Spring 4-1-2012

Major Revision to Document Imaging System to Improve Efficiency and Reporting Capabilities

Marek Juracek
Dakota State University

MAJOR REVISION TO DOCUMENT IMAGING SYSTEM TO IMPROVE EFFICIENCY AND REPORTING CAPABILITIES

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Systems

April, 2012

By
Marek Juracek

Project Committee:

Dr. Ronghua Shan
Dr. Stephen Krebsbach
Professor Chris Olson

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Marek Juracek

Master's Project Title: Major Revision to Document Imaging System to Improve Efficiency and Reporting Capabilities

Faculty supervisor: Ronghua Shan Date: 5/1/12

Committee member: [signature] Date: 5/1/12

Committee member: Chris Olson Date: 5/1/12

ACKNOWLEDGMENT

I would like to thank my friends and co-workers at Daytona State College for the technical support they have provided to me throughout the development of this project.

Most of all, I would like to thank my amazing family, my daughter Tatiana, my son Sebastian, and my loving wife Carolina for their incredible support which has helped me to overcome all the obstacles I have encountered during my studies. My achievements would not be possible without their love.

ABSTRACT

Daytona State College's Document Imaging system has served well for nearly 10 years. Since 2002 the college has processed more than 2.4 million scans, handling about 50 million pages, storing about 1TB of data. However, recently the users of the system started experiencing significant slowdowns during the operation of the system due to its growth. Another design drawback to the current Document Imaging system is that it only operates in the append mode, meaning document entities that already exist in the imaging system will get appended to the end of the existing entity. This makes the system to grow much more quickly than necessary, making our backup, especially our full-backup operations, very resource and time consuming. Additionally, the current append-only design does not and cannot provide functionality for maintaining different retention policies.

This project is about redesigning the Document Imaging database structure and system functionality to speed up its operations while also taking into account the future growth of the system, as well as modifying the original design from append-only to no-append approach.

Since the beginning of the project managers of this system had limited reporting capabilities. All the reporting needs have been performed against the live transactional database that is also serving for all the Document Imaging operations. To speed up some of the operations, the log table has been cleaned up periodically to remove non-critical records, which resulted in losing some of the reporting capabilities.

This project will also handle the reporting needs by creating a simple data warehouse that will permanently store all the needed log entries as well as additional supporting data to give the managers the needed reporting flexibility.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Marek Juracek

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Background of the Problem

Daytona State College has been using its' in-house developed Document Imaging system to meet its imaging needs since 2002. The system was initially developed to be used exclusively by the newly formed Document Imaging department and its 15 full time employees. These employees received in-depth training which allowed them to utilize the Document Imaging system to its full potential, while avoiding the system's pitfalls. This department used to be solely responsible for providing imaging services to the rest of the departments within the college. In 2009 Daytona State College changed its document imaging strategy and decided to de-centralize the imaging tasks and responsibilities. Today, each department has the opportunity to operate the Document Imaging system and therefore process their own documents at their own pace. There are almost 400 users registered and authorized to use the Document Imaging system today. This decentralization provides many advantages, but also accounts for many issues.

The Document Imaging system consists of several applications. A scanning part, written in C++ as a client/server application, a document retrieval part written in C# as a web application hosted in the college's web portal, and several other maintenance applications used for managing users, maintaining the integrity of the data, and other essential tasks. The back end of the applications is Informix database. This transactional database is also a core data source for the college's ERP system, as well as over 300 other custom-developed applications. The database contains over 800 tables with over 60 million records of data. The

actual images are stored on a file server as TIFF files. The Document Imaging system uses 8 dedicated DB tables and 5 main support tables.

As mentioned previously, in the beginning, the Document Imaging system was designed to operate in the append-only mode, meaning document entities that already exist in the imaging system get appended to the end of the existing file. This makes the system to grow much more quickly than necessary, making our daily backups, and especially our full-backup operations, very time consuming, and requiring a very large storage area. Additionally, the current append-only design does not and cannot provide functionality for maintaining different retention policies.

Statement of the problem

The Document Imaging system has served the college efficiently. However, with the large amount of data have been processed since 2002, the system's operations have started to slow down. These slow-downs are quite significant as they affect the operator's performance and cause higher database server resource usage and higher network usage. Since 2002 the college has processed more than 2.4 million scans, handling about 50 million pages, storing about 1TB of data.

