Creating an ILLiad Transaction Archive

Janetta Waterhouse, Systems Librarian

1425 Jayhawk Blvd.
Lawrence, KS 66045

Table of Contents

Introduction

KU Libraries started with ILLiad[1] lending during October 2003 and borrowing and document delivery during February 2004. At the time this project was finally implemented, University of Kansas (KU) Libraries[2] had over 115,000 loan requests and nearly 160,000 article and document delivery requests, representing approximately 200,000 lending, 52,000 borrowing and 23,000 document delivery transactions. This ILLiad database is also shared with KU Medical Center (KUMC) Library, but the archive project is specific to the main campus in Lawrence.

In this technical paper I will describe the process we followed to create a transaction archive for KU Libraries' ILLiad database. At the time of this writing, we are at ILLiad version 7.2. I assume the reader has advanced user or administrative knowledge of his or her own institution's ILLiad software and related user data. I do not provide extensive details at the Microsoft SQL Server level, only referring to some actions that need to be performed on the server. I have included the SQL code I created for testing purposes as appendices. There are undoubtedly other ways, some of them perhaps more elegant, to achieve the same result with SQL. The code has been included here for reference; please feel free to use it with caution. I am happy to respond to questions or comments about this document, the code or the process described. Please direct them to me at jan@janwaterhouse.com.

The KU ILLiad configuration is non-standard in two ways. First, as mentioned, the Libraries share our locally managed server with KUMC. KU and KUMC Transactions and Users thus are in the same table but appear separately through views. The SQL code we used takes this into account, but I modified the code in the appendices of this document to be mostly generic. Second, KU Libraries has multiple branches and uses the NVTGC[3] code as the pickup location. Patrons select a pickup location upon first login, which modifies the value of the NVTGC field, and they can modify it and a few other user values with the "Change User Information" option on the menu.

---

[1] OCLC ILLiad is a software implementation of the interlibrary loan process. It began at Virginia Polytechnic Institute and State University, has been developed by Atlas Systems, Inc. (see http://www.atlas-sys.com/), and is now officially "OCLC ILLiad Resource Sharing Management Software". I will refer to it as "ILLiad".

[2] KU Libraries is treated as a singular, proper noun in this document.

[3] NVTGC, originally an acronym for Northern Virginia Tech Graduate Center, now refers to a pickup location or processing site for libraries that want to distinguish between locations. For details see http://www.atlas-sys.com/documentation/illiad/content/NVTGC.pdf

The transaction archive project

Unlike the Voyager integrated library system (ILS) used by KU Libraries, which archives circulation transactions once an item is discharged and immediately disassociates patron information, ILLiad retains interlibrary loan and document delivery transactions and their related user data indefinitely unless some extra action is taken. There are vendor procedures in place to remove transactions and to remove the username from transaction records. However, simply removing the transactions was not an acceptable solution for KU Libraries because we need to retain some requests for copyright information for three years and would like to retain other information related to transactions for bibliographic reports and workflow analysis indefinitely. Removing the username from the transaction was not acceptable either since that would also disassociate all patron information, and we particularly want to keep the patron department, status and pick-up location related to a transaction for reporting purposes. The solution proposed was to create a separate database without related patron information for archived transactions.

Lars Leon, Head of Access Services and Resource Sharing, proposed a project to archive non-patron transaction data and subsequently remove the transactions from the ILLiad database. He specified which data were needed and outlined criteria for selecting and deleting the records. The result of this project is a Microsoft SQL Server database, external to ILLiad, that allows us to pull together information from a variety of sources to maximize support for collection development as well as workflow analysis. It is available to selected staff for searching and generating reports using Microsoft Access.

Designing the archive

After clarifying some details for this project, I designed a simple database structure for an archive based on the table structure in the ILLiad database, ILLData (see Figure 1). The basic structure was determined by starting with a diagram provided by Atlas Systems titled "Key Table Relationships", available at http://www.atlas-sys.com/documentation/illiad/content/ILLiadDatabaseDiagram.pdf This Atlas document presents a simple, graphical representation of the main tables that we used to decide what should be included in the archive. It is noteworthy that the Atlas diagram is a version or update or two behind the current version. At the time of our implementation, there were some fields in the Transactions table that were not listed in the diagram. Figure 1 is a simple implementation of the initial test database that I created using Microsoft Access as a mockup. Note that most of the fields in the Transaction table are not included in the figure.

