)	External visit ID number.
encountertype	char(20)	Extra information related to the encounter type.
patientclass	char(20)	Admit system patient classification ID.
patienttype	char(20)	Facility-specific patient type classification.
encountercomment	char(50)	Encounter-specific comments from admit system.
ptmiddlename	char(36)	Patient's middle name.
mrcontrolsys	int	Application that controls this chart (i.e., MRMS, Epic, Eclipsys). (See Appendix for enumeration values.)
wfitemnum	int	Internal unique identifier that represents the virtual e-form used to route the chart through Workflow; foreign key to **ITEMDATA.**

The indexes for the **CHART** table are located in DBSpace1.

Index Name	Included Columns
chart1	chtnum, facilitynum
chart10	ptlastname, ptfirstname
chart11	medrecnumber
chart12	mpinumber
chart13	dischargedate
chart14	admitdate

Index Name	Included Columns
chart15	mrnum
chart16	wfitemnum
chart17	admitdate, chtstatus
chart2	dischargedate, chtstatus
chart3	onhold
chart4	codingonhold
chart5	chtidnumber, mrnum, ptssn
chart6	mpinum
chart7	chtstatus, datereanalyzed
chart8	dptnum
chart9	admittypenum

ADMITTYPE

The **ADMITTYPE** table stores one row for each admit type in an organization. The admit type configuration defines the chart's requirements and the time frame for initiation of Coding and Analysis. The **ADMITTYPE** table is located in DBSpace8.

Column Name	Data Type	Description
admittypenum	int	The internal unique identifier for the Admit Type.
admittypename	char(30)	The user-friendly name for the Admit Type.
flags	int	Describes the Admit Type's requirements and the time frame for initiation of Coding and Analysis. (See Appendix for enumeration values.)
admittypenamehl7	char(30)	The Admit Type name that appears in an HL7 message.

The indexes for the **admittype** table are located in DBSpace8.

Index Name	Included Columns
admittype1	admittypenum

CHARTANALYSIS

The **CHARTANALYSIS** table stores rows related to the processing of a chart through Coding, Analysis, Completion, Reanalysis, and other Medical Records Management queues. An individual chart (chtnum) may have one-to-many associated rows in the **CHARTANALYSIS** table indicating multiple simultaneous processes. The **CHARTANALYSIS** table is located in DBSpace1.

Column Name	Data Type	Description
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
chtstatus	int	The status of the chart in one of many queues. (See Appendix for enumeration values.)
availablephys	datetime	The date and time the associated chart became available to the physician.
analystnum	int	The internal unique identifier of the user account that was assigned to the chart in the Analysis queue; a foreign key to the **USERACCOUNT** table.
dateanalyzed	datetime	The date and time that the associated chart completed Analysis. (1964-01-01 00:00:00 if not yet completed or if not applicable.)
dateadded	datetime	The date and time that the associated chart was added to the process represented by the **CHTSTATUS** column.
reanalystnum	int	The internal unique identifier of the user account that was assigned to the chart in the Reanalysis queue; a foreign key to the **USERACCOUNT** table.
datereanalyzed	datetime	The date and time that the associated chart completed Reanalysis. (1964-01-01 00:00:00 if not yet completed or if not applicable.)
holdreason	char(250)	Not implemented.
physusernum	int	The internal unique identifier of the physician user account that is assigned to the chart in a Completion queue; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
opendfcy	int	Indicates whether the associated chart has pending (open) deficiencies for the specified **PHYSUSERNUM**. 0 = No Deficiencies, 1 = Open Deficiencies.
chtqueuetxnum	int	The internal unique identifier of the transaction log row associated with the **CHARTANALYSIS** row; a foreign key to the **CHARTQUEUETXLOG** table.
mrcontrolsys	int	Application that controls this **CHARTANALYSIS** row (i.e., MRMS, Epic, Eclipsys). (See Appendix for enumeration values.)

The indexes for the **CHARTANALYSIS** table are located in DBSpace1.

Index Name	Included Columns
chartanalysis1	chtnum
chartanalysis2	chtstatus, chtnum
chartanalysis3	analystnum, chtstatus, chtnum
chartanalysis4	reanalystnum, chtstatus, chtnum
chartanalysis5	physusernum, chtstatus, chtnum, opendfcy
chartanalysis6	opendfcy, chtnum

CHARTCODEFLOW

The **CHARTCODEFLOW** table stores rows related to the chart's current routing to one or multiple queues related to Coding, whether MRMS- or Workflow-based. One-to-many rows may exist for a given chart while processing is active. The **CHARTCODEFLOW** table is located in DBSpace1.

Column Name	Data Type	Description
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
codingflownum	int	The coding flow (life cycle) number of the coding flow (life cycle) associated with the row. 100 = System Life Cycle, 101+ = User Created Life Cycle and foreign key to the **CODINGFLOW** table.
codingqueuenum	int	The coding queue number of the coding queue associated with the row. 100 = System Queue, 101+ = User Created Queue and foreign key to the **CODINGQUEUE** table.
usernum	int	The internal unique identifier of the user assigned to the row; a foreign key to the **USERACCOUNT** table.
entrydate	datetime	The date and time that the chart associated with the row entered the life cycle and queue associated with the row.
iscompleted	int	Indicates whether the chart associated with the row has completed the life cycle and queue associated with the row. 0 = Not completed, 1 = Completed.
admittypenum	int	The internal unique identifier of the admit type associated with the chart; a foreign key to the **ADMITTYPE** table.
facilitynum	int	The internal unique identifier of the facility associated with the chart; a foreign key to the **MEDFACILITY** table.
processpriority	int	The process priority level for the chart associated with the row. 0 = Normal, 1 = High (indicates an administrative user routed the chart).
lcnum	int	The Workflow life cycle number associated with the row, when Workflow-wrapped MRMS is in use; a foreign key to the **LIFECYCLE** table.
statenum	int	The Workflow queue number associated with the row, when Workflow-wrapped MRMS is in use; a foreign key to the **LCSTATE** table.

The indexes for the **CHARTCODEFLOW** table are located in DBSpace1.

Index Name	Included Columns
chartcodeflow1	chtnum
chartcodeflow2	codingflownum, codingqueuenum
chartcodeflow3	usernum
chartcodeflow4	facilitynum, admittypenum

Index Name	Included Columns
chartcodeflow5	chtnum, statenum

CHARTCODEHIST

The **CHARTCODEHIST** table stores rows related to a chart's completed processing in one or multiple queues related to Coding, whether MRMS- or Workflow-based. One-to-many rows may exist for a given chart as each step of coding processing completes. Additionally, active rows may be found in the **CHARTCODEFLOW** table. The **CHARTCODEHIST** table is located in DBSpace10.

Column Name	Data Type	Description
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
codingflownum	int	The coding flow (life cycle) number of the coding flow (life cycle) associated with the row. 100 = System Life Cycle, 101+ = User Created Life Cycle and foreign key to the **CODINGFLOW** table.
codingqueuenum	int	The coding queue number of the coding queue associated with the row. 100 = System Queue, 101+ = User Created Queue and foreign key to the **CODINGQUEUE** table.
transituser	int	The internal unique identifier of the user who transitioned the chart out of the life cycle and queue associated with this historical row; a foreign key to the **USERACCOUNT** table.
usernum	int	The internal unique identifier of the user who was assigned to the chart in the life cycle and queue associated with this historical row; a foreign key to the **USERACCOUNT** table.
entrydate	datetime	The date and time the chart associated with the row entered the life cycle and queue associated with this historical row.
finishdate	datetime	The date and time the chart associated with the row exited the life cycle and queue associated with this historical row.
lcnum	int	The Workflow life cycle number associated with the row, when Workflow-wrapped MRMS is in use; a foreign key to the **LIFECYCLE** table.
statenum	int	The Workflow queue number associated with the row, when Workflow-wrapped MRMS is in use; a foreign key to the **LCSTATE** table.

The indexes for the **CHARTCODEHIST** table are located in DBSpace10.

Index Name	Included Columns
chartcodehist1	chtnum
chartcodehist2	codingflownum, codingqueuenum
chartcodehist3	usernum

CHARTDEFICIENCY

The **CHARTDEFICIENCY** table stores a row for each unique chart-level deficiency associated with a chart. Chart-level deficiencies include Missing Documents, Missing Dictations, and external deficiencies. Once a chart finishes Reanalysis, chart-level deficiencies are migrated from the **CHARTDEFICIENCY** table to the **COMPLETEDCHTDFCY** table. The **CHARTDEFICIENCY** table is located in DBSpace1.

Column Name	Data Type	Description
dfcynum	int	The internal unique identifier for the deficiency.
dfcytype	int	The deficiency categorization for chart-level deficiencies. (See Appendix for enumeration values.)
dfcystatus	int	The status of the deficiency within the Completion process. 1 = Pending, 2 = Completed, 3 = Verified by Reanalyst, 4 = Burn to Document, 5 = Rejected, 6 = Completed by Secondary, pending Primary.
dfcymessage	text	Field for the Analyst/Reanalyst to communicate with the physician regarding the deficiency.
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
analystnum	int	The internal unique identifier of the user account (typically an Analyst) that created the deficiency; a foreign key to the **USERACCOUNT** table.
dateadded	datetime	The date and time the deficiency was added to the chart.
physassignednum	int	The internal unique identifier of the physician who was assigned the deficiency; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
physcompletednum	int	The internal unique identifier of the physician who completed the deficiency; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
physdecisiondate	datetime	The date and time that the deficiency was decided on by the physician referenced in the **physcompletednum** column.
reanalystnum	int	The internal unique identifier of the user account that verified this deficiency in Reanalysis; a foreign key to the **USERACCOUNT** table.
datereanalyzed	datetime	The date and time the deficiency was verified in Reanalysis.
rejectreason	char(250)	Field that allows the physician to communicate with the Analyst/Reanalyst to indicate a reason for rejecting the deficiency.
itemtypenum	int	Document type number of the missing document; a foreign key to the **DOCTYPE** table.
flags	int	Additional information related to the chart-level deficiency. A bitflag value 0x0000 = None, 0x0001 = Analysis Server Generated.
delinqlevel	int	Current delinquency level of the deficiency as calculated by the MRMS Delinquency process and Aging & Levels configuration; a foreign key to the **DELINQUENCYLEVEL**

Column Name	Data Type	Description
		table.
hourstillsuspen	int	Number of hours before (+) or after (-) physician suspension as calculated by the MRMS Delinquency process and Aging & Levels configuration.
resassignednum	int	The internal unique identifier of the assigned secondary signer in a dual-signature implementation; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
rescompletednum	int	The internal unique identifier of the secondary signer who completed the deficiency in a dual-signature implementation; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
resdecisiondate	datetime	The date and time that the deficiency was decided on by the secondary signer referenced in the **rescompletednum** column.
proceduredate	datetime	The date of the procedure associated with the deficiency.
extdoctypenum	int	Foreign key to the **EXTDFCYDOCTYPE** table, which stores the external deficiency Document Types for external medical integrations.
reviewstatus	int	Status to mark a deficiency as reviewed or not reviewed.
reviewusernum	int	The internal unique identifier of the last user to mark the deficiency as reviewed; a foreign key to the **USERACCOUNT** table.
reviewdate	datetime	The date and time when the deficiency was last reviewed. (1964-01-01 00:00:00 if never reviewed.)

The indexes for the **CHARTDEFICIENCY** table are located in DBSpace1.

Index Name	Included Columns
chartdeficiency1	dfcynum
chartdeficiency3	chtnum, dfcystatus, physassignednum, dfcytype
chartdeficiency4	dfcystatus
chartdeficiency5	physassignednum, dfcytype

CHARTDELINQLOG

The **CHARTDELINQLOG** table stores one row for each chart, physician, and unique deficiency combination that reaches a Delinquency Level status higher than 0 based on the Delinquency Aging & Levels configuration. As further levels of delinquency are reached or delinquent deficiencies are completed, the rows are updated accordingly. The **CHARTDELINQLOG** table is located in DBSpace10.

Column Name	Data Type	Description
chtdfcynum	int	The internal unique identifier for the deficiency; a foreign key to the **CHARTDEFICIENCY** or **DOCDEFICIENCY** tables.
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
physusernum	int	The internal unique identifier of the physician user account that is assigned the deficiency; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
lastdatedelinq	datetime	The date and time that the deficiency was last evaluated as delinquent.
hoursdelinq	int	The number of hours that the deficiency has been at the delinquency level indicated by the **delinqlevel** column.
isactivelydelinq	int	Status column indicating if the deficiency is currently delinquent; 0 = not delinquent, 1 = delinquent.
dateadded	datetime	The date and time the deficiency was first evaluated as delinquent.
delinqlevel	int	Either the current delinquency level, if **isactivelydelinq** = 1, or the highest delinquency level reached, if **isactivelydelinq** = 0.

The indexes for the **CHARTDELINQLOG** table are located in DBSpace10.

Index Name	Included Columns
chartdelinqlog1	chtnum, physusernum
chartdelinqlog2	dateadded, lastdatedelinq, physusernum, isactivelydelinq
chartdelinqlog3	isactivelydelinq

CHARTLESSDOCS

The **CHARTLESSDOCS** table contains entries for documents that OnBase was unable to associate with a specific chart on archive. The **CHARTLESSDOCS** table is located in DBSpace1.

Column Name	Data Type	Description
itemnum	int	The internal unique identifier of the document; a foreign key to **ITEMDATA**.
mpinum	int	The internal unique identifier of the patient's Master Patient Index number, if identified on archive; a foreign key to **MSTRPTINDEX**.
mrnum	int	The internal unique identifier of the patient's Medical Record number, if identified on archive; a foreign key to **MEDREC**.

The indexes for the **CHARTLESSDOCS** table are located in DBSpace1.

Index Name	Included Columns
chartlessdocs1	mpinum, mrnum
chartlessdocs2	mrnum, mpinum
chartlessdocs3	itemnum

CHARTQUEUETXLOG

The **CHARTQUEUETXLOG** table is a logging table that stores log entries for each queue or "context" to which the chart has been routed within Medical Records. The **CHARTQUEUETXLOG** table is located in DBSpace10.

Column Name	Data Type	Description
chtqueuetxnum	int	The internal unique identifier of the log entry.
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
chtqueuetype	int	Indicates the type of queue associated with the log entry. 1 = Coding, 2 = Analysis, 3 = Completion, 4 = Reanalysis, 101 = QA Coding, 102 = QA Analysis, 104 = QA Reanalysis.
availabletime	datetime	The date and time the chart associated with the log became available for the associated queue.
entrytime	datetime	The date and time the chart associated with the log was assigned to a user with access to the queue.
exittime	datetime	The date and time the assigned user transitioned/completed the chart associated with the log out of the queue.
usernum	int	The internal unique identifier of the user assigned to the chart in the queue associated with the log entry.
flags	int	Not implemented.

The indexes for the **CHARTQUEUETXLOG** table are located in DBSpace10.

Index Name	Included Columns
chartqueuetxlog1	chtqueuetxnum
chartqueuetxlog2	chtnum
chartqueuetxlog3	chtqueuetype, chtnum

CHARTXITEMDATA

The **CHARTXITEMDATA** table is a cross reference table that links documents to charts. The **CHARTXITEMDATA** table is located in DBSpace1.

Column Name	Data Type	Description
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
itemnum	int	The internal unique identifier of a document; a foreign key to the **ITEMDATA** table.
analysisstatus	int	Indicates the document's status within the Analysis context; 0 = undefined, 1 = normal, 2 = loose document.
currentpage	int	Not implemented.
itemtypenum	int	The document type of the referenced document; a foreign key to the **DOCTYPE** table.
hl7externaldocid	char(63)	Stores the external report ID for the document.
seqnum	int	Determines how documents are ordered within a tab or document list. This value is set by users manually sorting the document list.
relateditemnum	int	Reference to a related document; a foreign key to the **ITEMDATA** table.
relatedpagenum	int	Reference to a specific page of a related document (**relateditemnum**); a foreign key to the **ITEMDATAPAGE** table.

The indexes for the **CHARTXITEMDATA** table are located in DBSpace1.