Backup and storage operations can be very costly in terms of hardware and personnel resources. Efficient management and usage of various retention policies, as opposed to permanent-only retention, can greatly reduce the storage and backup requirements, ultimately leading to reduction of cost of the imaging operations.

Another problem with the current Document Imaging system is that all the reporting needs are currently executed against the transactional database. This adds an unnecessary

overhead to the main database. In addition, to speed up the Document Imaging system older database records have been periodically deleted in the past. This means that the college has lost the ability to accurately report past operator's performances and provide trend analysis.

Objectives of the project

This project will resolve both issues. Restructuring of the Document Imaging related tables (indexing, keys, stored procedures, and several key SQL cost analysis) and the file operation, and the storage strategy will increase the efficiency of the system. A new data warehouse will provide an accurate data repository that will address the current reporting needs as well as give the necessary foundation for any future reporting requests. In addition, this project will also handle a few Cognos reports that will allow the users to report on the performance and system utilization. These Cognos reports will be based on the new data warehouse.

Deliverables of the project

1. New/Revised database tables (including keys, indexes, explanation of changes)
2. New design strategy on handling file storage (creating new files every time instead of appending to existing files)
3. New/Revised SQL statements (including SQL cost analysis on critical SQLs)
4. Suggestions to the existing C++ and/or C# systems to enhance efficiency
5. Efficiency comparison data
6. A new data warehouse to allow for accurate reporting
7. Cognos reports to show various aspects of operators' performances and trend analysis

CHAPTER 2

LITERATURE REVIEW

History of the document imaging database design

Daytona State College has been using Jenzabar CX Enterprise Resource Planning (ERP) system since 1998. This ERP system stores majority of personal information for both, students and employees, in id_rec and profile_rec tables as shown in figure 1. The college's Document Imaging system has originally been intended mainly for scanning and storing student records, and since all the student information is already being stored in the id_rec and profile_rec tables, there has not been any need to store the same information in the related document imaging tables. Instead, the Document Imaging system only stores the student ID within its' database structure. Soon after student records, human resources records have also been requested to get scanned and permanently stored in the document imaging system. Since employees' information is also stored in the ERP's id_rec and profile_rec, the document imaging system can utilize the same database structure as for students without any modifications.

A few years into this project, the document imaging team was challenged with scanning non-personal documents, such as tracking of scholarships, accounting's checks and vendor information, and other report type documents. In order to accommodate these requests, additional database tables were needed to handle the various indexing needs. It was desirable for the document imaging to easily adopt and store other non-personal records. With these

requirements in mind, additional tables, dbimg_id_rec and dbimg_profile_rec, were added to the database structure. Also, the dbimgdoctype_table was expanded with the var1 through var5 columns to allow for indexing flexibility and to easily accommodate any future non-personal imaging requests.

Each department could utilize one or more document imaging types (document type). The specifications of all document types are stored in the dbimgdoctype_table as shown in table 9.

Each department can also select one or more staff members for scanning documents as well as retrieval of the documents. Initially, these 2 groups of users were implemented and managed separately in the system. All the users that have retrieval-only permissions are stored in the dbimg_usrsec_rec (table 8). The database table dbimg_security_rec (table 6) contains information for all users that have the ability to utilize the imaging application to scan, approve, deny, and other operations. The dbimg_security_rec has also been used for storing user credentials as the scanning application requires access validation. However, since 2009, the scanning application has become a part of the college's application portal which utilizes different set of credentials, and thus the dbimg_security_rec has become unnecessary at this time. The only reason to keep this table as part of the document imaging system is in case the scanning application needs to become standalone again if, for example, the college's ERP changes.

The dbimg_filelock_rec (table 2) has been added to the imaging system after the initial release date to address one of the data integrity issues. This table serves as a locking mechanism that prevents multiple imaging operators to act on the same document and document type to prevent file corruption and loss of data.

The dbimg_stat_rec (table 7) stores information of operators' document preparation activities to provide the managers with the ability to track and report on various aspects of imaging operations.

One of the most important tables that create the core of document imaging is the dbimg_mstr_rec (table 4). This table links the document descriptor, document type, document physical storage, and other operator's information together and allows the end users to retrieve the documents.

Document imaging operations

In order to understand and analyze how to improve the efficiency of the imaging system it is necessary to be familiar with the operations of the system.