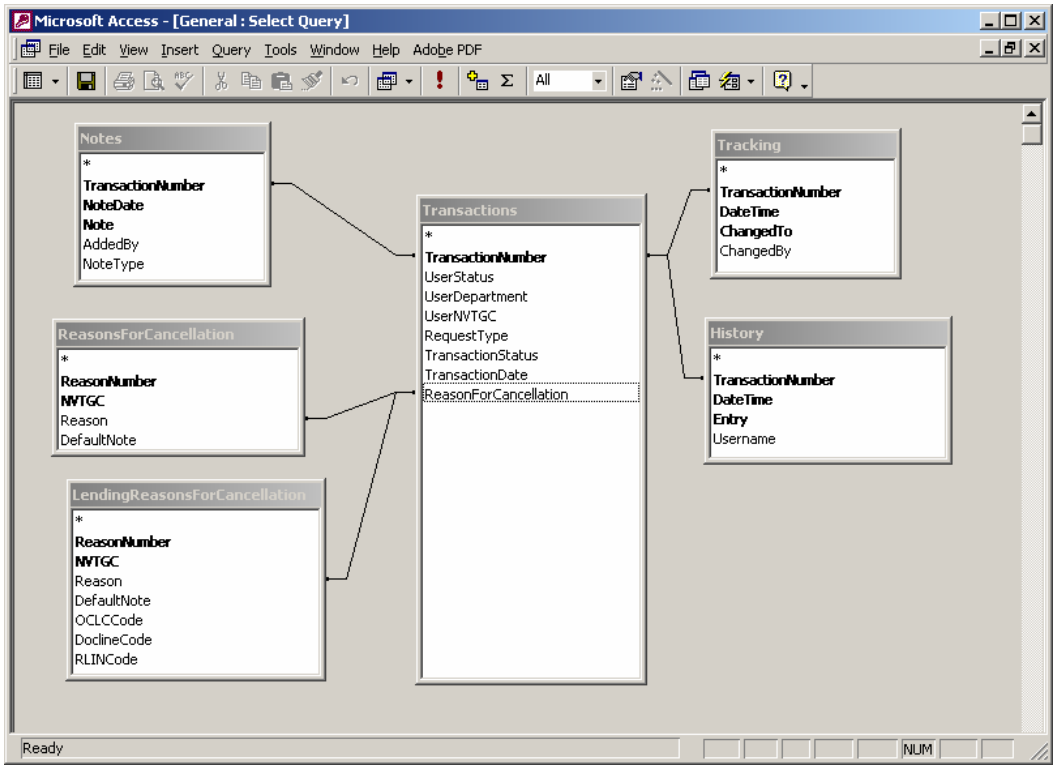

FIGURE 1 Basic relational diagram of the transaction archive

The archive database includes the following tables:

- ReasonsForCancellation and LendingReasonsForCancellation, which need to be populated once and updated only when reasons are added
- Tracking, History and Notes without modification
- Transactions with modification

In addition, the Status, Department, and NVTGC (pickup location) fields from the Users table are pulled into the Transactions table in the archive for reporting purposes. The Transactions table also has a date archived field.

Creating the archive database

See Appendix A for the SQL code used to create the database.

Creating the archive database was fairly straightforward. The SQL code in Appendix A used for creating the database was produced with Microsoft SQL Server Query Analyzer. Selecting the table of interest and right-clicking will give a set of options, one of which is "Script Object to New Window As → Create". This creates a window with the code to create that specific table. By selecting each table of interest, six total, and copying the create table codes into one file, all that remains is to add the initial line to create the database and code at the end to create an index. It is also

possible to create a script for the entire database within the Microsoft SQL Server Enterprise Manager. However, the additional code and method for creating primary keys seemed much more cumbersome.

Please note that the code in Appendix A has been created specifically for KU. A few Transaction table fields that are not used here were omitted in both the database creation and record selection code. As mentioned, three fields from the Users table are pulled into the archive Transactions table: Status, Department and NVTGC. Libraries with no special use for the NVTGC field will not need to include it in the archive.

Populating the archive

See Appendices B through E for the SQL code used to select transactions and related data for populating the archive.