Index Name	Included Columns
chartxitemdata1	analysisstatus, chtnum
chartxitemdata2	itemtypenum
chartxitemdata3	chtnum, itemtypenum
chartxitemdata4	itemnum
chartxitemdata5	relateditemnum

CHTCORRECTION

The **CHTCORRECTION** table contains corrections required for the associated chart. Corrections generally include adding a new document or rescanning a page. The **CHTCORRECTION** table is located in DBSpace1.

Column Name	Data Type	Description
chtcorrectionnum	int	The internal unique identifier of the chart correction row.
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
itemnum	int	The internal unique identifier of the document requiring correction; a foreign key to the **ITEMDATA** table.
itemtypenum	int	The document type number of the document requiring correction or of the missing document; a foreign key to the **DOCTYPE** table.
datecreated	datetime	The date and time that the correction request row was created.
usercreated	int	The internal unique identifier of the user who created the chart correction request; a foreign key to the **USERACCOUNT** table.
datecompleted	datetime	The date and time that the chart correction request was closed or completed; (1964-01-01 00:00:00 if not yet closed or completed).
userassigned	int	The internal unique identifier of the user that is assigned to the chart correction request; a foreign key to the **USERACCOUNT** table.
status	int	The current status of the chart correction request; 1 = open, 2 = completed, 3 = rejected.
requesttype	int	The type of chart correction request; 1 = add document, 2 = add page, 3 = scan page, 4 = transcription, 5 = add deficiency.
flags	int	Not currently implemented.
primaryphysnum	int	The internal unique identifier for the primary physician/signer associated with the correction task; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
secondaryphysnum	int	The internal unique identifier for the secondary signer associated with the correction task, if configured for dual signing; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.

The indexes for the **CHTCORRECTION** table are located in DBSpace1.

Index Name	Included Columns
chtcorrection1	chtcorrectionnum
chtcorrection2	chtnum

Index Name	Included Columns
chtcorrection3	status

CHTEXTENSION

The **CHTEXTENSION** table is an extension of the **CHART** table and stores information related to the specific patient visit. The **CHTEXTENSION** table is located in DBSpace1.

Column Name	Data Type	Description
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
medservicenum	int	The internal unique identifier of the Medical Service associated with the chart; a foreign key to the **MEDSERVICE** table.
disposition	int	Not implemented. (See **chtdisposition** column.)
totalcharges	numeric(9)	The total charges for the visit.
financialclass	char(50)	Character representation of the financial class for the associated visit (e.g., A = Medicare).
admitsource	int	Not implemented. (See **admitsrc** column.)
medpayornum	int	Uniquely identifies the visit's payor; a foreign key to the **MEDPAYOR** table.
chtdisposition	char(50)	Stores the disposition returned from HL7 for the associated chart. The disposition is the status of the patient upon discharge (e.g., 20 = Expired).
admitsrc	char(50)	Stores the admit source returned from HL7 for the associated chart.
referphysnum	int	Internal unique identifier of the physician who referred the patient for the associated visit; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
primcarephysnum	int	Internal unique identifier of the patient's primary care physician at the time of the associated visit; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.

The indexes for the **CHTEXTENSION** table are located in DBSpace1.

Index Name	Included Columns
chtextension1	chtnum

CODINGFLOW

The **CODINGFLOW** table stores configuration information for Coding Life Cycles configured within the Medical Records Management Solution. The **CODINGFLOW** table is located in DBSpace8.

Column Name	Data Type	Description
codingflownum	int	The internal unique identifier of the Coding Life Cycle.
codingflowname	char	The display name of the Coding Life Cycle.
admittypenum	int	The internal unique identifier of the Admit Type associated with the Coding Life Cycle; a foreign key to the **ADMITTYPE** table.
flags	int	Value that indicates the type of Coding Life Cycle; 0 = unstructured, 1 = structured, 2 = none.
facilitynum	int	The internal unique identifier of the facility associated with the Coding Life Cycle; a foreign key to the **MEDFACILITY** table.
processpriority	int	Not implemented.

The indexes for the **CODINGFLOW** table are located in DBSpace8.

Index Name	Included Columns
codingflow1	codingflownum
codingflow3	facilitynum, admittypenum

CODINGFLOWXQUEUE

The **CODINGFLOWXQUEUE** table is a cross-reference table that associates a Coding Life Cycle (**CODINGFLOW**) with a Coding Queue (**CODINGQUEUE**). The **CODINGFLOWXQUEUE** table is located in DBSpace8.

Column Name	Data Type	Description
codingflownum	int	The internal unique identifier of the Coding Life Cycle; a foreign key to the **CODINGFLOW** table.
codingqueuenum	int	The internal unique identifier of the Coding Queue; a foreign key to the **CODINGQUEUE** table.
seqnum	int	The sequence or order number for the associated queue within the associated life cycle; 0 = initial queue, 1 = the next queue in a structured life cycle, 2-N = the subsequent queue(s) in a structured life cycle.
transittype	int	Used to indicate whether the queue is the last queue in the sequence for a structured life cycle; 0 = Not last in sequence, 1 = Last in sequence.

The indexes for the **CODINGFLOWXQUEUE** table are located in DBSpace8.

Index Name	Included Columns
codingflowxqueue1	codingqueuenum
codingflowxqueue2	codingflownum

CODINGQUEUE

The **CODINGQUEUE** table is a configuration table that stores information about Coding Queue configured within the Medical Records Management solution. The **CODINGQUEUE** table is located in DBSpace8.

Column Name	Data Type	Description
codingqueuenum	int	The internal unique identifier of a Coding queue.
codingqueuename	char(100)	The display name of the Coding Queue.
queuetype	int	Not implemented.
flags	int	Not implemented.
iconnum	int	Not implemented.
bitmapnum	int	Not implemented.

The indexes for the **CODINGQUEUE** table are located in DBSpace8.

Index Name	Included Columns
codingqueue1	codingqueuenum

COMPLETEDCHTDFCY

The **COMPLETEDCHTDFCY** table stores a historical reference of completed chart-level deficiencies after the chart has fully completed Reanalysis. Entries in this table are migrated from the **CHARTDEFICIENCY** table upon processing after Reanalysis is fully complete. The **COMPLETEDCHTDFCY** table is located in DBSpace1.

Column Name	Data Type	Description
dfcynum	int	The internal unique identifier for the deficiency.
dfcytype	int	The deficiency categorization for chart-level deficiencies. (See Appendix for enumeration values.)
dfcymessage	text	Field for the Analyst/Reanalyst to communicate with the physician regarding the deficiency.
chtnum	int	The internal unique identifier of a chart object; a foreign key to the **CHART** table.
analystnum	int	The internal unique identifier of the user account (typically an Analyst) that created the deficiency; a foreign key to the **USERACCOUNT** table.
dateadded	datetime	The date and time the deficiency was added to the chart.
physassignednum	int	The internal unique identifier of the physician who was assigned the deficiency; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
physcompletednum	int	The internal unique identifier of the physician who completed the deficiency; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
completeddate	datetime	The date and time the deficiency was decided on by the physician referenced in the **physcompletednum** column.
reanalystnum	int	The internal unique identifier of the user account that verified this deficiency in Reanalysis; a foreign key to the **USERACCOUNT** table.
datereanalyzed	datetime	The date and time the deficiency was verified in Reanalysis.
delinqlevel	int	Highest delinquency level reached for the specified deficiency as calculated by the MRMS Delinquency process and Aging & Levels configuration; a foreign key to the **DELINQUENCYLEVEL** table.
resassignednum	int	The internal unique identifier of the assigned secondary signer in a dual-signature implementation; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
rescompletednum	int	The internal unique identifier of the secondary signer who completed the deficiency in a dual-signature implementation; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
resdecisiondate	datetime	The date and time that the deficiency was decided on by the secondary signer referenced in the **rescompletednum** column.
flags	int	Additional information related to the chart-level deficiency. A bitflag value 0x0000 = None, 0x0001 = Analysis Server

Column Name	Data Type	Description
		Generated.
reviewstatus	int	Status to mark a deficiency as reviewed or not reviewed.
reviewusernum	int	The internal unique identifier of the last user to mark the deficiency as reviewed; a foreign key to the **USERACCOUNT** table.
reviewdate	datetime	The date and time when the deficiency was last reviewed.
itemtypenum	int	Document type number of the missing document; a foreign key to the **DOCTYPE** table.

The indexes for the **COMPLETEDCHTDFCY** table are located in DBSpace1.

Index Name	Included Columns
completedchtdfcy1	chtnum
completedchtdfcy2	physassignednum

COMPLETEDDOCDFCY

The **COMPLETEDDOCDFCY** table stores a historical reference of completed document-level deficiencies after the chart has fully completed Reanalysis. Entries in this table are migrated from the **DOCDEFICIENCY** table upon processing after Reanalysis is fully complete. The **COMPLETEDDOCDFCY** table is located in DBSpace1.

Column Name	Data Type	Description
dfcynum	int	The internal unique identifier of the deficiency.
dfcytype	int	The deficiency type. (See Appendix for enumeration values.)
dfcymessage	text	Field for the Analyst/Reanalyst to communicate with the physician regarding the deficiency.
chtnum	int	The internal unique identifier of the chart; a foreign key to the **CHART** table.
itemnum	int	The internal unique identifier of the document on which the deficiency has been created; a foreign key to the **ITEMDATA** table.
itemtypenum	int	The internal unique identifier of the deficient document's document type; a foreign key to the **DOCTYPE** table.
pagenum	int	The document page number where the deficiency was placed.
notenum	int	The internal unique identifier of the OnBase note that represents the document level deficiency; a foreign key to the **NOTETABLE** table.
analystnum	int	The internal unique identifier of the user who created/placed the deficiency; a foreign key to the **USERACCOUNT** table.
dateadded	datetime	The date and time that the deficiency was created.
physassignednum	int	The internal unique identifier of the physician who was assigned the deficiency; a foreign key to the **PHYSICIANINFO** and **USERACCOUNT** tables.
physcompletednum	int	The internal unique identifier of the physician who completed the deficiency; a foreign key to the **PHYSICIANINFO** and **USERACCOUNT** tables.
physdecisiondate	datetime	The date and time that the deficiency was completed by the associated **physcompletednum** user.
reanalystnum	int	The internal unique identifier of the user who was assigned the deficiency in Reanalysis; a foreign key to the **USERACCOUNT** table.
datereanalyzed	datetime	The date and time that the deficiency was reviewed in Reanalysis.
resassignednum	int	The internal unique identifier of the assigned secondary signer in a dual-signature implementation; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
rescompletednum	int	The internal unique identifier of the secondary signer who completed the deficiency in a dual-signature implementation; a foreign key to the **USERACCOUNT**

Column Name	Data Type	Description
		and **PHYSICIANINFO** tables.
resdecisiondate	datetime	The date and time that the deficiency was decided on by the secondary signer referenced in the **rescompletednum** column.
delinqlevel	int	Highest delinquency level reached for the specified deficiency as calculated by the MRMS Delinquency process and Aging & Levels configuration; a foreign key to the **DELINQUENCYLEVEL** table.
flags	int	Stores status values related to the document level deficiency; 0x0000001 = HL7 message sent for Editable Transcriptions; 0x00000002 = Deficiency will not be burned into document; 0x00000004 = Deficiency is a Physician Query.
reviewstatus	int	Status to mark a deficiency as reviewed or not reviewed.
reviewusernum	int	The internal unique identifier of the last user to mark the deficiency as reviewed; a foreign key to the **USERACCOUNT** table.
reviewdate	datetime	The date and time when the deficiency was last reviewed.

The indexes for the **COMPLETEDDOCDFCY** table are located in DBSpace1.

Index Name	Included Columns
completeddocdfcy2	physassignednum
completeddocdfcy3	chtnum, dfcytype
completeddocdfcy4	itemnum, dfcytype

DELINQUENCYLEVEL

The **DELINQUENCYLEVEL** table stores configuration information for any configured Delinquency Levels. The **DELINQUENCYLEVEL** table is located in DBSpace8.

Column Name	Data Type	Description
delinqlevel	int	The internal unique identifier of the delinquency level; 0 = Active, 1+ = user configured.
levelfullname	char(30)	The full name for the delinquency level.
levelabbrevname	char(10)	The abbreviated name for the delinquency level.
delinqlevelcolor	int	The color associated with the level for display in the UI.

There are no indexes on the **DELINQUENCYLEVEL** table.

DFCYTXLOG

The **DFCYTXLOG** table stores the transaction log of activity occurring on a chart throughout its existence. Each chart will typically have multiple related rows in this table. The **DFCYTXLOG** table is located in DBSpace1.

Column Name	Data Type	Description
dfcytxlognum	int	The internal unique identifier of the transaction being logged.
chtnum	int	The internal unique identifier of the chart; a foreign key to the **CHART** table.
logdate	datetime	The date and time that the transaction occurred.
action	int	An enumeration value indicating a specific activity for the transaction. (See Appendix for enumeration values.)
usernum	int	The internal unique identifier of the user who performed the transaction.
itemnum	int	The internal unique identifier of the document on which the transaction was performed, if applicable.
extrainfo1	int	Stores additional information about the transaction, dependent on the value in the **action** column.
extrainfo2	int	Stores additional information about the transaction, dependent on the value in the **action** column.
extrainfo3	int	Stores additional information about the transaction, dependent on the value in the **action** column.
dfcynum	int	The internal unique identifier of the deficiency that the transaction is related to, if applicable.
messagetext	char(200)	Text description of the transaction that is being logged.
registernum	int	The internal unique identifier of the workstation on which the transaction occurred; a foreign key to the **REGISTEREDUSERS** table.
actionnum	int	Not currently implemented.
subactionnum	int	Not currently implemented.
severityflag	int	For future use.
tracelvl	int	Logging detail level.

The indexes for the **DFCYTXLOG** table are located in DBSpace1.

Index Name	Included Columns
dfcytxlog1	logdate, action
dfcytxlog2	action

Index Name	Included Columns
dfcytxlog3	chtnum, action
dfcytxlog4	usernum, action

DOCDEFICIENCY

The **DOCDEFICIENCY** table stores a row for each unique document-level deficiency associated with a chart. Document-level deficiencies include Missing Signatures, Missing Information, Editable Transcriptions, and other deficiency types. Once a chart finishes Reanalysis, document-level deficiencies are burned into the documents and then migrated from the **DOCDEFICIENCY** table to the **COMPLETEDDOCDFCY** table. The **DOCDEFICIENCY** table is located in DBSpace1.

Column Name	Data Type	Description
dfcynum	int	The internal unique identifier of the deficiency.
dfcytype	int	The deficiency type. (See Appendix for enumeration values.)
dfcystatus	int	The status of the deficiency within the Completion process. 1 = Pending, 2 = Completed, 3 = Verified by Reanalyst, 4 = Burn to Document, 5 = Rejected, 6 = Completed by Secondary, pending Primary.
dfcymessage	text	Field for the Analyst/Reanalyst to communicate with the physician regarding the deficiency.
chtnum	int	The internal unique identifier of the chart; a foreign key to the **CHART** table.
itemnum	int	The internal unique identifier of the document on which the deficiency has been created; a foreign key to the **ITEMDATA** table.
pagenum	int	The document page number where the deficiency was placed.
notenum	int	The internal unique identifier of the OnBase note that represents the document level deficiency; a foreign key to the **NOTETABLE** table.
analystnum	int	The internal unique identifier of the user that created/placed the deficiency; a foreign key to the **USERACCOUNT** table.
dateadded	datetime	The date and time that the deficiency was created.
physassignednum	int	The internal unique identifier of the physician who was assigned the deficiency; a foreign key to the **PHYSICIANINFO** and **USERACCOUNT** tables.
physcompletednum	int	The internal unique identifier of the physician who completed the deficiency; a foreign key to the **PHYSICIANINFO** and **USERACCOUNT** tables.
physdecisiondate	datetime	The date and time that the deficiency was completed by the associated **physcompletednum** user.
reanalystnum	int	The internal unique identifier of the user who was assigned the deficiency in Reanalysis; a foreign key to the **USERACCOUNT** table.
datereanalyzed	datetime	The date and time that the deficiency was reviewed in Reanalysis.
rejectreason	char(250)	The reason that the deficiency was rejected by a

Column Name	Data Type	Description
		physician.
flags	int	Stores status values related to the document level deficiency; 0x0000001 = HL7 message sent for Editable Transcriptions; 0x00000002 = Deficiency will not be burned into document; 0x00000004 = Deficiency is a Physician Query.
dirtykey	int	Used by the Epic Analysis server to update deficiencies that have been modified.
resassignednum	int	The internal unique identifier of the assigned secondary signer in a dual-signature implementation; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
rescompletednum	int	The internal unique identifier of the secondary signer who completed the deficiency in a dual-signature implementation; a foreign key to the **USERACCOUNT** and **PHYSICIANINFO** tables.
resdecisiondate	datetime	The date and time that the deficiency was decided on by the secondary signer referenced in the **rescompletednum** column.
delinqlevel	int	Highest delinquency level reached for the specified deficiency as calculated by the MRMS Delinquency process and Aging & Levels configuration; a foreign key to the **DELINQUENCYLEVEL** table.
hourstillsuspen	int	Number of hours before (+) or after (-) suspension.
reviewstatus	int	Status to mark a deficiency as reviewed or not reviewed.
reviewusernum	int	The internal unique identifier of the last user to mark the deficiency as reviewed; a foreign key to the **USERACCOUNT** table.
reviewdate	datetime	The date and time when the deficiency was last reviewed.