Once the desired document type and users are setup in the respective database tables, operators can begin preparing and scanning documents. Currently the imaging system works in two modes, a quick mode and a batch mode. In the quick mode, every scanned document must be either accepted or rejected immediately after the scan. In the batch mode, preparation (barcoding), scanning, and acceptance of documents is performed in a batch mode usually in a team of 3 users. All the operations are currently being recorded for retrieval and statistical purposes. The scanning and accepting operations are being recorded in the dbimg_mstr_rec database table. The preparation operation is being recorded in the dbimg_stat_rec table. While retrieval information is critical in order to get to the physical file, the rejected records are non-critical. When an additional document is scanned under the same document descriptor (i.e. student), and document type, then the document is appended to the original document. This functionality simplifies the storage and handling of the records, but significantly adds to the

network traffic during both, scanning and retrieval operations. This means, that a student after 2 years of studies at the college can end up with a 100 page document for each document type, such as student record or financial aid. This is currently a huge issue because it significantly slows down the scanning operations. For example, in order to accept and scan an application for scholarship that has 2 pages, a file of 100 pages, roughly 3MB of storage, needs to be first transferred to the local PC, the new file gets appended, and then the new file is transferred back to the centralized storage area.

Transactional database structure of the current system

The following section shows the details about the current Document Imaging database structure.