Data for populating the archive database is captured with select statements that restrict by TransactionDate and a TransactionStatus of either "Request Finished" or "Cancelled by ILL Staff" and saved to a file that can be imported to the archive by someone with appropriate access to the Microsoft SQL Server. We added additional select criteria to restrict records to KU Libraries and not KUMC that is noted in the comments in Appendices C, D and E. Other libraries in a shared server environment will need to take this into account.

Considerations for selecting data:

1. ReasonsForCancellation and LendingReasonsForCancellation: This data needs to be selected from ILLData and imported into the archive once and then updated only when the reasons tables change. No code has been provided for this in the appendices. A simple SELECT * from [table_name] will capture all of the relevant data. Institutions such as KU with a shared server may need to add the criteria WHERE NVTGC != 'value'.

2. Transactions table: These records are selected partially and related fields are included. The Transactions fields Username and Patron will have patron data and should not be selected if patron privacy is to be maintained in the archive. Other fields unused at this institution are also not selected, but the Status, Department and NVTGC fields from the Users table are added to the transaction record.

3. Tracking, History and Notes tables: While it is possible to bring these tables into the archive in their entirety, they may contain user data. The fields [Tracking]Changedby, [History]Username, and [Notes]Addedby may contain a patron's username, system values or pertinent staff usernames useful for workflow analysis. In order to remedy the patron privacy issue, two versions of select were used for each table. One version compares the field values in question with acceptable values and selects the entire record. The other version selects records with patron data and replaces the

appropriate field, ChangedBy, Username or AddedBy, with the text "Patron" so that no user data is captured for the archive.[4]

Date range and field for selecting records

The transaction date range used in this code is for a specific period, "older than June 1, 2006", and thus must be changed manually to match the desired range before submitting the query. Once we have determined how often transactions will be automatically archived, this will be replaced with different code to select non-active transactions from more than x number of days ago. The date criteria would become WHERE (TransactionDate < GETDATE() – 30), which would ignore transactions from the past 30 days.

We have chosen to use the TransactionDate field in the Transactions table as the date to select records for archiving and deletion. While this date changes often during the course of an active transaction, it shouldn't change once a transaction's status has changed to 'Request Finished' or 'Cancelled by ILL Staff'. Rather than trawl related Tracking records for another date field to use, we have chosen to keep it simple and accept that this combination of status and date will be acceptable for our purposes.

How to access and search the archive

Creating and populating the archive all happens on the Microsoft SQL Server. Authorized library staff are able to connect to, search and create reports from the archive using Microsoft Access. In order to do this, an ODBC connection between the archive database on the Microsoft SQL Server and Microsoft Access on individual staff workstations must be created. The database system administrator will need to make sure that users have the appropriate access to the archive and can assist in setting up the Microsoft Access connection.

Note

Please use caution when using the SQL code included in this document, especially Appendix F which will delete records from the ILLData database.

[4] The code in Appendices D and E compare the Username field from the Staff table. Additional select criteria will need to be added for departed staff whose records have been removed from the Staff table via Customization Manager or else the code will replace a valid, departed staff id with the text "Patron".

Deleting data from the main database

See Appendix F for the SQL code to delete transactions from the ILLData database

One of the main reasons for the archive is to remove historical patron information. Once the archive was created and populated, the next step was to delete transactions and related records that had been archived from the main ILLiad database. Deleting transactions from the ILLData database needs to be done with care because the TransactionNumber, the primary key for the Transactions table, references records in several other tables. If the related records aren't removed from the secondary tables prior to being removed from the Transactions table, they will be 'orphaned' and difficult to find and remove. The "Table Fields" chapter of the ILLiad Administrative Suite: Reference Guide, located at http://www.atlas-sys.com/documentation/illiad/, lists all of the tables that use the TransactionNumber. We reviewed the local data before choosing to delete from the Notes, Tracking and History table prior to deleting from the Transactions table. It is also very important to use the same date range for selecting and deleting records. The delete code in Appendix F removes records from these secondary tables with a join based on TransactionNumber using the same date and status criteria as the select statements before deleting records from the Transactions table. It is also possible to delete records from ILLiad using the Database Manager. Contact OCLC support for assistance or check the ILLiad documentation[5] for more information.