The indexes for the **DOCDEFICIENCY** table are located in DBSpace1.

Index Name	Included Columns
docdeficiency1	dfcynum
docdeficiency10	itemnum, dfcystatus, dfcytype
docdeficiency3	itemnum, physassignednum
docdeficiency4	notenum
docdeficiency5	chtnum, dfcystatus, physassignednum, dfcytype
docdeficiency6	itemnum, resassignednum
docdeficiency7	chtnum, dfcystatus, resassignednum, dfcytype
docdeficiency8	dfcystatus

Index Name	Included Columns
docdeficiency9	physassignednum, dfcytype

MEDDEPARTMENT

The **MEDDEPARTMENT** table stores information about configured Medical Departments. The **MEDDEPARTMENT** table is located in DBSpace8.

Column Name	Data Type	Description
dptnum	int	The internal unique identifier of the department.
dptname	char(60)	The display name of the department.
dptnamehl7	char(30)	The HL7 name configured for this department.

The indexes for the **MEDDEPARTMENT** table are located in DBSpace8.

Index Name	Included Columns
meddepartment1	dptnum

MEDFACILITY

The **MEDFACILITY** table stores a row for each configured Medical Facility. The **MEDFACILITY** table is located in DBSpace8.

Column Name	Data Type	Description
facilitynum	int	The internal unique identifier of the medical facility.
facilityname	char(50)	The display name of the medical facility.
activeperiod	int	The facility level setting for the number of days before a chart is marked inactive.
chtautoname	char(100)	The configured auto-name string for charts associated with this facility.
delinqdays	int	The facility-level setting for the number of days before a chart's deficiencies are flagged as delinquent; used when Aging & Levels are not configured.
legalstatusdelay	int	The facility-level setting for the number of days before a chart is flagged as closed.
allowemergencyacc	int	The level of emergency access allowed for charts associated with this facility; 0 = No emergency access, 1 = All users, 2 = Only physicians.
facilitynamehl7	char(30)	The HL7 name of the medical facility.
requestdays	int	The facility-level setting for the number of days before chart access expires.
flags	int	Facility-level flags.
securityflags	int	Additional flag for security options; 1 = Facility is locked down.
termdigitmask	char(20)	The mask for this facility's terminal digits.

The indexes for the **MEDFACILITY** table are located in DBSpace8.

Index Name	Included Columns
medfacility1	facilitynum

MEDREC

The **MEDREC** table stores one row for each patient Medical Record, which is typically a unique identifier of a patient within a facility but across multiple visits. The **MEDREC** table is located in DBSpace1.

Column Name	Data Type	Description
mrnum	int	The internal unique identifier of the Medical Record.
medrecnumber	char(20)	The Medical Record Number issued by the facility or hospital organization.
facilitynum	int	The medical facility associated with this patient's records; a foreign key to the **MEDFACILITY** table.
mpinumber	char(20)	The Master Patient Index number issued by the facility or hospital organization.
ptfirstname	char(20)	Patient's first name.
ptlastname	char(30)	Patient's last name.
ptdob	datetime	Patient's date of birth.
ptssn	char(20)	Patient's social security number.
ptsex	int	Patient's gender; 0 = Undefined, 1 = Male, 2 = Female, 3 = Other, 4 = Ambiguous, 5 = Not Applicable.
hl7source	char(50)	The source of the HL7 message that created the Medical Record.
mpinum	int	The internal unique identifier of the patient's Master Patient Index number; a foreign key to the **MSTRPTINDEX** table.
ptmiddlename	char(36)	Patient's middle name.
birthplace	char(75)	Patient's birthplace.
address1	char(80)	Address field for the patient.
address2	char(80)	Address field for the patient.
address3	char(80)	Address field for the patient.
city	char(75)	The city of the patient's address.
stateabbr	char(2)	The state of the patient's address.
zipcode	char(10)	The zip code of the patient's address.
mrcontactname	char(100)	The patient's emergency contact information.
homephone	char(30)	The patient's home phone number.
workphone	char(30)	The patient's work phone number.

The indexes for the **MEDREC** table are located in DBSpace1.

Index Name	**Included Columns**
medrec2	medrecnumber, mpinumber
medrec3	mpinumber, medrecnumber
medrec4	mpinum, medrecnumber, ptssn
medrec5	mrnum, medrecnumber

MEDUNIT

The **MEDUNIT** table describes a configured Medical Unit. The **MEDUNIT** table is located in DBSpace8.

Column Name	Data Type	Description
unitnum	int	The internal unique identifier of the Medical Unit.
unitname	char(50)	The display name for the Medical Unit.
unitnamehl7	char(30)	The HL7 field specifier for that identifies the unit.

There are no indexes on the **MEDUNIT** table.

MSTRPTINDEX

The **MSTRPTINDEX** table is the master index of all patients across the system. The **MSTRPTINDEX** table is located in DBSpace1.

Column Name	Data Type	Description
mpinum	int	The internal unique identifier of the patient's Master Patient Index (MPI).
mpinumber	char(20)	The Master Patient Index number issued by the hospital organization.
ptfirstname	char(20)	The patient's first name.
ptlastname	char(30)	The patient's last name.
ptdob	datetime	The patient's date of birth.
ptssn	char(20)	The patient's social security number.
ptsex	int	The patient's gender; 0 = Undefined, 1 = Male, 2 = Female, 3 = Other, 4 = Ambiguous, 5 = Not Applicable.
primecarephysnum	int	The internal unique identifier of the patient's primary care physician; a foreign key to the **PHYSICIANINFO** table.
ptmiddlename	char(36)	The patient's middle name.
ptspecialinst	text	Special instructions related to the patient.

The indexes for the **MSTRPTINDEX** table are located in DBSpace1.

Index Name	Included Columns
mstrptindex1	mpinumber, ptssn
mstrptindex3	mpinum, mpinumber

NURSINGSTATION

The **NURSINGSTATION** table stores information related to each configured Nursing Station. The **NURSINGSTATION** table is located in DBSpace8.

Column Name	Data Type	Description
nursestationnum	int	The internal unique identifier of the nursing station.
nursestationname	char(50)	The display name of the nursing station.
nursestationhl7	char(30)	The HL7 identifier for the nursing station.

There are no indexes on the **NURSINGSTATION** table.

PHYSICIANINFO

The **PHYSICIANINFO** table stores information related to configured physician user accounts. The **PHYSICIANINFO** table is located in DBSpace8.

Column Name	Data Type	Description
usernum	int	The internal unique identifier of the user account; a foreign key to the **USERACCOUNT** table.
fullphysname	char(50)	The physician's full name.
physnumber	char(20)	The configured physician number.
upinn	char(20)	Not implemented.
statelicnumber	char(20)	The number of the state license associated with the physician.
address1	char(80)	Physician's address.
address2	char(80)	Physician's address.
email	char(80)	Physician's email address.
faxnum	char(30)	Physician's fax number.
signphysname	char(50)	The physician's name as displayed on signature deficiencies.
flags	int	Flags related to physician settings; 1 = Secondary Signer.
defdisplaysec	int	The number of seconds a deficiency is displayed in slide-show (auto play) mode.
physspecialty	char(50)	Physician specialty information from an external system.
onhold	int	Flag to determine if the physician is on hold; 1 = On Hold.
lastpolltime	datetime	Used for external integrations to store the last time the physician's deficiencies were polled.
mtprovmnemonic	char(15)	The Meditech provider mnemonic for the physician.
provspecialtynum	int	The internal unique identifier used to associate the physician with a specialty; a foreign key to the **PROVIDERSPECIALTY** table.
lastname	char(50)	The physician's last name.
firstname	char(50)	The physician's first name.
middlename	char(50)	The physician's middle name.
nameprefix	char(10)	The prefix of the physician's name.
namesuffix	char(50)	The suffix of the physician's name.

Column Name	Data Type	Description
physdegree	char(50)	The physician's academic degree.

The indexes for the **COMPLETEDCHTDFCY** table are located in DBSpace8.

Index Name	Included Columns
physicianinfo1	usernum

Workflow Tables

LIFECYCLE

LIFECYCLE is the table that stores a record for each Workflow Life Cycle created. This is the parent table that cross-references with the **ITEMLC** table to obtain the name (**lifecyclename**) of a life cycle in which a document exists. Another important relationship exists with the **LCSTATE** table, in that the **scope** contains the related **lcnum** from **ITEMLC**. The **LIFECYCLE** table is located in DBSpace8.

Column Name	Data Type	Description
lcnum	int	The unique identifier of the life cycle in OnBase.
lifecyclename	char(50)	The text representation of the life cycle name.
lifecycledesc	char(80)	
initialstatenum	int	
helptext	char(250)	
bitmapnum	int	
iconnum	int	
foldertypenum	int	
flags	int	
encryptedpassword	char(40)	
wfcontenttype	int	Content Type for the life cycle. Legacy Life Cycles have wfcontenttype=0. Unity Life Cycles use Enumeration: Document = 1, Folder = 2, WorkviewItem = 3.
contentclassnum	int	
maxitemsowned	int	

The index for the **LIFECYCLE** table is located in DBSpace8.

Index Name	Included Columns
lifecycle1	lcnum

LCSTATE

LCSTATE is the table that stores a record for every queue configured in Workflow. This is the parent table that cross-references with the **ITEMLC** table to obtain the name (**statename**) of the queue in which a document exists. The **LCSTATE** table is located in DBSpace8.

Column Name	Data Type	Description
statenum	int	The unique identifier of the queue in OnBase.
statename	char(50)	The text representation of the queue name.
statedesc	char(80)	
statehelp	char(250)	
validactionflags	int	
bitmapnum	int	
iconnum	int	
queuetype	int	
defsystemwork	int	
defuserwork	int	
scope	int	The unique identifier of the life cycle to which the queue belongs (cross-referenced to **lcnum** in the **LIFECYCLE** table).
flags	int	
foldertypenum	int	
templatenum	int	
docseltasklistnum	int	
cqnum	int	
loadbalancingwork	int	
obrefresh	int	
requiredrole	int	

flags2	int	
keynum	int	
wftimerservername	char(60)	

The index for the **LCSTATE** table is located in DBSpace8.

Index Name	Included Columns
lcstate1	statenum

Legacy and Unity Life Cycles

Prior to the release of OnBase 12, only documents could be in workflow. Documents in workflow were represented using table **ITEMLC**. OnBase 12 introduced Unity Life Cycles. As well as documents, Unity Life Cycles can contain other items such as WorkView objects. (In addition, new features such as ownership & portfolios are only supported in Unity Life Cycles). Work Items in Unity Life Cycles are represented in 2 new tables, **WORKITEMCONTAINER** and **WORKITEMLC**. Document-only life cycles, now known as *Legacy* Life Cycles, continue to be supported, and they continue to be represented in **ITEMLC**.

***Documents* can reside in both Legacy and Unity Life Cycles; *non-Documents* can only reside in Unity Life Cycles.**

ITEMLC (Legacy Life Cycles)

This table contains the current representation of documents in Workflow Legacy Queues. **ITEMLC** and **WORKITEMLC** are constantly updated as work items are routed between different queues. As such, this table does not provide information on the location of a document at a past (historical) point in time; the **WFLOG** table provides such data.

Every document that exists in Workflow Legacy Life Cycles has a record in the **ITEMLC** table. **ITEMLC** stores a row for every instance of a document appearing in a Legacy Life Cycle. If a document exists in four different Legacy Life Cycles, there are four rows in the **ITEMLC** table. The **ITEMLC** table is located in DBSpace1. (A document can exist in Unity Life Cycles as well as Legacy Life Cycles; but those in Unity Life Cycles are represented in table **WORKITEMLC**).

Column Name	Data Type	Description
lcnum	int	The unique identifier of the Legacy Life Cycle in which the document resides.
itemnum	int	The Document Handle, which is the unique identifier for a document in OnBase.
statenum	int	The unique identifier of the Legacy Queue in which the document resides.
foldernum	int	
transdate	datetime	
transitnum	int	
usernum	int	

assignedtogroup	int	
assignedtouser	int	
status	int	Lists whether the document has been deleted (16 = deleted, 0 = not deleted).

The indexes for the **ITEMLC** table are located in DBSpace1

Index Name	Included Columns
itemlc3	itemnum, lcnum, statenum, status
itemlc4	statenum, status, itemnum
itemlc5	statenum, status, transdate, itemnum
itemlc6	statenum, itemnum
itemlc7	statenum, transdate, itemnum

WORKITEMCONTAINER (Unity Life Cycles)

The **WORKITEMCONTAINER** table contains a single entry for each Work Item if it resides in any Unity Life Cycle.

(A Work Item may be in multiple Unity Life Cycles, in which case it will have a single **WORKITEMCONTAINER** entry and multiple **WORKITEMLC** entries, one for each Unity Life Cycle the Work Item is in).

Column Name	Data Type	Description
contentnum	int	ID of work item. contentnum may not be unique for a given contenttype; if contentnum is not unique, then the combination contentnum + contentclassnum is unique. For documents contentnum is unique; it is the doc handle.
wfcontenttype	int	Enumeration of content type. Document = 1, Folder = 2, WorkviewItem = 3.
portfolionum	int	
flags	int	
contentclassnum	int	ID of work item class. Only needed if contentnum is not unique for a given contenttype; in those cases, the unique identifier is contentnum + contentclassnum. For documents, contentnum is unique so contentclassnum is zero. For WorkView items, contentnum is not unique, so contentclassnum is the WorkView class ID.

Index Name	Included Columns
workitemcontainer1	contentnum, wfcontenttype
workitemcontainer2	contentnum, contentclassnum, wfcontenttype

WORKITEMLC (Unity Life Cycles)

Every active work item that exists in Workflow Unity Life Cycles has a record in the **WORKITEMLC** table. **WORKITEMLC** stores a row for every instance of a work item (not necessarily a document) appearing in a Unity Life Cycle. (For example, if a work item resides in four different Unity Life Cycles, there are four rows in the **WORKITEMLC** table.) The **WORKITEMLC** table is located in DBSpace1. (A document work item can exist in Legacy Life Cycles as well as Unity Life Cycles, but the Legacy Life Cycles are represented in a different table, **ITEMLC***.* Non-document work items can only exist in Unity Life Cycles).

Column Name	Data Type	Description
lcnum	int	The unique identifier of the Unity Life Cycle in which the Work Item resides.
statenum	int	The unique identifier of the Unity Queue in which the Work Item resides.
contentnum	int	ID of work item. contentnum may not be unique for a given contenttype; if contentnum is not unique then the combination contentnum + contentclassnum is unique. For documents contentnum is unique; it is the doc handle.
wfcontenttype	int	Enumeration of content type. Document = 1, Folder = 2, WorkviewItem = 3.
transdate	datetime	
priority	int	
versionid	int	
ownernum	int	
ownedstatus	int	
ownedsince	datetime	
lastupdated	datetime	
flags	int	
contentclassnum	int	ID of work item class. Only needed if contentnum is not unique for a given contenttype; in those cases the unique identifier is contentnum + contentclassnum. For documents, contentnum is unique so contentclassnum is zero. For WorkView items contentnum is not unique, so contentclassnum is the WorkView class ID.