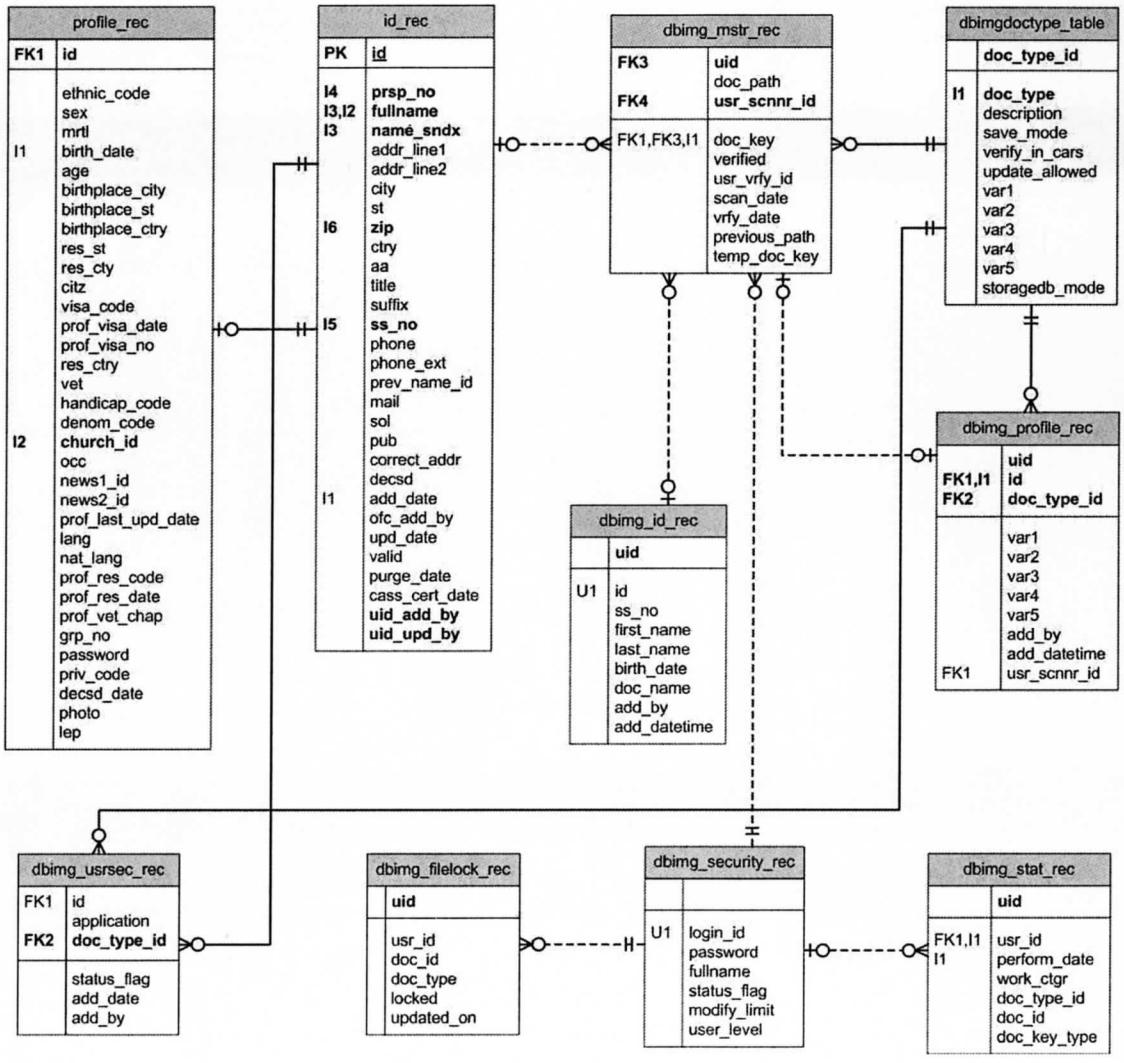


Figure 1. Current Document Imaging table structure

Table 1. Current Document Imaging core tables

Table Name	Description	Number of Rows
dbimg_filelock_rec	Contains information about locked files	0
dbimg_id_rec	Contains personal/document information	80,318
dbimg_mstr_rec	Contains information about scanned documents	2,506,540
dbimg_profile_rec	Contains additional personal/document information	48,316
dbimg_security_rec	Contains list of all users for scanning	125
dbimg_stat_rec	Contains prep/scan/verify statistics	520,265
dbimg_usrsec_rec	Contains list of all users for retrieving only	1,108
dbimgdoctype_table	Contains list of all document types	63

Table 2. The dbimg_filelock_rec table attributes

Attribute Name	Data Type	Attribute Length	Comments
doc_id	char	25	Unique document key
doc_type	char	20	Group associated to the document
Uid	serial	4	Unique serial id
Locked	char	1	Correct values: y-yes, n-no
station_id	char	25	PC Netbios name
updated_on	char	15	format: yyyymmddhhmmss
usr_id	char	20	File locked by this user

Table 3. The dbimg_id_rec table attributes

Attribute Name	Data Type	Attribute Length	Comments
Id	char	10	Unique key - 'D'+uid
ss_no	char	11	Social Security Number
first_name	char	15	First Name
last_name	char	20	Last Name
birth_date	date	4	Birth date
Uid	serial	4	Unique serial id
doc_name	char	100	Document Name
add_datetime	char	14	Date time stamp:YYYYMMDDHHMMSS
add_by	char	20	Record added by user\Login ID

Table 4. The dbimg_mstr_rec table attributes

Attribute Name	Data Type	Attribute Length	Comments
doc_key	char	25	Unique key for finding document
doc_type	char	20	Group associated to the document
doc_path	char	100	Path of the document
Verified	char	1	Signifies verification of scan
Uid	serial	4	Unique serial id
usr_scnnr_id	char	20	Scanner's unique network id
previous_path	char	100	Previous path before appending
usr_vrfy_id	char	20	Verifier's unique network id
scan_date	date	4	Date stamp of document scan
vrfy_date	date	4	Date stamp of document verification
temp_doc_key	char	25	Copy of the Unique key

Table 5. The dbimg_profile_rec table attributes

Attribute Name	Data Type	Attribute Length	Comments
Id	char	10	Unique key - 'D'+uid
doc_type_id	char	3	Document type id from dbimgdoctype_table
var1	char	50	Represents field as defined in dbimgdoctype_table for an individual doc type
var2	char	50	see desc of var1
var3	char	50	see desc of var1
var4	char	50	see desc of var1
Uid	serial	4	Unique serial id
var5	char	50	see desc of var1
add_datetime	char	14	Date time stamp:YYYYMMDDHHMMSS
add_by	char	20	Record added by user\Login ID

Table 6. The dbimg_security_rec table attributes

Attribute Name	Data Type	Attribute Length	Comments
password	char	20	password
login_id	char	20	Unique id
fullname	char	32	Users full name
user_level	integer	4	User's permissions level
status_flag	char	1	status of the login id
modify_limit	integer	4	days for modification

Table 7. The dbimg_stat_rec table attributes

Attribute Name	Data Type	Attribute Length	Comments
Uid	serial	4	Unique identification number
doc_id	char	10	ID of the document/student/employee/etc...
usr_id	char	20	User id as defined in dbimg_security_rec.login_id table
perform_date	date	4	Date of performance
work_ctgr	char	1	Work category: p - prepped s - scanned v - verified q - quality assurance
doc_key_type	char	15	Document key + Document type id
doc_type_id	integer	4	Id of document type