Future directions for the archive

One remaining step is to have a clearly defined date range for selecting, archiving and deleting the records from the ILLiad database so that the process can be automated on the server and more current records will be available in the archive for reports. One unresolved issue related to this is identifying transactions with outstanding billing. To date we have only selected transactions that are over a year old and have not been concerned about selecting and subsequently removing transactions that have outstanding billing issues. Since KU Libraries does not currently use ILLiad's billing module, we will need to develop another method for not selecting these transactions before we begin archiving and deleting transactions from the recent past.

Removing transactions on a regular basis will help with the maintenance of patron records. KU Libraries is just beginning to pre-register patrons by importing records into the Users table. Since we will not remove patrons that are either currently active at KU or have related records in the transactions table, more inactive patrons can be removed as transactions are archived and removed from the main database.

[5] For OCLC support see http://www.oclc.org/support/default.htm For Atlas ILLiad documentation see http://www.atlas-sys.com/documentation/illiad/

Now that the archive database is in place and it can be accessed via Microsoft Access, Lars can begin the next project: converting his current Microsoft Access reports for copyright payment and compliance, bibliographic reports and workflow analysis used against the main ILLiad database to run against the transaction archive. Working with the archive in this way will undoubtedly result in some modifications to the database structure and perhaps changes to the way data is selected for import. For example, it may not be necessary to capture all of the records or fields in the Notes, History or Tracking. It would also be possible to retain some calculations created for reports in a field for later use. Although the structure and processes may change or be refined over time, getting the initial archive created and populated is the necessary first step.