The indexes for the **WORKITEMLC** table are located in DBSpace1

Index Name	Included Columns

workitemlc7	contentnum, contentclassnum, lcnum, statenum, wfcontenttype
workitemlc8	statenum, contentnum, contentclassnum, wfcontenttype
workitemlc9	statenum, transdate, contentnum, contentclassnum, wfcontenttype
workitemlc10	ownedstatus, statenum, contentnum, contentclassnum, wfcontenttype
workitemlc11	ownernum, statenum, contentnum, contentclassnum, wfcontenttype

WORKITEMLCUNAVAIL (Unity Life Cycles)

This table has identical structure to **WORKITEMLC**. It is used to hold entries for Work Items that have been deleted but not purged. When a Work Item is deleted and it resides in one or more Unity Life Cycles, all its corresponding **WORKITEMLC** entries are moved to **WORKITEMLCUNAVAIL**.
If the Work Item is subsequently undeleted, the entries are moved back.

Column Name	Data Type	Description
lcnum	int	The unique identifier of the Unity Life Cycle in which the Work Item resides.
statenum	int	The unique identifier of the Unity Queue in which the Work Item resides.
contentnum	int	ID of work item. contentnum may not be unique for a given contenttype; if contentnum is not unique then the combination contentnum + contentclassnum is unique. For documents contentnum is unique; it is the doc handle.
wfcontenttype	int	Enumeration of content type. Document = 1, Folder = 2, WorkviewItem = 3.
transdate	datetime	
priority	int	
versionid	int	
ownernum	int	
ownedstatus	int	
ownedsince	datetime	
lastupdated	datetime	

flags	int	
contentclassnum	int	ID of work item class. Only needed if contentnum is not unique for a given contenttype; in those cases the unique identifier is contentnum + contentclassnum. For documents, contentnum is unique so contentclassnum is zero. For WorkView items contentnum is not unique, so contentclassnum is the WorkView class ID.

The index for the **WORKITEMLCUNAVAIL** table is located in DBSpace1

Index Name	Included Columns
workitemlcunavail2	contentnum, contentclassnum, wfcontenttype

Deleting Work Items

If a Work Item is deleted from OnBase while it is still in Workflow, then any **ITEMLC** and **WORKITEMLC** records for the Work Item are treated differently.

ITEMLC Entries

If the deleted Work Item is in any Legacy Life Cycles, it must be a Document. On deletion, the **status** for the document in any **ITEMLC** table entries is changed to 16. If the document is subsequently undeleted, the **status** for the document in any **ITEMLC** table entries is changed back to 0.

WORKITEMLC Entries

If the deleted Work Item is in any Unity Life Cycles, on deletion any **WORKITEMLC** table entries for the Work Item are moved to another table, **WORKITEMLCUNAVAIL**. If the Work Item is subsequently undeleted, any **WORKITEMLCUNAVAIL** entries are moved back to **WORKITEMLC**.

Note: The reason that for the changed behavior is because the presence in **ITEMLC** of a large number of entries for deleted but not purged documents could skew the database statistics and lead to a performance degradation. By using a separate table, only active items contribute to the statistics.

Purging Work Items

If a Work Item is purged, then any **ITEMLC**, **WORKITEMLC** and **WORKITEMLCUNAVAIL** entries for the Work Item are removed.

Workflow Log Tables

Tables **ITEMLC** and **WORKITEMLC** contain the current representation of items in Workflow.
They are constantly updated as work items are routed between different queues. As such, these tables provide a snapshot of the current state of items within workflow. **ITEMLC** and **WORKITEMLC** do *not* provide information on the location of a work item at a past (historical) point in time; the workflow log tables **WFLOG, WFTRANSACTIONLOG,** and **WFTRANSACTIONMSG** provide such historical data.

WFLOG

The **WFLOG** table lists the Workflow Queues in which Work Items have been. When a Work Item transitions from one queue to another, a record is created for each transition. Also, the prior record is updated with the **exittime**, **statenumto,** and **exitusernum**. All historical queue transitions are stored in this table. The **WFLOG** table is located in DBSpace1.

Column Name	Data Type	Description
statenum	int	The unique identifier of the Queue that the Work Item entered.
lcnum	int	The unique identifier of the life cycle that the Work Item entered.
itemnum	int	ID of work item. Corresponds to itemnum in itemlc, contentnum in workitemlc (for documents this is the Document Handle)
usernum	int	The unique identifier of the OnBase user who transitioned the Work Item into the queue.
entrytime	datetime	The time the Work Item entered the queue.
exittime	datetime	The time the Work Item exited the queue. A default value of 1964-01-01 00:00:00.000 indicates that the Work Item is still in the queue.
flags	int	
exitusernum	int	The unique identifier of the OnBase user who transitioned the Work Item out of the queue. If the value is 0 and the exittime is no longer the default value (1964-01-01 00:00:00.000), then the document exited the life cycle.
statenumto	int	The queue to which the Work Item was transitioned.
wfcontenttype	int	Enumeration of content type. Document = 1, Folder = 2, WorkviewItem = 3.
contentclassnum	int	ID of work item class. non zero only if itemnum is not unique for a given contenttype; in those cases the unique

		identifier is itemnum + contentclassnum. For documents, itemnum is unique so contentclassnum is zero. For WorkView items itemnum is not unique, so contentclassnum is the WorkView class ID.

The indexes for the **WFLOG** table are located in DBSpace1

Index Name	Included Columns
wflog8	statenum, entrytime, exittime
wflog9	itemnum, contentclassnum, lcnum, entrytime, statenum, wfcontenttype
wflog10	itemnum, contentclassnum, exittime, wfcontenttype

WFTRANSACTIONLOG

The **WFTRANSACTIONLOG** and **WFTRANSACTIONMSG** tables, along with their corresponding functionalities, were added in OnBase 5.0. For these tables to contain data, it is necessary to enable logging via one of the following methods:

- Using the **SYS-Custom Log** Action.
- Selecting the **Log Execution** check box for a Workflow Task List, Rule, or Action.
- Selecting the **Log Start/Stop** check box for a Timer.

The **WFTRANSACTIONLOG** table is located in DBSpace1.

Column Name	Data Type	Description
wftransactionnum	int	The unique identifier of the transaction.
lcnum	int	Life cycle number (see **LIFECYCLE** table).
statenum	int	Queue number (see **LCSTATE** table).
itemnum	int	ID of work item. Corresponds to itemnum in itemlc, contentnum in workitemlc (for documents this is the Document Handle), unless the **objecttype** is 35/36 (timer start/stop), in which case the value is zero.
usernum	int	The unique identifier of the OnBase user who performed the logged activity (see **useraccount** table).
logdate	datetime	The date/time of the entry.
objecttype	int	See Comments below.
objectnum	int	Dependent upon the value in the **objecttype** column, see comments below.
param1	int	If object type is 36 (timer stop), then this contains the number of documents that were processed by the timer. Otherwise, this is always zero.
param2	int	Not used, always zero.
flags	int	Not used, always zero.
objectname	char(50)	
wfcontenttype	int	Enumeration of content type. Document = 1, Folder = 2, WorkviewItem = 3
contentclassnum	int	ID of work item class. non zero only if itemnum is not unique for a given contenttype; in those cases the unique identifier is itemnum + contentclassnum.

		For documents, itemnum is unique so contentclassnum is zero. For WorkView items itemnum is not unique, so contentclassnum is the WorkView class ID.

Comments

The value in the **objecttype** column can be one of the following values:

objecttype Value	objectnum Value	Description
31	ID of the action (see **ACTION** table)	This entry is made by the **SYS – Custom Log Entry** workflow action.
32	ID of the action (see **ACTION** table)	This entry is made if the action has the **Log Execution** check box selected in Config.
33	ID of the tasklist (see **TASKLIST** table)	This entry is made if the task list has the **Log Execution** check box selected in Config.
34	ID of the rule (see **RULETABLE** table)	This entry is made if the rule has the **Log Execution** check box selected in Config.
35	ID of the timer (see **LCTIMER** table)	This entry is made if the timer has the **Log Execution** check box selected in Config. The entry is made when the timer starts.
36	ID of the timer (see **LCTIMER** table)	This entry is made if the timer has the **Log Execution** check box selected in Config. The entry is made when the timer finishes.

The indexes for the **WFTRANSACTIONLOG** table are located in DBSpace1.

Index Name	Included Columns
wftransactionlog1	wftransactionnum
wftransactionlog3	objecttype, objectnum, logdate
wftransactionlog4	itemnum, contentclassnum, logdate, wfcontenttype

WFTRANSACTIONMSG

A row is created in the **WFTRANSACTIONMSG** table under the following conditions:
· The **SYS – Custom Log Entry** Workflow Action has executed.
· The Action, Task List, or Rule has the **Log Execution** check box selected, and the Action, Task List, or Rule is disabled.

The **WFTRANSACTIONMSG** table is located in DBSpace1.

Column Name	Data Type	Description
wftransactionnum	int	Relates to **wftransactionnum** column in **WFTRANSACTIONLOG** table.
flags	int	Not used, always zero.
wfmessage	char(250)	Dependent upon the **objecttype** in the **WFTRANSACTIONLOG** table (see comments below).

Comments

The value in the **wfmessage** column depends on the object type:

Object Type	wfmessage Value
31 (SYS - Custom Log Entry action)	The text specified in the action's configuration.
32 (Action	If the action is disabled, the text reads:

execute)	**Action is disabled**.
33 (Task list execute)	If the task list is disabled, the text reads: **Task list is disabled**.
34 (Rule execute)	If the rule is disabled, the text reads: **Rule is disabled**.

The index for the **WFTRANSACTIONMSG** table is located in DBSpace1**.**

Index Name	**Included Columns**
wftransactionmsg1	wftransactionnum, flags

WorkView Tables

RMAPPLICATION

The **RMAPPLICATION** table stores a record for every configured WorkView Application. The **RMAPPLICATION** table is located in the Primary database file.

Column Name	Data Type	Description
rmApplicationID	int	Unique identifier of the WorkView Application.
rmApplicationName	char(100)	Name of the WorkView Application.
filterBarPosition	int	
filterBarWidth	int	
bTrackChanges	int	
bTrackMemoChanges	int	
calendarID	int	
userIdentity	char(100)	
defaultFilterID	int	
appSessionScriptID	int	
commonScriptID	int	
appGroupName	varchar(51)	
logVersion	int	
maxFavorites	int	
maxHistory	int	
defaultCatalogID	int	

The index for the **RMAPPLICATION** table is located in the Primary database file and is a clustered index.

Index Name	Included Columns
rmApplication_pk	rmApplicationID (primary key)

RMCLASS

The **RMCLASS** table contains one row for each Class created within WorkView. The **RMCLASS** table is located in the Primary database file.

Column Name	Data Type	Description
classID	int	Unique identifier of the WorkView Class.
className	char(255)	Name of the WorkView Class.
displayName	char(255)	Display name of the WorkView Class.
storeRevisions	int	
fAllowDirectCreate	int	
objNamePattern	char(255)	
dynamicObjectNames	int	
bTrackChanges	int	
bTrackMemoChanges	int	
itemtypenum	int	
eventScriptID	int	
extendsClassID	int	
flags	int	
ODBCSourceName	varchar(101)	
ODBCUserName	varchar(128)	
ODBCPassword	varchar(128)	
extTableName	varchar(101)	
bCacheODBC	int	
LinkedServerName	varchar(101)	
LinkedServerDBName	varchar(61)	
Description	varchar(1024)	
bMismatchedIds	int	
defaultFilterID	int	

The index for the **RMCLASS** table is located in the Primary database file and is a clustered index.

Index Name	Included Columns
rmClass_pk	classID (primary key)

RMAPPLICATIONCLASSES

The **RMAPPLICATIONCLASSES** table stores the association between a Class and an Application. A Class can be used in one or many Applications. The **RMAPPLICATIONCLASSES** table is located in the Primary database file.

Column Name	Data Type	Description
rmApplicationClassID	int	Unique identifier of the Application-Class relationship.
rmApplicationID	int	Unique identifier of the Application (cross-referenced to **RMAPPLICATION**).
classID	int	Unique identifier of the Class (cross-referenced to **RMCLASS**).
sequenceID	int	

The index for the **RMAPPLICATIONCLASSES** table is located in the Primary database file and is a clustered index.

Index Name	Included Columns
rmAppClasses_pk	rmApplicationClassID (primary key)

RMATTRIBUTE

The **RMATTRIBUTE** table stores field-level data entities used in WorkView Applications. Attributes are unique to classes. The **RMATTRIBUTE** table is located in the Primary database file.

Column Name	Data Type	Description
attributeName	char(100)	Display name of the Attribute.
dataType	int	Type of data.
attributeID	int	Unique identifier of the Attribute (cross-referenced to **attr****** in **RMOBJECTINSTANCE####**).
relatedClassID	int	
dataLen	int	
dataID	int	
defaultValue	char(100)	
bParentMustExist	int	
indexType	int	
bTrackChanges	int	
displayName	char(100)	
extColumnName	varchar(101)	
description	varchar(1024)	
maskPattern	varchar(101)	
maskStatics	varchar(101)	
maskflags	int	
extappFlags	int	
extappID	int	

The index for the **RMATTRIBUTE** table is located in the Primary database file and is a clustered index.

Index Name	Included Columns
rmAttribute_pk	attributeID (primary key)

RMCLASSATTRIBUTES

The **RMCLASSATTRIBUTES** table stores the association between an Attribute and a Class. An Attribute can only be linked to a single Class. The **RMCLASSATTRIBUTES** table is located in the Primary database file.

Column Name	Data Type	Description
classAttributeID	int	Unique identifier of the Class-Attribute relationship.
classID	int	Unique identifier of the Class (cross-referenced to **RMCLASS**).
attributeID	int	Unique identifier of the Attribute (cross-referenced to **RMATTRIBUTE**).
sequenceID	int	

The index for the **RMCLASSATTRIBUTES** table is located in the Primary database file and is a clustered index.

Index Name	Included Columns
rmClassAttrs_pk	classAttributeID (primary key)

RMOBJECT

The **RMOBJECT** table contains one row for every object that is created in WorkView. The **RMOBJECT** table is located in the Primary database file.

Column Name	Data Type	Description
objectID	int	Unique identifier of the WorkView object.
objectName	char(255)	Name of the WorkView object.
parentObjectID	int	
classID	int	The class to which the object belongs (cross-references to **RMCLASS**).
createdBy	char(30)	The user who created the WorkView object.
createdDate	datetime	Date the WorkView object was created.
writeStatus	int	
statusID	int	
activeStatus	int	0 = active, 1 = inactive, 2 = deleted.

The index for the **RMOBJECT** table is located in the Primary database file and is a clustered index.

Index Name	Included Columns
rmObject_pk	objectID (primary key)

RMOBJECTHISTORY

The **RMOBJECTHISTORY** table contains one row for every modification that occurs for an RM object. The **RMOBJECTHISTORY** table is located in the Primary database file.

Column Name	Data Type	Description
transactionID	int	Unique identifier of the modification.
objectID	int	Unique identifier of the WorkView object that was changed (cross-referenced to **RMOBJECT**).
attributeID	int	Unique identifier of the attribute that was changed.
transactionDate	datetime	Date/time of the change.
startValue	char(255)	Original attribute value.
endValue	char(255)	Value to which the attribute was changed.
userName	char(50)	User who enacted the change.

The indexes for the **RMOBJECTHISTORY** table are located in the Primary database file.

Index Name	Included Columns
rmObjectHistory_objectID_ix	objectID
rmObjectHistory_txID_ix	transactionID

RMOBJECTINSTANCE####

The **RMOBJECTINSTANCE####** table stores one row for every object created in WorkView and the attribute values for that object. The **RMOBJECTINSTANCE####** table is located in the Primary database file.