Table 8. The dbimg_usrsec_rec table attributes

Attribute Name	Data Type	Attribute Length	Comments
application	char	20	Application name
Id	char	10	ERP's Cars id
doc_type	char	20	doc_type
can_scan	char	1	Access to DocuImg scanning application
status_flag	char	1	status of the login id
add_date	date	4	Record added on date
add_by	char	14	Record added by user add_by

Table 9. The dbimgdoctype_table table attributes

Attribute Name	Data Type	Attribute Length	Comments
doc_type	char	20	Unique group id
description	char	100	Description of the document type
save_mode	char	1	A-append, N-create new file
verify_in_cars	char	1	Enforce validation of the doc type against cars id_rec, y-yes must be
doc_type_id	serial	4	Unique doc type id
var1	char	100	Field num.1 used for search
storagedb_mode	integer	4	Storage+Database Mode: 1-SAN+CARS DB, 2-local+no DB
var2	char	16	Field num.2 used for search
var3	char	16	Field num.3 used for search
var4	char	16	Field num.4 used for search
var5	char	16	Field num.5 used for search
update_allowed	char	1	If set to y(yes), docuImg personnel can update personal info in dbimg_id_rec

CHAPTER 3

SYSTEM DESIGN

Planning phase

Due to the nature of the problem, a waterfall development model will be used. This approach makes a perfect fit for this project since the project requirements have already been well defined and are completely static. The waterfall approach is based on the idea of executing well defined phases in only one sequence. The output of each phase serves as the input for the next phase. The following figure depicts the model adapted to our needs.