Appendices - SQL code

Appendix A - Create the transaction archive

```
CREATE DATABASE [ILLIAD_TRANSACTION_ARCHIVE]
GO

USE ILLIAD_TRANSACTION_ARCHIVE
CREATE TABLE [ReasonsForCancellation] (
	[ReasonNumber] int NOT NULL ,
	[NVTGC] varchar (20)  NOT NULL ,
	[Reason] varchar (150)  NULL ,
	[DefaultNote] varchar (255)  NULL ,
	CONSTRAINT [ReasonNVTGC4_PK] PRIMARY KEY  CLUSTERED
	(
		[ReasonNumber],
		[NVTGC]
	)  ON [PRIMARY]
) ON [PRIMARY]


CREATE TABLE [LendingReasonsForCancellation] (
	[ReasonNumber] int NOT NULL ,
	[NVTGC] varchar (20)  NOT NULL ,
	[Reason] varchar (150)  NULL ,
	[DefaultNote] varchar (255)  NULL ,
	[OCLCCode] varchar (50)  NULL ,
	[DoclineCode] varchar (50)  NULL ,
	[RLINCode] varchar (50)  NULL ,
	CONSTRAINT [LReasonNVTGC_PK] PRIMARY KEY  CLUSTERED
	(
		[ReasonNumber],
		[NVTGC]
	)  ON [PRIMARY]
) ON [PRIMARY]


CREATE TABLE [Tracking] (
	[TransactionNumber] [float] NOT NULL ,
	[DateTime] datetime NOT NULL ,
	[ChangedTo] varchar (40)  NOT NULL ,
	[ChangedBy] varchar (50)  NULL ,
	CONSTRAINT [PK_Tracking_1__14] PRIMARY KEY  CLUSTERED
	(
		[TransactionNumber],
		[DateTime],
		[ChangedTo]
	)  ON [PRIMARY]
) ON [PRIMARY]
```

```
CREATE TABLE [History] (
	[TransactionNumber] int NOT NULL ,
	[DateTime] datetime NOT NULL ,
	[Entry] varchar (250)  NOT NULL ,
	[Username] varchar (50)  NULL ,
	CONSTRAINT [PK_History] PRIMARY KEY  NONCLUSTERED
	(
		[TransactionNumber],
		[DateTime],
		[Entry]
	)  ON [PRIMARY]
) ON [PRIMARY]


CREATE TABLE [Notes] (
	[TransactionNumber] float NOT NULL ,
	[NoteDate] datetime NOT NULL ,
	[Note] varchar (800) NOT NULL ,
	[AddedBy] varchar (20)  NULL ,
	[NoteType] varchar (50) NULL CONSTRAINT [DF_Notes_NoteType]
DEFAULT ('User'),
	CONSTRAINT [PK_Notes_1__12] PRIMARY KEY  CLUSTERED
	(
		[TransactionNumber],
		[NoteDate],
		[Note]
	)  ON [PRIMARY]
) ON [PRIMARY]
GO


CREATE TABLE [Transactions] (
	[TransactionNumber] int NOT NULL ,
	[DateArchived] datetime NULL ,
	[Status] varchar (15)  NULL ,
	[Department] varchar (255)  NULL ,
	[PickupLocation] varchar (20) NULL ,
	[RequestType] varchar (8)  NULL ,
	[LoanAuthor] varchar (100)  NULL ,
	[LoanTitle] varchar (255)  NULL ,
	[LoanPublisher] varchar (50)  NULL ,
	[LoanPlace] varchar (30)  NULL ,
	[LoanDate] varchar (30)  NULL ,
	[LoanEdition] varchar (30)  NULL ,
	[PhotoJournalTitle] varchar (255)  NULL ,
	[PhotoJournalVolume] varchar (30)  NULL ,
	[PhotoJournalIssue] varchar (30)  NULL ,
	[PhotoJournalMonth] varchar (30)  NULL ,
	[PhotoJournalYear] varchar (30)  NULL ,
	[PhotoJournalInclusivePages] varchar (30)  NULL ,
	[PhotoArticleAuthor] varchar (100)  NULL ,
	[PhotoArticleTitle] varchar (250)  NULL ,
	[CitedIn] varchar (40)  NULL ,
	[NotWantedAfter] varchar (40)  NULL ,
```

```
[TransactionStatus] varchar (40)  NULL ,
[TransactionDate] datetime NULL ,
[ISSN] varchar (20)  NULL ,
[ILLNumber] varchar (32)  NULL ,
[ESPNumber] varchar (32)  NULL ,
[LendingString] varchar (150)  NULL ,
[BaseFee] [money] NULL ,
[PerPage] [money] NULL ,
[Pages] int NULL ,
[DueDate] datetime NULL ,
[RenewalsAllowed] varchar (3)  NULL ,
[SpecIns] varchar (40)  NULL ,
[Pieces] int NULL ,
[LibraryUseOnly] varchar (3)  NULL ,
[AllowPhotocopies] varchar (3)  NULL ,
[LendingLibrary] varchar (16)  NULL ,
[ReasonForCancellation] varchar (100)  NULL ,
[CallNumber] varchar (100)  NULL ,
[Location] varchar (255)  NULL ,
[Maxcost] varchar (50)  NULL ,
[ProcessType] varchar (10)  NULL ,
[ItemNumber] varchar (10)  NULL ,
[LenderAddressNumber] [float] NULL ,
[Ariel] varchar (3)  NULL ,
[PhotoItemAuthor] varchar (100)  NULL ,
[PhotoItemPlace] varchar (40)  NULL ,
[PhotoItemPublisher] varchar (40)  NULL ,
[PhotoItemEdition] varchar (40)  NULL ,
[DocumentType] varchar (15)  NULL ,
[InternalAcctNo] [float] NULL ,
[PriorityShipping] varchar (3)  NULL ,
[Rush] varchar (30)  NULL ,
[WantedBy] varchar (25)  NULL ,
[SystemID] varchar (32)  NULL ,
[IFMCost] varchar (30)  NULL ,
[ShippingOptions] varchar (50)  NULL ,
[LendingChecksReceived] varchar (50)  NULL ,
[ReferenceNumber] varchar (50)  NULL ,
[CopyrightComp] varchar (3)  NULL ,
[TAddress] varchar (100)  NULL ,
[ReceivedVia] varchar (20)  NULL ,
[CancellationCode] varchar (50)  NULL ,
[CCSelected] varchar (3)  NULL ,
[OriginalTN] int NULL ,
[OriginalNVTGC] varchar (20)  NULL ,
[InProcessDate] varchar (8)  NULL ,
[InvoiceNumber] int NULL ,
[BorrowerTN] int NULL ,
[WebRequestForm] varchar (100)  NULL ,
[TName] varchar (100)  NULL ,
[IFMPaid] varchar (3)  NULL ,
[BillingAmount] varchar (15)  NULL ,
[ConnectorErrorStatus] varchar (50)  NULL ,
[BorrowerNVTGC] varchar (20)  NULL ,
```

```
	[TISOPaymentMethod] varchar (10)  NULL ,
	[CCCOrder] varchar (3)  NULL ,
	[ISOStatus] varchar (50)  NULL ,
	[ShippingDetail] varchar (50)  NULL ,
	[OdysseyErrorStatus] varchar (50)  NULL ,
	[WorldCatLCNumber] varchar (50)  NULL ,
	[Locations] varchar (255)  NULL ,
	CONSTRAINT [PK_Transactions] PRIMARY KEY  NONCLUSTERED
	(
		[TransactionNumber]
	)  ON [PRIMARY]
) ON [PRIMARY]
GO


CREATE  UNIQUE  INDEX [byTransactionNumber] ON
[Transactions]([TransactionNumber]) ON [PRIMARY]
GO
```

```
SELECT  t.TransactionNumber,
        GETDATE()as DateArchived,
        t.Username,u.Status,
        u.Department,u.NVTGC,
        t.RequestType,
        t.LoanAuthor, t.LoanTitle,
        t.LoanPublisher,t.LoanPlace,
        t.LoanDate,t.LoanEdition,
        t.PhotoJournalTitle,t.PhotoJournalVolume,
        t.PhotoJournalIssue,t.PhotoJournalMonth,
        t.PhotoJournalYear,t.PhotoJournalInclusivePages,
        t.PhotoArticleAuthor,t.PhotoArticleTitle,
        t.CitedIn,t.NotWantedAfter,
        t.TransactionStatus,t.TransactionDate,
        t.ISSN,t.ILLNumber,
        t.ESPNumber,t.LendingString,
        t.BaseFee,t.PerPage,t.Pages,
        t.DueDate,t.RenewalsAllowed,
        t.SpecIns,t.Pieces,
        t.LibraryUseOnly,t.AllowPhotocopies,
        t.LendingLibrary,t.ReasonForCancellation,
        t.CallNumber,t.Location,t.Maxcost,
        t.ProcessType,t.ItemNumber,
        t.LenderAddressNumber,t.Ariel,
        t.PhotoItemAuthor,t.PhotoItemPlace,
        t.PhotoItemPublisher,t.PhotoItemEdition,
        t.DocumentType,t.InternalAcctNo,
        t.PriorityShipping,t.Rush,t.WantedBy,
        t.SystemID, t.IFMCost,t.ShippingOptions,
        t.CCCNumber,t.ReferenceNumber,
        t.CopyrightComp,t.TAddress,
        t.ReceivedVia,t.CancellationCode,
        t.CCSelected,t.OriginalTN,t.OriginalNVTGC,
        t.InProcessDate,t.InvoiceNumber,
        t.BorrowerTN,t.WebRequestForm,
        t.TName,t.IFMPaid,t.BillingAmount,
        t.ConnectorErrorStatus,t.BorrowerNVTGC,
        t.TISOPaymentMethod,t.CCCOrder,
        t.ISOStatus,t.ShippingDetail,
        t.OdysseyErrorStatus,t.WorldCatLCNumber,
        t.Locations
/* FROM Transactions t INNER JOIN [Users-KKU] u */
FROM Transactions t INNER JOIN Users u
ON t.Username = u.Username
WHERE t.TransactionDate < '06/01/2006'
AND (t.TransactionStatus='Request Finished'
OR t.TransactionStatus='Cancelled by ILL Staff')
```