Column Name	Data Type	Description
objectID	int	ID of the object (cross-referenced to **RMOBJECT**).
activeStatus	int	
objectInstanceID	int	
revisionID	int	
revisionDate	datetime	
revisionBy	char(30)	
attr****	determined by attribute	Attribute assigned to the object, where ******** is the unique identifier of the attribute (cross-referenced to **RMATTRIBUTE**).

The index for the **RMOBJECTINSTANCE####** table is located in the Primary database file and is a clustered index.

Index Name	Included Columns
rmObjectInstance_####_pk	objectid (primary key)

Appendix A

PROCESSING TABLE STATUS

The **ARCHIVEDQUEUE** and **PARSEDQUEUE** tables both have a **status** field. This field is very important as it designates the queue in which the batch currently exists (e.g., Awaiting Commit, Committed). The status numbers in the database are the same for both batch tables. The table below lists all the possible batch statuses along with their mappings to an OnBase queue.

OnBase Queue (Archived/Parsed Queue)	**Status Value**
Awaiting Indexing	0
Incomplete Process/Index In Progress	1
Awaiting Commit	2
Incomplete Commit	3, 4, 5, 6
Committed	8
Incomplete Purge	9
Awaiting OCR	14
Checked Out Disconnected Scanning	17
Disconnected Scan Incomplete Upload	18
Incomplete Archive	19
Secondary Awaiting Index	20
Secondary Index in Progress	21
Failed Automatic OCR	22
Awaiting Doc Separation	23
Line Item Separation (Image Segment Archiver Queue)	24
ADF Error Queue	25
Awaiting Re-Index	26
Re-Index in Progress	27
Check Error Queue	28
ADF Decisioning Queue	29
Administrator Repair	30
Awaiting QA Image Quality Review	31
Awaiting QA Review	32
Awaiting QA ReScan	33
Awaiting Manager Resolution	34

OnBase Queue (Archived/Parsed Queue)	**Status Value**
Awaiting QA Re-Index	35
QA Re-Index in Progress	36
In process	37
Awaiting PDF Conversion	38
Scheduled Processes	39
Error Correction Queue	40
Awaiting Transfer to Host (SAP Early Archiving)	41
Awaiting External Index	43
Awaiting Barcode Processing	44
ADF Decision Error Queue	45
Awaiting Image Process	46
Custom Process	47
Ad Hoc Re-scan	48
Branch Capture Balancing Queue	49
Branch Capture In Process Queue	50
Awaiting Zonal OCR	51
Awaiting Ad Hoc Zonal OCR	52
Pull Slips	53
Awaiting Ad Hoc Verification	54
QA Review In Progress	55
Synchronization Pending	56
Synchronization Complete	57
Synchronization Failed	58
Synchronization Queued	59
Synchronization Processed	60
Export Awaiting Transfer	61
Export Pending Verification	62
Export Complete	63
Export Error	64
Synchronization History	65
Doc Transfer Export Hisory	66
Awaiting Formless Indexing	67
Awaiting Queue Sorting	68

SCANNING LOG ACTIONS

Actions that occur against a scanned batch get logged in the **SCANNINGLOG** table. Actions that involve a batch transition have an **eventnum** of 1; those that involve the actual action have an **eventnum** of 2.

Action	eventnum	actionnum
To Awaiting Index	1	1
To Index in Progress	1	2
To Second Awaiting Index	1	3
To Second in Progress	1	4
To Awaiting OCR	1	5
To Awaiting Commit	1	6
To Begin Commit	1	7
To Finish Commit	1	8
To Begin Purge	1	9
To Failed OCR	1	10
To Document Slicer	1	11
To EOB Queue	1	12
To Reindex Queue	1	13
To Reindex in Progress	1	14
To Incomplete Disconnected Scan	1	15
To Checked Out Disconnected Scan	1	16
To Administrative Repair	1	17
To QA Image Quality	1	18
To QA Review	1	19
To QA Needs Rescan	1	20
To Manager Resolution	1	21
To QA Reindex	1	22
To QA Reindex in Progress	1	23
To PDF Conversion	1	24
To Awaiting Archive Link	1	25
To External Indexing	1	26
To Awaiting Barcode	1	27
To Awaiting Image Process	1	28
To Awaiting Custom Process	1	29
To Ad-Hoc Rescan	1	30
To Awaiting Zonal OCR	1	31
Create Batch	2	200

Action	eventnum	actionnum
Delete Batch	2	201
Perform Index	2	202
Perform Second Index	2	203
Perform OCR	2	204
Scan More Batch	2	205
Skip OCR	2	206
Full Index Batch	2	207
Cancel Batch	2	208
Commit Batch Manual	2	209
Commit Batch Auto	2	210
OCR Batch Auto	2	211
Change Scan Queue	2	212
Perform Reindex	2	213
Barcode Append Page	2	214
Supervisor Advance	2	215
Supervisor Checkin	2	216
Perform QA Image Quality	2	217
Perform QA Review	2	218
Perform QA Rescan	2	219
Perform Manager Resolution	2	220
Perform QA Reindex	2	221
Perform Administrative Repair	2	222
Perform PDF Conversion	2	223
Skip PDF Conversion	2	224
PDF Convert Auto	2	225
Skip Document Slicer	2	226
Perform Document Slicer	2	227
Cancel External Index	2	228
Create Batch from DIP	2	229
Scan More Pages	2	230
Barcode Append Document	2	231
Keyword Match Append Document	2	232
Perform Barcode Processing	2	233
Perform Image Processing	2	234
Image Process Auto	2	235

Action	eventnum	actionnum
Barcode Process Auto	2	236
Sent to ArchiveLink	2	237
Perform EOB Slicing	2	238
Perform Custom Process	2	239
Skip Custom Process	2	240
Create Batch from Existing	2	241
Skip EOB Slicer	2	242
Spik Pre-Index	2	243
Perform Ad-Hoc Rescan	2	244
Perform Zonal OCR	2	245
Auto Zonal OCR	2	246
Skip Zonal OCR	2	247

LOG TABLE ACTIONS

Prior to OnBase version 5.0, all activity was logged to the **TRANSACTIONXLOG** table. In OnBase version 5.0 and later, transactions are logged to multiple log tables. Document-related transactions are all stored in the OnBase **TRANSACTIONXLOG** table. The **txlog.action** column in the tables below lists the action from the **TRANSACTIONXLOG** for versions prior to 5.0 for historical reporting purposes.

ADMINLOG

Log Message	**txlog.action**	**actionnum**	**subactionnum**
Exception Report Created	16386	1	1
Configuration Report Created	16387	1	2
Keyset Usage Report Created	16388	1	3
Document History Report Created	16389	1	4
Transaction Log Report Created	16390	1	5
Email Audit Log Report Created		1	6
Folder History Report Created		1	7
Records Mgmt Report Created		1	8
Purged Transaction Log	4194304	2	1
Transaction Log Viewed	4194305	2	2
Purged Workflow Log	4194308	2	3
Removed Process Lock	69	2	4
Performed a Full Text Index Search	32774	3	1
Started an Export	32772	3	2
Started an Import	32773	3	3
Terminal Emulation	32768	3	4
Removed Lock		3	5
Verity Search		3	6
Delete Folders		3	7
Delete Orphan Folders		3	8
Purged Document		3	9
Successfully created Test System		4	1
Successfully create Stand Alone Test System		4	2
Successfully Migrated Test System Back to Production		4	3
Unlinked Test System from Production		4	4

Log Message	txlog.action	actionnum	subactionnum
Test Migration to Production		4	5
Copied System Docs to Test		4	6
Unable to Migrate or Unlink – Mismatched GUIDs		4	7
Unable to Migrate – Mismatched Database Version		4	8
Unable to Migrate – Mismatched IDs		4	9
Error when Migrating System Docs		4	10
Unable to Migrate back to Prod		4	11
Unable to Migrate – Mismatched DB		4	12
Error Creating Test System		4	13
Force database upgrade on login		4	14

CBLOG

Log Message	txlog.action	actionnum	subactionnum
Viewed Threads		1	1
Viewed Post		1	2
Created Post		1	3
Subscribed to Notifications		1	4
Unsubscribed from Notifications		1	5
Created Workspace		2	1
Created Workspace from Template		2	2
Activated Workspace		2	3
Deactivated Workspace		2	4
Viewed Workspace		2	5
Added Document		3	1
Removed Document		3	2
Added Object		3	3
Removed Object		3	4
Posted to Workspace		3	5
Added User to Workspace		4	1
Removed User from Workspace		4	2
Added User to Thread		4	3
Removed User from Thread		4	4
Added User to Template		4	5
Removed User from Template		4	6
Added User Group to Thread		4	7
Removed User Group from Thread		4	8
Created Template		5	1
Changed Template Settings		5	2
Changed Template Defaults		5	3
Viewed Attributes		6	1
Added Attribute		6	2
Deleted Attribute		6	3
Searched Workspaces		7	1
Viewed Meeting		8	1
Joined Meeting		8	2
Closed Meeting		8	3

Log Message	txlog.action	actionnum	subactionnum
Started Meeting		8	4
Created Meeting		8	5
Added External User to Meeting		8	6
Added OnBase User to Meeting		8	7
Meeting Organizer Assigned		8	8

CONFIGLOG

Log Message	txlog.action	actionnum	subactionnum
Scan Format Modified		1	1

DOCDISTLOG

Log Message	txlog.action	actionnum	subactionnum
Statement Generation	4096	1	9
Statement Mailed	4097	1	2
Statement Faxed	4098	1	3
Statement Published	4099	1	4
Statement Rendered	4100	1	5
Created Document Distribution Recipient	25	3	1
Deleted Document Distribution Recipient	67	3	2
Modified Document Distribution Recipient	130	3	3

LBDECISIONINGLOG

Log Message	txlog.action	actionnum	subactionnum
Lockbox Keyword Change		0	
Lockbox Transaction Status		1	
Lockbox Batch Submitted		2	
Lockbox Initial Email		3	
Lockbox Escalation Email		4	
Lockbox Cutoff Email		5	
Lockbox End of Day Email		6	

LOCKBOXLOG

Log Message	txlog.action	actionnum	subactionnum
IMS Lockbox User Logged In		1	1
IMS Lockbox User Logged Out		1	2
IMS Lockbox Document Search		1	3
IMS Lockbox Document Viewed		1	4
IMS Lockbox Batch Search		1	5
IMS Lockbox Deposit Search		1	6
IMS Lockbox Pocket Search		1	7
IMS Lockbox Report Search		1	8
IMS Lockbox Report Document Viewed		1	9
IMS Lockbox User Group Created		2	1
IMS Lockbox User Group Privileges Modified		2	2
IMS Lockbox User Group Name Modified		2	3
IMS Lockbox User Group Deleted		2	4
IMS Lockbox Assigned to User Group		2	5
IMS Lockbox Notification Created		2	6
IMS Lockbox Notification Modified		2	7
IMS Lockbox Notification Deleted		2	8
IMS Lockbox User Created		3	1
IMS Lockbox User Deleted		3	2
IMS Lockbox Password Modified		3	3
IMS Lockbox Admin Password Modified		3	4
IMS Lockbox User Name Modified		3	5
IMS Lockbox Display Name Modified		3	6
IMS Lockbox User Email Modified		3	7
IMS Lockbox User to Group		3	8
IMS Lockbox Group to User		3	9
IMS Lockbox Unlock Disabled User		3	10

PLTRMGMTLOG

Log Message	txlog.action	actionnum	subactionnum
Critical Platter Management Error	8193	1	1
Platter Management Failure	8194	1	2
Platter Management Failure with Confirm	8195	1	3
Platter Management Warning	8196	1	4
Platter Management Warning with Confirm	8197	1	5
Platter Management Confirm	8198	1	6
Platter Management Commit Failure		1	7
Platter Management	8192	2	1
Platter Deleted		2	2
Platter Analyzed		2	3
Platter Copied		2	4
Platter Moved		2	5
Platter Backed Up		2	6
Platter Exported		2	7
Platter Automatically Backed Up		2	8
Platter Automatically Exported		2	9
Platter Automatically Deleted		2	10
Platter Automatically Analyzed		2	11
Prompt for Platter Mount		2	12

PROCESSINGLOG

Log Message	txlog.action	actionnum	subactionnum
Committed Batch	65536	1	1
Purged Data	65537	1	2
Archived Data	65538	1	3
Batch OCR'ed	65539	1	4
Indices Extracted from a Batch	65540	1	5
Batch Re-dated	65541	1	6
Batch Full Text Indexed	65542	1	7
Daily Report Cleared	65543	1	8
Batch Renamed	65545	1	9
Processed Format	2097152	1	10
Created Batch	20	1	11
Auto Folder Documents		1	12
Batch Zonal OCR		1	13
Ran an All Items Process	2097153	2	1
Ran a Repass Process	2097154	2	2
Purged the Daily and All Items Table	2097155	2	3
Executed Autofill Keyset Importer	32770	3	1
Unable to Create Batch or Verification Report		3	2
Failed to Create Lockbox DCN File		4	1

RIMLOG

Log Message	txlog.action	actionnum	subactionnum
Open Event Posted		1	1
Close Event Posted		1	2
Cutoff Event Posted		1	3
Managed Folder Created		2	1
Folder Opened		2	2
Folder Closed		2	3
Folder Cut Off		2	4
Folder Transitioned to Awaiting Approval		2	5
Folder Destroyed		2	6
Folder Purged		2	7
Folder Retained		2	8
Folder Deleted		2	9
Hold Placed		3	1
Hold Removed		3	2
Approved for Final Disposition		4	1
Retention Plan Changed		4	2
Cannot acquire Time Stamp Digital Signature License		5	1
Keyword Added to Document		6	1
Keyword Removed from Document		6	2
Keyword Added to Folder		6	3
Keyword Removed from Folder		6	4
Document Added		7	1
Document Removed		7	2
Document Removed/Folder Deleted		7	3

ROIREQUESTLOG

Log Message	txlog.action	actionnum	subactionnum
ROI Request Created		1	1
ROI Request Deleted		1	2
ROI Request Viewed		1	3
ROI Document Added		2	1
ROI Document Removed		2	2
ROI Print Bill		3	1
ROI Print Request		3	2
ROI Export Request		3	3
ROI Export Request - Condre		3	4
ROI Export Request - Rimage		3	5
ROI Fax Request		3	6
ROI Forward Request		3	7
ROI Mail Request		3	8
ROI Mail Request - Express		3	9
ROI Mail Request - Office		3	10

SECURITYLOG

Log Message	txlog.action	actionnum	subactionnum
Logged into Client Module	131072	1	1
Logged into Configuration Module	131073	1	2
Logged into Print Server	131076	1	3
Logged into Search Server	131077	1	4
Logged into Enterprise Server Communication	131081	1	5
Logged into OBAPI	131082	1	6
Login Failure	131083	1	7
License Overage	131086	1	8
Logged into Core Services		1	9
Logged into Disconnected Scanning		1	10
Logged into Front Office Scanning		1	11
The maximum number of API queries exceeded	32775	2	1
API cannot acquire concurrent license	131084	2	2
API cannot acquire named license	131085	2	3
Out of Email Archiver Licenses		2	4
Out of Workstation Licenses		2	5
Logged out of Client Module	262144	3	1
Logged out of Configuration Module	262145	3	2
Logged out of Print Server	262148	3	3
Logged out of Search Server	262149	3	4
Logged out of Enterprise Server Communication	262153	3	5
Logged out of OBAPI	262154	3	6
Logged out of Core Services		3	7
Changed User Privileges	524289	4	1
Changed User Password	524290	4	2
Changed User Products	524291	4	3
Updated Product Licenses	524292	4	4
Validated Product Licenses	524293	4	5
Created New User	524294	4	6
Deleted User	524295	4	7
Global License Agreement Accepted	524296	4	8

Log Message	txlog.action	actionnum	subactionnum
Envelope Shared		4	9
Protected Life Cycle		4	10