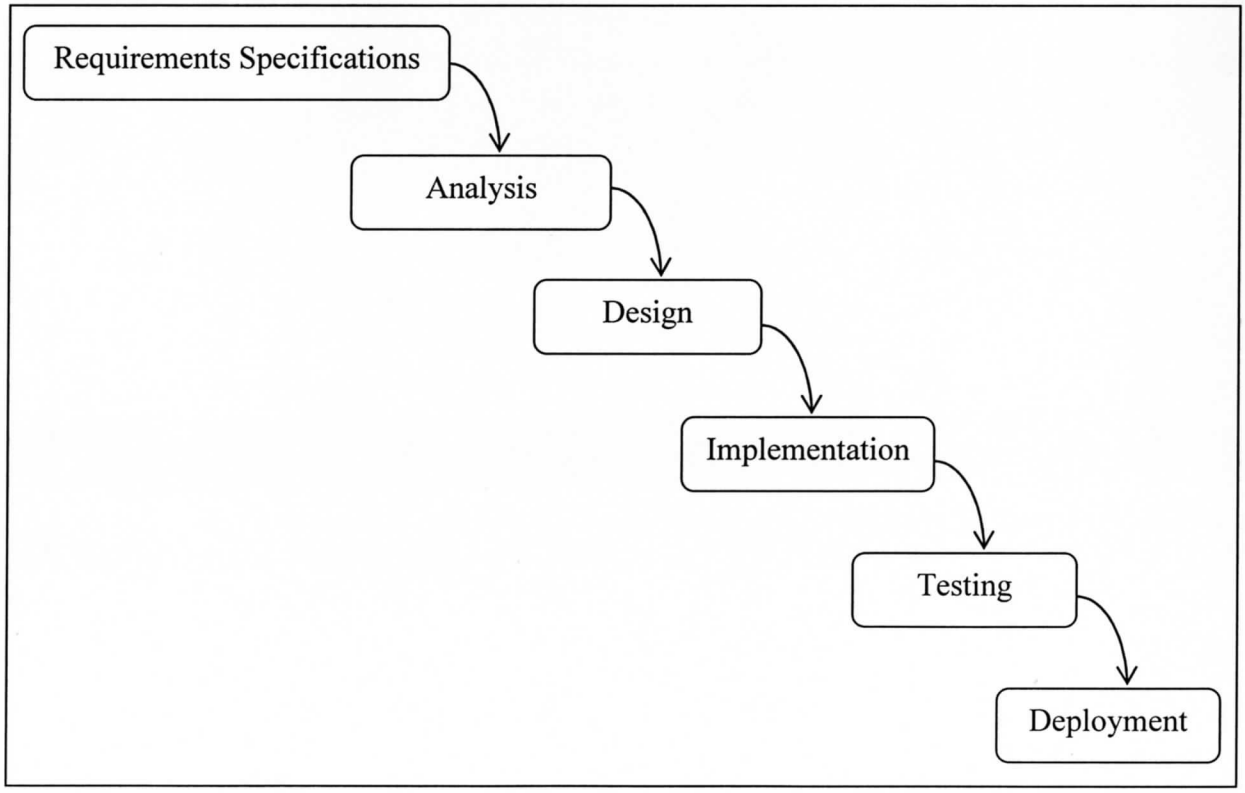


Figure 2. Waterfall development model

As previously stated, the back-end of the Document Imaging system is an Informix 11 database. Since the Document Imaging system is dependent on other database tables from the college's ERP system (id_rec, profile_rec, etc.) that reside on the same database, all the table-related revisions will be implemented in the same Informix database as well. After the creation of the new tables, or re-design of the current tables, those SQLs that are affected by the changes will be identified and modified, or re-created as needed. Furthermore, the SQLs that are affected the most will be identified and evaluated on the estimated cost for both, before and after the modifications, versions. All the results will be thoroughly documented. Since these SQLs are one of the main components of the Document Imaging system, the goal is to clearly demonstrate the gains in performance after the system redesign.

Since Daytona State College's current data warehouse is housed on MySQL database engine, the design of the new data warehouse database will also be based on MySQL database engine. One of the scripting languages that are currently available under our HP UNIX environment will be utilized during the implementation of this data warehouse. Also, the goal is to utilize the same tools that are currently being utilized already simply because of the current common knowledge base; anybody else on the IT team will later on be readily available to expand the warehouse as needed without any steep learning curve.

The college is currently migrating all of its reporting needs to Cognos 10 reporting tool. Cognos 10 offers many benefits to our IT department as well as to our end users. Some of the main advantages that this tool offers are centralized permission management and report management. Cognos will be used to create the reports and take advantage of our existing permission and report structures. Additionally, the Cognos' framework manager will be used

for adding the additional warehouse tables to the existing package. Since Cognos 10 is a web-based product, it will also make the testing and deployment of the reports very simple.

The Appendix A contains the work breakdown structure of the project. The Appendix B shows the actual project plan with timelines, durations, and dependencies. Several milestones of the process have been identified throughout the project to easier track progress and make sure the project will be completed on time. The milestones include Initiating Completed, Planning Completed, Redesign Completed, Database Warehouse Completed, Executing Completed, and of course, Project Completed.

Design Phase

Moving to no-append scanning mode

In the first step of this phase the imaging system will be redesigned to handle all subsequent scans as separate document entities as opposed to the constant appending of any subsequent scans.

Currently, after a scanned document has been accepted, the imaging system searches for an existing document with the same document entity and document type in the database, and if found, the system will download the existing TIFF file, append the newly scanned and accepted document, and then upload the final file back to the file server. Table 10 demonstrates the outcomes of the current process and the related database entries. For the document key 272635 and document type SR_OldRecords, the imaging system appended 27 times to the original physical file.

Table 10. Examples records for document entity in the append-only mode

uid	doc_key	doc_type	verified	scan_date	vrfy_date
2471755	272635	SR_OldRecords	Y	6/13/2011	6/13/2011
2258113	272635	SR_OldRecords	D	9/15/2010	9/15/2010
2221443	272635	SR_OldRecords	D	8/3/2010	8/3/2010
2012733	272635	SR_OldRecords	D	10/5/2009	10/5/2009
2005773	272635	SR_OldRecords	D	9/23/2009	9/23/2009
1989528	272635	SR_OldRecords	D	9/1/2009	9/1/2009
1989520	272635	SR_OldRecords	D	9/1/2009	9/1/2009
1960445	272635	SR_OldRecords	D	7/31/2009	7/31/2009
1960439	272635	SR_OldRecords	D	7/31/2009	7/31/2009
1958430	272635	SR_OldRecords	D	7/29/2009	7/29/2009
1958426	272635	SR_OldRecords	D	7/29/2009	7/29/2009
1958423	272635	SR_OldRecords	D	7/29/2009	7/29/2009
1920034	272635	SR_OldRecords	D	6/4/2009	6/5/2009
1920020	272635	SR_OldRecords	D	6/4/2009	6/5/2009
1734133	272635	SR_OldRecords	D	6/27/2008	8/20/2008
1735730	272635	SR_OldRecords	D	6/30/2008	8/20/2008
1161267	272635	SR_OldRecords	D	10/10/2006	10/10/2006
968708	272635	SR_OldRecords	D	1/23/2006	1/24/2006
420181	272635	SR_OldRecords	D	4/12/2004	4/12/2004
420182	272635	SR_OldRecords	D	4/12/2004	4/12/2004
408488	272635	SR_OldRecords	D	3/29/2004	4/1/2004
392041	272635	SR_OldRecords	D	3/12/2004	3/16/2004
387747	272635	SR_OldRecords	D	3/9/2004	3/9/2004
323174	272635	SR_OldRecords	D	11/6/2003	11/6/2003
322929	272635	SR_OldRecords	D	11/6/2003	11/6/2003
198324	272635	SR_OldRecords	D	6/25/2003	6/25/2003
1808	272635	SR_OldRecords	D	8/21/2002	8/21/2002
1223	272635	SR_OldRecords	D	8/20/2002	8/21/2002