Appendix C - Select records from Notes table

```
/* Code to select records from Notes table
Notes: This select uses the TransactionDate from
the Transactions table because that is the date used to
select records from the Transactions table for the archive.

The first select retrieves records with patron information based on
the field NoteType='User' and replaces the username with the text
"Patron". The second select retrieves everything else as is.

For comparison, this is the code to select all, regardless of NoteType:
SELECT n.* from Notes n  INNER JOIN Transactions t
ON n.TransactionNumber=t.TransactionNumber
INNER JOIN [Users-KKU] u
ON t.Username=u.Username
WHERE EXISTS (SELECT t.Username FROM [Users-KKU])
AND (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'
ORDER BY n.NoteType
*/



SELECT n.TransactionNumber, n.NoteDate, n.Note, 'Patron' as AddedBy, n.NoteType
FROM Notes n INNER JOIN Transactions t
ON n.TransactionNumber=t.TransactionNumber
WHERE n.NoteType ='User'
AND (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'


SELECT n.* from Notes n INNER JOIN Transactions t
ON n.TransactionNumber=t.TransactionNumber
WHERE n.NoteType !='User'
AND (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'
```

Appendix D - Select records from History table

/* Code to select records from History table
Notes: This select uses the TransactionDate from
the Transactions table because that is the date used to
select records from the Transactions table for the archive.