SYSTEMLOG

Log Message	txlog.action	actionnum	subactionnum
Automated CD Server Connection Attempt	32769	1	1
Updated Database Version	524288	2	1
Database Backup	33554432	2	2
Enterprise Server Communication Message	1048576	2	3
Email Archiver Start up		3	1
Email Archiver Shut Down		3	2
API Core Session Script		4	1
API Core Document Script		4	2
Copy to Secondary Disk Group(s) Failed		4	3
Archived Document Not Added to SAP – No Document Type Mapping		5	1
Autofill Process Updated Document		6	1
Keyword Merge		6	2

TRANSACTIONXLOG

Log Message	txlog.action	actionnum	subactionnum
Created Document	16	1	1
Created Document for Existing	24	1	2
Deleted Document	64	1	3
Deleted Un-indexed Document	70	1	4
Undeleted Document	2052	1	5
List Content Report Created	16384	1	6
Search Result Report Created	16385	1	7
Document Destroyed by Records Management		1	8
Created Folder	17	2	1
Viewed Folder	33	2	2
Viewed Folder Keywords	36	2	3
Deleted Folder	65	2	4
Added an item to a Folder	512	2	5
Removed an Item from a Folder	513	2	6
Added an Item to a Managed Folder	514	2	7
Modify Folder Keywords		2	8
Viewed Folder and Children		2	9
Failed to Update Child Folder Keywords		2	10
Folder Purged		2	11
Folder Moved		2	12
Folder Copied		2	13
Folder Keyword Deleted		2	14
Folder Keyword Added		2	15
Folder Modified Date		2	16
Folder Undeleted		2	17
Created Note	18	3	1
Viewed Note	34	3	2
Deleted Note	66	3	3
Modified Note	128	3	4
Unity Note Retrieved		3	5
Viewed Document	32	4	1
Printed a Document	1025	4	2
Mailed a Document	1026	4	3

Log Message	txlog.action	actionnum	subactionnum
Document Re-indexed	1036	4	4
Checked Out Document	2049	4	5
Checked In Document	2050	4	6
Cancelled Check Out	2051	4	7
Created Document Revision	21	4	8
Created Document Rendition	22	4	9
Deleted Revision	68	4	10
Burn Redaction Bitmaps	2055	4	11
Unique Barcode Match – Page appended	2056	4	12
Dropped Page into Document	1030	4	13
Dragged Page from Document	1031	4	14
Document Rotation Saved	1038	4	15
Added Page to Document	2062	4	16
Modify Document Date	1039	4	17
Modified Document FileType	2057	4	18
Document Password	1028	4	19
Document Version Overwritten		4	22
Append Imported Image		4	23
Document Field Modified		4	24
Unity API Retrieved Document		4	25
Viewed Document Keywords	35	5	1
Keyword Update – Doc Type	1032	5	3
Keyword Update – Global	1033	5	4
Deleted Keyword	1034	5	5
Deleted Document Keyword	1034	5	5
Add Keyword	1035	5	6
Add Document Keyword	1035	5	6
Unity API Retrieved Keyword		5	7
Created Document Redaction	23	6	1
Exported a Document	1027	6	2
Copied to Clipboard	2048	6	3
Added Electronic Signature	2053	6	4
Removed Electronic Signature	2054	6	5
Document Full Text Indexed	1037	6	6
Document added to Verity Full Text	2059	6	7

Log Message	txlog.action	actionnum	subactionnum
Document Removed for Verity Full Text	2060	6	8
Document OCR'ed	2061	6	9
Added Reference Document	8388608	6	10
Added to workspace	8388609	6	11
Removed from workspace	8388610	6	12
Added Post	8388611	6	13
Pages Copied to new Document	2058	6	14
Removed an item for Document Retention	67108865	6	15
Put item back into Document Retention	67108866	6	16
Digital Signatures Activity		6	17
DKT Administrator Acknowledgement		6	18
Verify Digital Signature		6	19
Add Time Stamp Digital Signature		6	20
Verify Time Stamp Digital Signature		6	21
Export Document for External Verification		6	22
Time Stamp Error		6	23
Remove DKT User Reading Requirement		6	24
Document Retention Re-index		6	25
Document Retention Delete		6	26
Cannot acquire Time Stamp Digital Signature License		6	27
Cannot acquire Time Stamp Digital Signature License		6	28
Viewed Document via Data Mining		6	29
Extracted Document via Data Mining		6	30
Printed Document via Data Mining		6	31
Encryption		6	33
Printed Substitute Check	2063	7	1
Physical Record Changed Repository		8	1
Physical Record Changed Location		8	2
Physical Record Checked In		8	3
Physical Record Checked Out		8	4
Physical Record Copied from Repository		8	5
Changed Check Out Location of Physical Record		8	6
Volume Tracking Locator Modified		9	1
Volume Tracking Locator Home Modified		9	2

Log Message	txlog.action	actionnum	subactionnum
Volume Tracking Locator Patient Modified		9	3
Volume Tracking Locator Chart Linked		9	4
Volume Tracking Locator Chart Unlinked		9	5

WFUSEREVENTS

Log Message	txlog.action	actionnum	subactionnum
User Open Workflow Inbox	32771	1	1

Appendix B

MEDICAL RECORDS ENUMERATION VALUES

Several tables within the Medical Records schema use enumeration values to indicate a status, activity type, or other setting. The tables below document these enumeration values.

CHART

The **CHART** table contains a **chtstatus** column which indicates the status of the chart's progression through the Chart Completion process. The table below lists all possible **chtstatus** values for the **CHART** table.

Chart Status (CHART table)	chtstatus Value
Has not yet begun processing	0
Waiting for Chart E-Form Creation	1
Needs Review (Charts Requiring Review)	5
In Coding (Chart requires both Analysis and Coding but has only progressed to Coding)	10
In Analysis (Chart requires both Analysis and Coding but has only progressed to Analysis)	20
Open (Chart has progressed to all appropriate contexts)	30
Inactive (Chart has met requirements to be marked Inactive)	40
Stalled Unknown (Chart went inactive prior to entering Coding or Analysis)	50
Stalled Coding (Chart went inactive prior to entering Analysis)	60
Stalled Analysis (Chart went inactive prior to entering Coding)	70
Admin Override Coding	80
Admin Override Analysis	90
Closed (Chart has met requirements for Closure)	100

The **CHART** table contains a **needsreviewcode** column which indicates one or multiple reasons why a chart is in the Charts Requiring Review queue. Although it does store multiple values, the value in this column is not a true bitflag. Translation of the value can be accomplished by the following:

Example 1:

needsreviewcode = 12 indicates the chart has a new admit type.

Example 2:

needsreviewcode = 854794 indicates that the chart has no attending physician, no admit type and no facility.

854794 (decimal) = D0B0A (hex)

Read the hex from right to left, in 2 character pairs, to find the values:

0A = 10, No Attending Physician

0B = 11, No Admit Type

D = 13, No Facility

Review Codes (CHART table)	Value
None	0
No MRN	1
No ChartID	2
Different SSN (Chart SSN must match MedRec SSN)	3
Missing SSN	4
No MPI	5
Duplicate ChartID	6
Bad Admit Date	7
Bad Discharge Date	8
Bad Birth Date	9
No Attending Physician	10
No Admit Type	11
New (unconfigured) Admit Type	12
No Facility	13
New (unconfigured) Facility	14

The **CHART** table contains a **decisioning** column, which indicates the chart's progression through Coding and/or Analysis based on the Admit Type configuration at the time the chart was created. The value of the **decisioning** column is a bitflag that can contain multiple values. Proper bitwise syntax should be used when querying this column. The following table lists the possible values for **decisioning.**

Decisioning (CHART table)	Value
Not Yet Determined	0
No Routing (Neither Coding nor Analysis required)	1
Coding After Discharge	2
Analysis After Discharge	4
Coding On Admission	8
Analysis On Admission	16

The **CHART** table contains a **mrcontrolsys** column, which indicates the application responsible for processing of the chart. The following table lists the possible values for the **mrcontrolsys** column.

Controling System (CHART table)	**Value**
No processing	0
Medical Records Management Solution	205
Signature Deficiencies for Epic	303

ADMITTYPE

The **ADMITTYPE** table contains a **flags** column, which stores information about the Admit Type's configuration for Coding and/or Analysis. The value of the **flags** column is a bitflag that can contain multiple values. Proper bitwise syntax should be used when querying this column. The following table lists the possible values for **flags**.

Admit Type Flags (ADMITTYPE table)	Value
Requires Coding after Discharge	1
Requires Analysis after Discharge	2
Requires Coding after Admit	4
Requires Analysis after Admit	8
Unconfigured (HL7 automatically created the Admit Type)	4096

CHARTANALYSIS

The **CHARTANALYSIS** table contains a **chtstatus** column that is used to determine the current status of a chart's progression through the individual queues of the Medical Records Management process. Multiple rows can exist for a given chart, each with a different and potentially unrelated **chtstatus**. The following table lists the possible values for the **chtstatus** column in the **CHARTANALYSIS** table.

Chart Status (CHARTANALYSIS table)	Value
In Coding	2
Post Coding	6
Coding Complete	8
In Analysis	12
Post Analysis	16
Analysis Complete	18
In Completion (Prior to Analysis Complete)	-22
In Completion (After Analysis Complete)	22
Post Completion (Prior to Analysis Complete)	-24
Post Completion (After Analysis Complete)	24
In Reanalysis	32
Reanalysis In Progress	33
Post Reanalysis	34
Preanalysis Complete	-40
Reanalysis Complete	40
Closed	50
QA Coding	102
Post QA Coding	106
QA Coding Completed	108
QA Analysis	112
Post QA Analysis	116
QA Analysis Completed	118
QA Reanalysis	132
Post QA Reanalysis	134
QA Reanalysis Completed	138

DOCDEFICIENCY / CHARTDEFICIENCY

The **DOCDEFICIENCY** and **CHARTDEFICIENCY** tables contain a **dfcytype** column, which indicates the specific type of deficiency. The following table indicates possible values for this column along with the deficiency level (document or chart) indicating which table would contain the value.

Deficiency Type (DOCDEFICIENCY/CHARTDEFICIENCY table)	**Level**	**Value**
Missing Signature	Document	1
Missing Other	Document	3
Missing Document	Chart	4
Missing Diagnosis	Document	5
Missing Dictation	Chart	6
Incomplete Form / Physician Query	Document	7
Missing Information	Document	8
Editable Transcription	Document	9
Missing Dual Signature	Document	10
Missing Dual Dictation	Chart	12
External Missing Signature	Chart	13
External Unsigned Order	Chart	14
External Missing Information	Chart	15

DFCYTXLOG

The **DFCYTXLOG** table stores a transaction log of activity that occurred on the specified chart. This table contains an **action** column, which indicates the specific activity being logged as well as **extrainfo1**, **extrainfo2**, and **extrainfo3** columns, which provide additional detail into the logged activity. The following table lists possible values for the **action** column along with a definition of the corresponding **extrainfo** values, if applicable.

Activity (DFCYTXLOG table)	Action	Extrainfo1	Extrainfo2	Extrainfo3
Chart Created	1			
Chart Entered Coding	2			
Chart Suspended In Coding	4			
Chart Unsuspended In Coding	5			
Chart Coding Completed	8			
Chart Entered Analysis	12			
Chart Suspended	14			
Chart Unsuspended	15			
Chart Analysis Completed	18			
Chart Entered Completion	22	physnum		
Chart Completion Completed	28	physnum		
Chart Entered Reanalysis	32	physnum		
Chart Reanalysis Completed	38	physnum		
Chart Waiting For Closure	40			
Chart Completed	50			
Chart Viewed	60			
Chart Printed	70			
Chart Reviewed For HL7	80			
Chart Modified	90			
Merge MPI	91			
Merge MRN	92			
Merge Chart ID	93	oldchtnum		
Closed Chart Modified	94			
Admit Type Changed	95	old admittypenum	new admittypenum	
Document Added To Chart	100			
Document Removed From Chart	101			
Chart Deleted	102			
Chart Delete Failed	103			
Chart Deleted Manually	104			
External Mail Sent	105	usernum	recipient usernum	
Internal Mail Sent	106	usernum	recipient usernum	
Admin Unassigned Analyst	110			
Admin Reassigned Analyst	111			
Admin Unassigned Reanalyst	112			
Admin Reassigned Reanalyst	113			
Admin Unassigned Coder	114			
Admin Reassigned Coder	115			
Admin Reassigned Physician	116			
Chart Tab Printed	117	uitabnum		
Chart Template Printed	118	chtprinttmplnum		
Physician Query Created	119	physician usernum		
Physician Query Edited	120	old physician usernum	new physician usernum	
Physician Query Deleted	121	physician usernum		
Physician Query Sent	122	physician usernum		

Activity (DFCYTXLOG table)	Action	Extrainfo1	Extrainfo2	Extrainfo3
Deficiency Created	123	physician usernum	secondary signer usernum	dfcytype
Deficiency Modified	124			
Deficiency Deleted	125	physician usernum	secondary signer usernum	dfcytype
Deficiency Accepted	126			
Deficiency Rejected	127			
Deficiency Resubmitted	128			
Reassigned Primary Signer	129	old usernum	new usernum	
Reassigned Secondary Signer	130	old usernum	new usernum	
Confirmed Deficiency	131			
Closed Chart Opened	132			
Analyst Review Note Completed	133	codingnotenum		
Analyst Review Note Created	134			
Deficiency Created	135	physician usernum		
Deficiency Deleted	136	physician usernum		
Analyst Review Note Assigned	137	codingnotenum		
Analyst Review Note Unassigned	138	codingnotenum		
Analyst Review Notes Viewed	139			
Auto Confirmed Deficiency	140			
Deficiency Merged	141	physician usernum	secondary signer usernum	dfcytype
Dual Deficiency Merged	142	physician usernum	secondary signer usernum	dfcytype
Document Merged	143	old chtnum		
Deficiency Marked Stat	144			
Deficiency Stat Cleared	145			

Keyword values for examples below:

Keytypenum	Keytype	Additional Information
101	Last Name	Dual table alpha: hsi.keytable101, hsi.keyxitem101
102	First Name	Dual table alpha: hsi.keytable102, hsi.keyxitem102
105	Acct. #	Dual table alpha: hsi.keytable105, hsi.keyxitem105
129	WF Task Date	This is a date, single table: hsi.keyitem129

Query to verify the internal Document Type unique ID for 3 Document Types:

```
select * from hsi.doctype where itemtypenum in (101,102,103)
```

Query to list all documents (displays Auto-Name string only) for Document Type # 101:

```
select itemname from hsi.itemdata where itemtypenum=101
```

Query to list all documents for Document Type # 103 with an October Document Date:

```
select itemname from hsi.itemdata
where itemtypenum=103
and itemdate between '20081001' and '20081031'
```

Query to list all documents for Document Type # 103 with a November Document Date (also display First Name and Last Name Keyword values):

```
select i.itemname, kt1.keyvaluechar, kt2.keyvaluechar
from hsi.itemdata i, hsi.keytable101 kt1, hsi.keytable102 kt2,
hsi.keyxitem101 kx1, hsi.keyxitem102 kx2
where i.itemtypenum=103
```

```
and i.itemnum=kx1.itemnum
and i.itemnum=kx2.itemnum
and kx1.keywordnum=kt1.keywordnum
and kx2.keywordnum=kt2.keywordnum
and i.itemdate between '20081101' and '20081130'
```

Query to list all documents for Document Type # 103 with a December Document Date that exist in Workflow queue # 110 (also display First Name, Last Name, and WF Task Date keyword values):

```
select itemname, kt1.keyvaluechar, kt2.keyvaluechar, k3.keyvaluedate
from hsi.itemdata i, hsi.itemlc il, hsi.keytable101 kt1, hsi.keytable102 kt2,
hsi.keyxitem101 kx1, hsi.keyxitem102 kx2, hsi.keyitem129 k3
where i.itemtypenum=103
and i.itemnum=il.itemnum
and i.itemnum=kx1.itemnum
and i.itemnum=kx2.itemnum
and i.itemnum=k3.itemnum
and kx1.keywordnum=kt1.keywordnum
and kx2.keywordnum=kt2.keywordnum
and i.itemdate between '20081201' and '20081231'
and il.statenum=110
```

Query to list all documents for Document Type # 101 that do not have an ACCT# (also displays First Name and Last Name; for SQL Server only):

```
select i.itemname, kt1.keyvaluechar, kt2.keyvaluechar
from hsi.keytable101 kt1, hsi.keytable102 kt2, hsi.keyxitem101 kx1,
hsi.keyxitem102 kx2, hsi.itemdata i
left outer join hsi.keyxitem105 kx5 on kx5.itemnum=i.itemnum
where i.itemtypenum=101
and i.itemnum=kx1.itemnum
and i.itemnum=kx2.itemnum
and kx1.keywordnum=kt1.keywordnum
and kx2.keywordnum=kt2.keywordnum
and kx5.itemnum is null
```

Writing Customized Custom Queries Within OnBase

A customized Custom Query provides the most control over the retrieval of documents within OnBase by allowing the user to code the actual SQL for a query. This option requires extensive knowledge of the system and must be configured correctly or the OnBase client will issue an ODBC error.

Caution: Badly constructed Custom Queries can adversely affect system performance and may also result in data loss or data corruption. It is recommended that only a database administrator set up a customized query and only after first consulting OnBase technical support.

To configure a database query in OnBase, launch the Configuration module and select **Custom Queries** from the **Queries** menu.

1. At the **Custom Query** dialog box, enter the name of a new Custom Query and click **Create**, or select an existing Custom Query from the list and click **Settings**.
2. At the **Custom Query Options** dialog box, select **Custom Written SQL**, then click **Edit SQL**. The **Customized Custom Query** dialog box is displayed, allowing the user to set the database query options.
3. In the **From Clause** field, enter a comma-separated list of the table names to use in the query (table aliases are not supported for the itemdata table).

   Note: The **itemdata** table must be the first table in the list.
4. In the **Where Clause** field, enter the specific search criteria for the query. This can include joins to tables given in the **From Clause**, as well as specific column values for the table.
5. In the **Order By Clause** field, enter the criteria that defines the sort order of the results, such as **datestored** (date stored), or **[column name] asc** (ascending), or **[column name] desc (descending)**, where **[column name]** is the column used for sorting.

Tip: For more information on Custom uQeries, see **Configuring Custom Queries** in the Configuration module help files or the System Administration module reference guide.

From/Where Clause Syntax Examples

From Clause

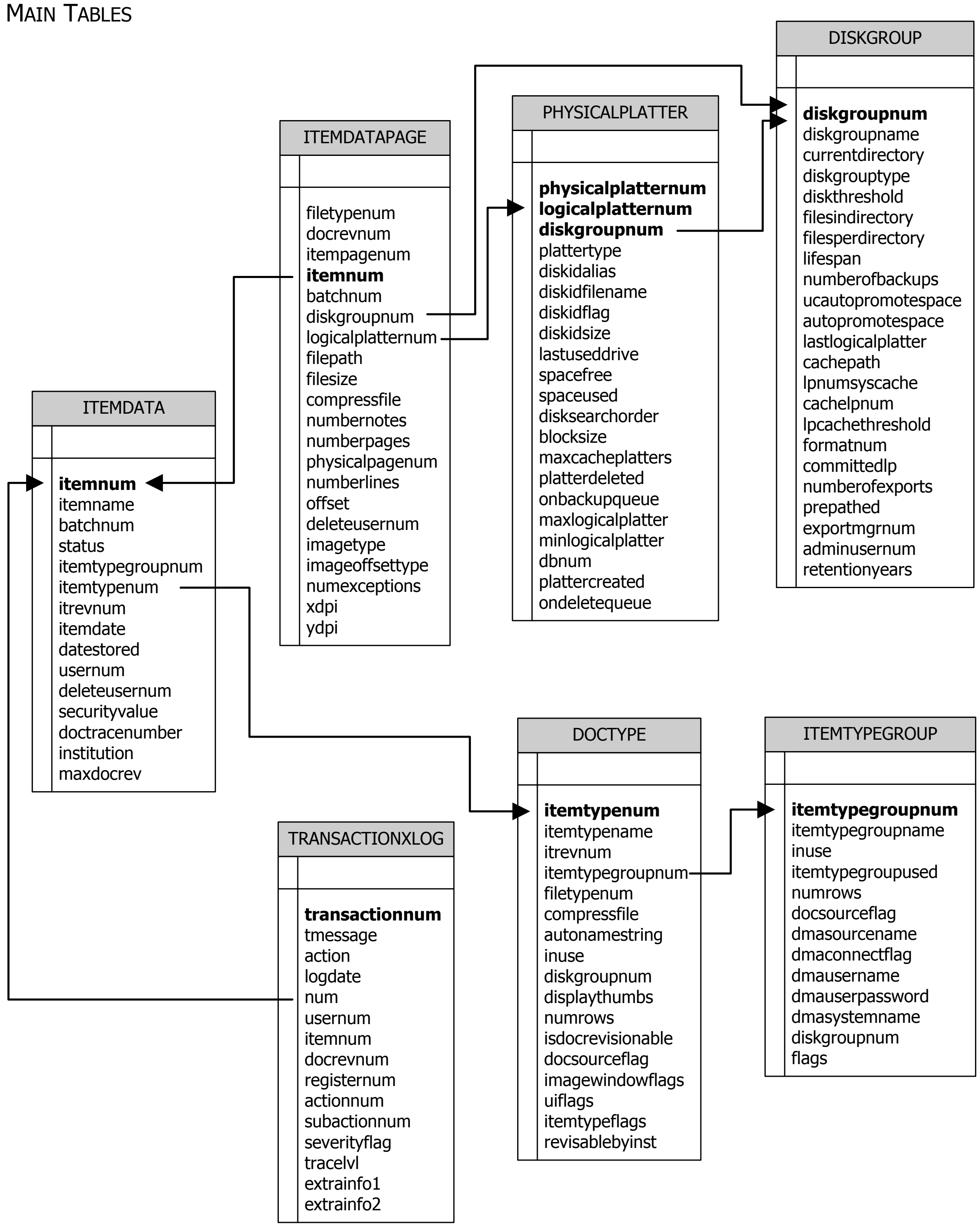
```
hsi.itemdata, hsi.keyxitem113, hsi.keytable113
```

Where Clause

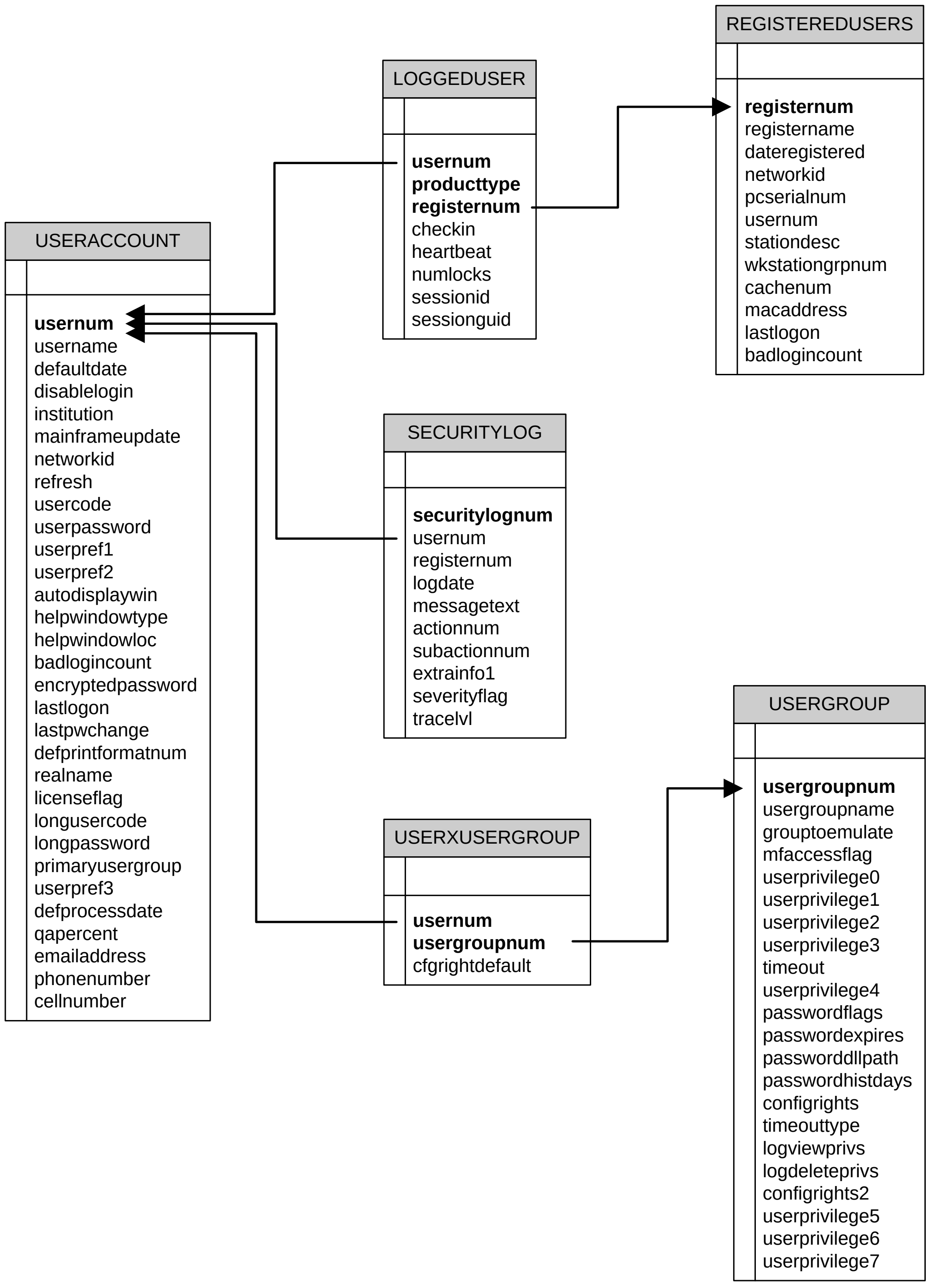
```
hsi.itemdata.itemnum= ki.itemnum

and hsi.keyxitem113.keywordnum= hsi.keytable113.keywordnum

and substring(hsi.keytable113.keyvaluechar,1,1)= 'A'

and hsi.itemdata.itemtypenum = 101 and hsi.itemdata.status + 0 = 0
```

Database Diagrams

MAIN TABLES



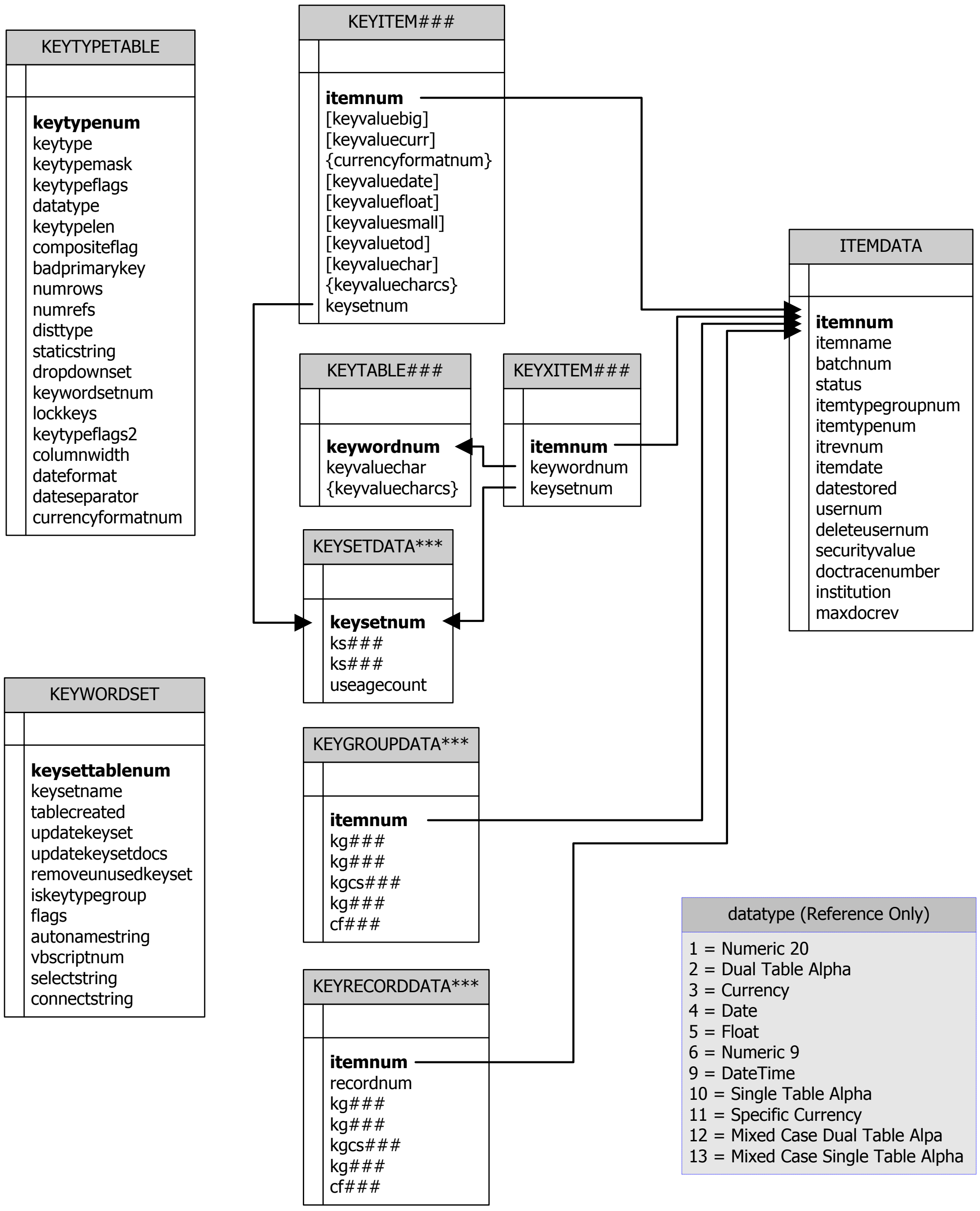
User Tables



REGISTEREDUSERS
registernum
registername
dateregistered
networkid
pcserialnum
usernum
stationdesc
wkstationgrpnum
cachenum
macaddress
lastlogon
badlogincount
LOGGEDUSER
usernum
producttype
registernum
checkin
heartbeat
numlocks
sessionid
sessionguid
USERACCOUNT
usernum
username
defaultdate
disablelogin
institution
mainframeupdate
networkid
refresh
usercode
userpassword
userpref1
userpref2
autodisplaywin
helpwindowtype
helpwindowloc
badlogincount
encryptedpassword
lastlogon
lastpwchange
defprintformatnum
realname
licenseflag
longusercode
longpassword
primaryusergroup
userpref3
defprocessdate
qapercent
emailaddress
phonenumber
cellnumber
SECURITYLOG
securitylognum
usernum
registernum
logdate
messagetext
actionnum
subactionnum
extrainfo1
severityflag
tracelvl
USERXUSERGROUP
usernum
usergroupnum
cfgrightdefault
USERGROUP
usergroupnum
usergroupname
grouptoemulate
mfaccessflag
userprivilege0
userprivilege1
userprivilege2
userprivilege3
timeout
userprivilege4
passwordflags
passwordexpires
passworddllpath
passwordhistdays
configrights
timeouttype
logviewprivs
logdeleteprivs
configrights2
userprivilege5
userprivilege6
userprivilege7

KEYWORD TABLES

IS THE ID (KEYTYPENUM) OF THE KEYWORD, FROM KEYTYPETABLE
*** IS THE ID (KEYSETTABLENUM) OF THE KEYSET OR KEYGROUP, FROM KEYWORDSET