The first select retrieves records with patron information and
replaces Username with the text "Patron". The second select
retrieves all records with staff usernames from the Staff table
or system information.

For comparison, this is the code to select all, regardless of Username:
SELECT h.*
FROM History h INNER JOIN Transactions t
ON h.TransactionNumber=t.TransactionNumber
INNER JOIN [Users-KKU] u
ON t.Username=u.Username
WHERE EXISTS (SELECT t.username FROM [Users-KKU])
AND (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'
ORDER BY h.Username
*/

SELECT h.TransactionNumber, h.DateTime, h.Entry, 'Patron' as Username
FROM History h INNER JOIN Transactions t
ON h.TransactionNumber=t.TransactionNumber
WHERE h.Username !='System'
AND h.Username !='Odyssey'
AND h.Username NOT IN (SELECT username FROM Staff)
AND (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'

SELECT h.* from History h INNER JOIN Transactions t
ON h.TransactionNumber=t.TransactionNumber
WHERE (h.username='System'
OR h.Username='Odyssey'
OR h.Username in (SELECT username FROM Staff))
AND (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'

/* Code to select records from Tracking table
Notes: This select uses the TransactionDate from
the Transactions table because that is the date used to
select records from the Transactions table for the archive.

The first select retrieves records with patron information and
replaces ChangedBy with the text "Patron". The second select
retrieves all records with staff usernames from the Staff table
or system information.

For comparison, this is the code to select all, regardless of ChangedBy:

```
SELECT k.*
FROM Tracking k INNER JOIN Transactions t
ON k.TransactionNumber=t.TransactionNumber
INNER JOIN [Users-KKU] u on
t.Username=u.Username
WHERE EXISTS (SELECT t.Username FROM [Users-KKU])
AND t.TransactionDate < '06/01/2006'
AND (t.TransactionStatus='Request Finished'
OR t.TransactionStatus='Cancelled by ILL Staff')
ORDER BY k.ChangedBy
*/


SELECT k.TransactionNumber, k.DateTime, k.ChangedTo, 'Patron' as Username
FROM Tracking k INNER JOIN Transactions t
ON k.TransactionNumber=t.TransactionNumber
WHERE t.TransactionDate < '06/01/2006'
AND (t.TransactionStatus='Request Finished'
OR t.TransactionStatus='Cancelled by ILL Staff')
AND k.ChangedBy !='System'
AND k.ChangedBy !='Odyssey'
AND k.ChangedBy NOT IN (SELECT Username FROM Staff)


SELECT k.*
FROM Tracking k INNER JOIN Transactions t
ON k.TransactionNumber=t.TransactionNumber
WHERE t.TransactionDate < '06/01/2006'
AND (t.TransactionStatus='Request Finished'
OR t.TransactionStatus='Cancelled by ILL Staff')
AND (k.ChangedBy ='System'
OR k.ChangedBy ='Odyssey'
OR k.ChangedBy IN (SELECT Username FROM Staff))
```

```
/* Notes: Records from the Notes, History and Tracking table must be deleted
before records from the Transactions table are deleted.

The inner join syntax used is not ANSI SQL compliant but is correct for
Transact SQL.

Use the same date criteria from the select statements used
to populate the archive.

Additional criteria for KU has been removed after the first join:
INNER JOIN [Users-KKU] u on
t.Username=u.Username
WHERE EXISTS (SELECT t.Username FROM [Users-KKU])
AND
remaining criteria here
*/


DELETE FROM Notes
FROM Notes n INNER JOIN Transactions t
ON n.TransactionNumber=t.TransactionNumber
WHERE (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'


DELETE FROM History
FROM History h INNER JOIN Transactions t
ON h.TransactionNumber=t.TransactionNumber
WHERE (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
AND t.TransactionDate < '07/01/2006'


DELETE FROM Tracking
FROM Tracking k INNER JOIN Transactions t
ON k.TransactionNumber=t.TransactionNumber
WHERE t.TransactionDate < '06/01/2006'
AND (t.TransactionStatus='Request Finished'
OR t.TransactionStatus='Cancelled by ILL Staff')


DELETE FROM Transactions
WHERE TransactionDate < '06/01/2006'
AND (TransactionStatus='Request Finished'
OR TransactionStatus='Cancelled by ILL Staff')
```