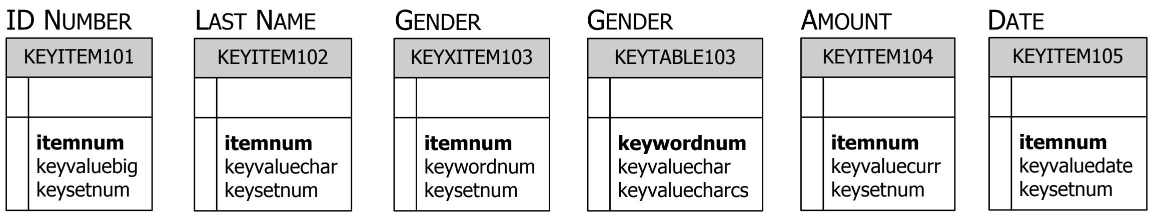
datatype (Reference Only)
1 = Numeric 20
2 = Dual Table Alpha
3 = Currency
4 = Date
5 = Float
6 = Numeric 9
9 = DateTime
10 = Single Table Alpha
11 = Specific Currency
12 = Mixed Case Dual Table Alpa
13 = Mixed Case Single Table Alpha

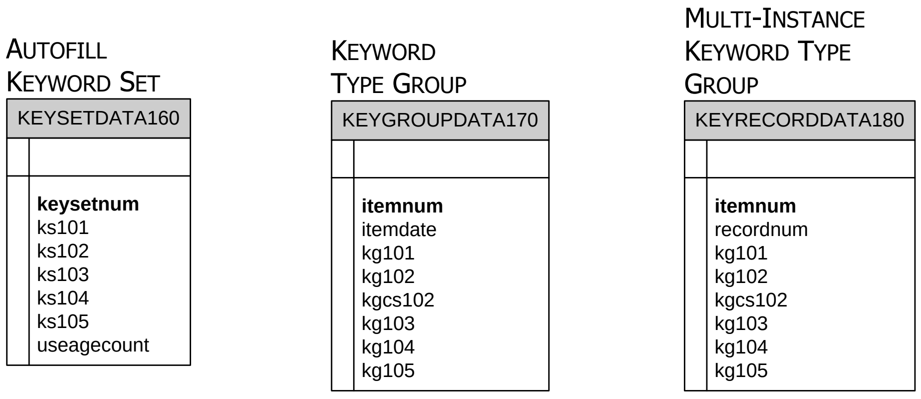
Keyword Tables - Example

This example is designed to demonstrate the name and design of the keyword tables for a sample configuration. For this example, the following keywords were created via configuration:

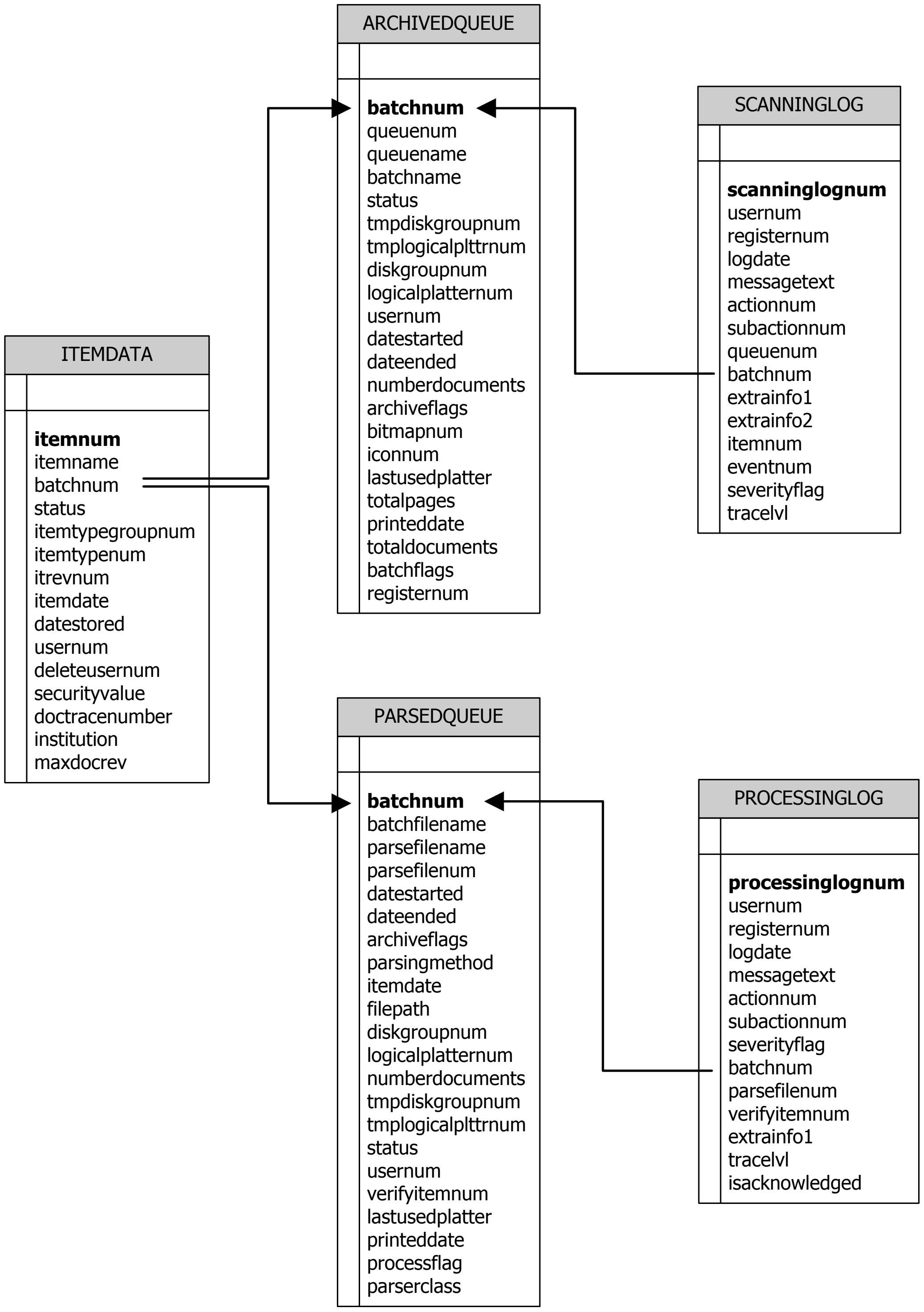
ID Number: Numeric 20 – OnBase ID 101
Last Name: Single Table Alphanumeric Mixed Case – OnBase ID 102
Gender: Dual Table Alphanumeric – OnBase ID 103
Amount: Currency – OnBase ID 104
Date of Service: Date – OnBase ID 105

For this group of keywords, an Autofill Keyset (OnBase ID 160), a Keyword Type Group (OnBase ID 170) which includes itemdate, and a Multi-Instance Keyword Type Group (OnBase ID 180) were created to show each table structure.

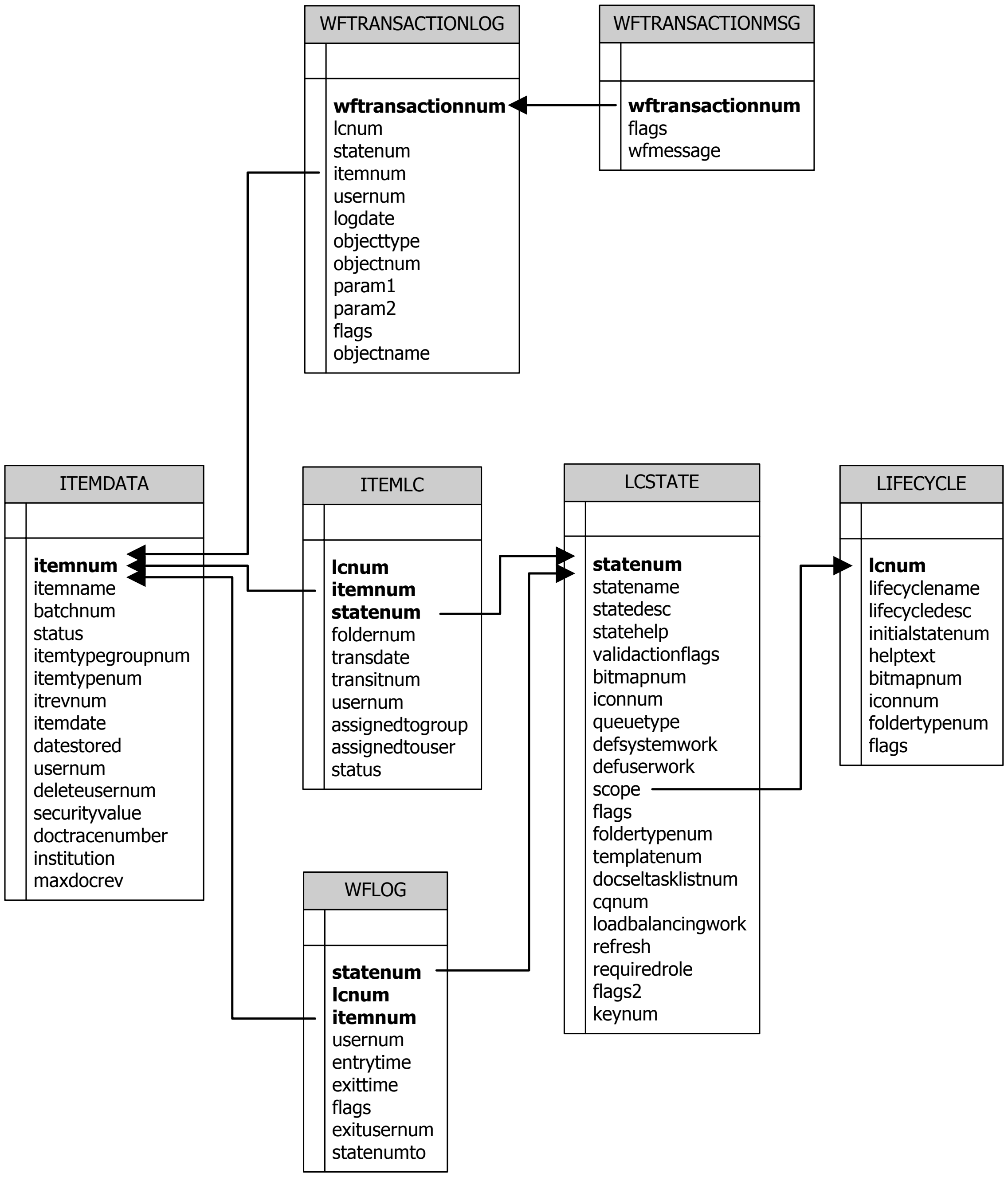




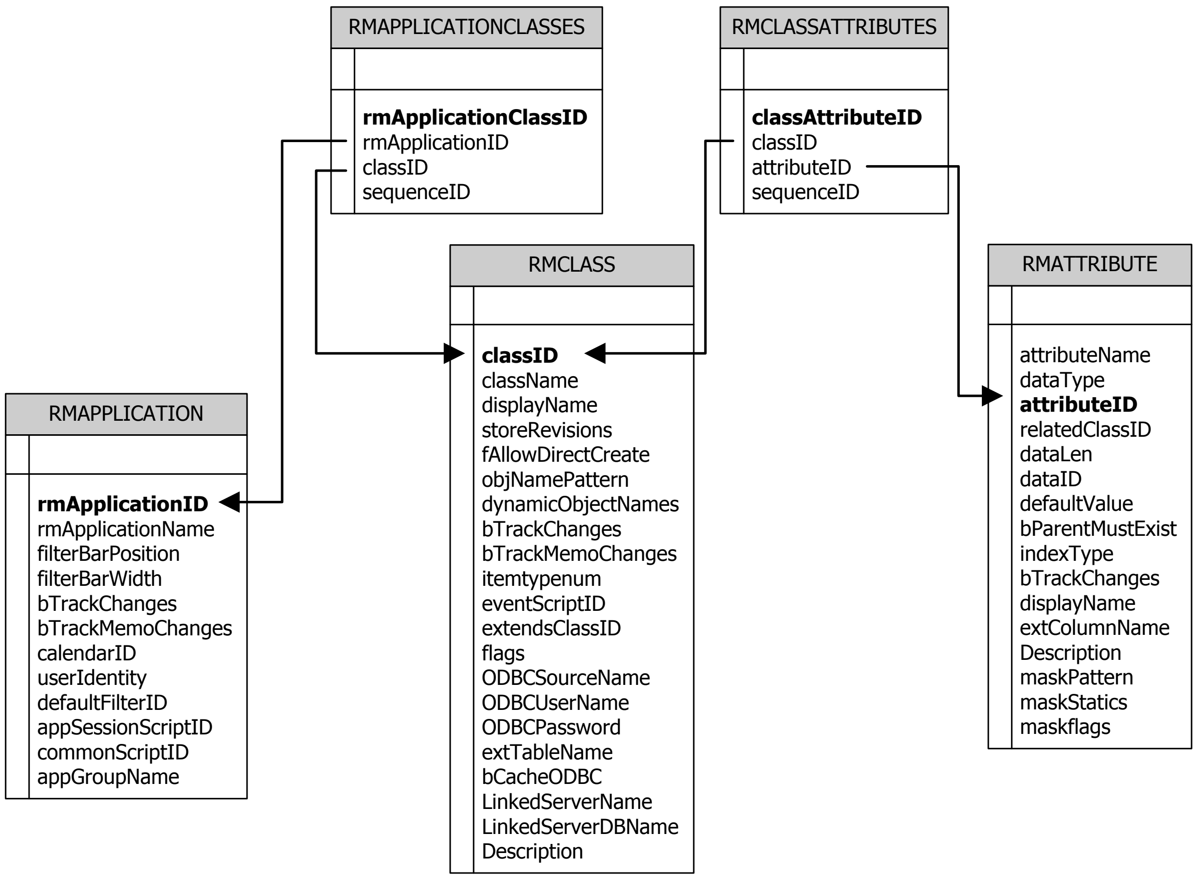
BATCH TABLES

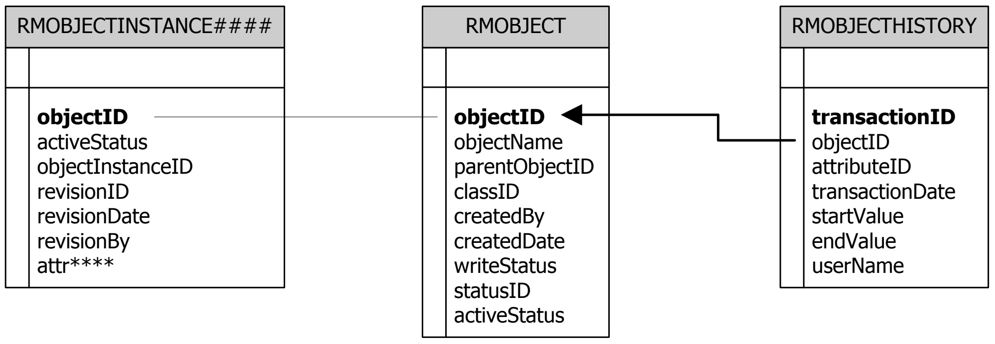


Workflow Tables



WorkView Tables


RMAPPLICATIONCLASSES
rmApplicationClassID
rmApplicationID
classID
sequenceID
RMCLASSATTRIBUTES
classAttributeID
classID
attributeID
sequenceID
RMCLASS
classID
className
displayName
storeRevisions
fAllowDirectCreate
objNamePattern
dynamicObjectNames
bTrackChanges
bTrackMemoChanges
itemtypenum
eventScriptID
extendsClassID
flags
ODBCSourceName
ODBCUserName
ODBCPassword
extTableName
bCacheODBC
LinkedServerName
LinkedServerDBName
Description
RMATTRIBUTE
attributeName
dataType
attributeID
relatedClassID
dataLen
dataID
defaultValue
bParentMustExist
indexType
bTrackChanges
displayName
extColumnName
Description
maskPattern
maskStatics
maskflags
RMAPPLICATION
rmApplicationID
rmApplicationName
filterBarPosition
filterBarWidth
bTrackChanges
bTrackMemoChanges
calendarID
userIdentity
defaultFilterID
appSessionScriptID
commonScriptID
appGroupName


RMOBJECTINSTANCE####
objectID
activeStatus
objectInstanceID
revisionID
revisionDate
revisionBy
attr****
RMOBJECT
objectID
objectName
parentObjectID
classID
createdBy
createdDate
writeStatus
statusID
activeStatus
RMOBJECTHISTORY
transactionID
objectID
attributeID
transactionDate
startValue
endValue
userName