In order to make the transition from the append-only mode to the no-append mode, an adjustment to the options for the dbimg_mstr_rec.verified field will be needed. Table 11 shows the current valid values for the append-only mode. Table 12 shows the adjusted values that are needed for the no-append mode.

Table 11. dbimg_mstr_rec.veified allowed values for append-only mode

verified	description
D	Duplicate
N	Not verified
R	Rejected
Y	Verified

Table 12. dbimg_mstr_rec.veified allowed values for no-append mode

verified	description
N	Not verified
R	Rejected
Y	Verified

Moving from the append-only to no-append mode presents a new challenge. Currently, when users try to retrieve a document, for example, documents for the document key 272635 and document type SR_OldRecords, they are presented with one entry in the search result, which means only one file to download and browse. However, in the no-append mode, the users would be presented with 28 entries. The solution to this problem will actually make the retrieving process more efficient. A new field doc_description will be added to the dbimg_mstr_rec (table 4) which will allow the operators of the imaging system to further specify what document they are scanning, such as Application for Admissions, Application for Financial Aid 1213, etc. What this solution means to the end users and to the imaging system is that the users will be able to choose the specific content they need, thus eliminating the need for searching in one large file for the specific content, and eliminating the need for downloading all the extra files, saving time and network bandwidth. On the other hand, if the operators deem that the extra document description is not needed, then the retrieval system could concatenate all the files without the detail document description (or those with the identical document description) together on the server side, still presenting the end users with

one single file per document description. In the future, this functionality could be further improved with the implementation of the OCR (Optical Character Recognition) capabilities that the current Document Imaging system lacks of.

New functionality to manage document retention policies

As discussed during the previous project phases, moving from the append-only mode to the no-append mode will allow the college to implement the much desired functionality for managing document retention policies.

The necessary database modifications will be very simple. A new field retention_prd will be added to the dbimgdoctype_table that will record the number of days each specific document type should retain its documents. In order to enforce the various retention policies, a new application will have to be written which will compare the age of each document to the related document type's retention period. If the age of the document is greater than the retention period than the specific document will be physically deleted from the file storage and the document's database information updated accordingly. A new status will be needed for the verified field in the dbimg_mstr_rec.

Redesign of the database structure

In the next step of the project the SQL statement (SQL A) shown in figure 3 will be analyzed. This is one of the queries that significantly contribute to the delays in processing documents and operating Document Imaging system. This SQL has been chosen because it is

used, in slight variations, in many areas of Document Imaging system, such as in scanning, quality control, and retrieval application.

```
SELECT
   dm.uid AS uid,
   i.id AS id,
   i.ss_no AS ss_no,
   i.fullname AS fullname,
   '' AS did,
   '' AS last_name,
   '' AS first_name,
   dm.doc_type AS doc_type,
   dm.verified AS verified,
   dm.usr_scnnr_id AS usr_scnnr_id,
   dm.scan_date AS scan_date,
   dm.usr_vrfy_id AS usr_vrfy_id,
   dm.vrfy_date AS vrfy_date,
   '' AS doc_name,
   dm.doc_path
FROM
   id_rec i,
   dbimg_mstr_rec dm,
   dbimgdoctype_table dd
WHERE
   dm.doc_type = dd.doc_type AND
   dm.doc_type IN (
      SELECT
         a.doc_type
      FROM
         dbimgdoctype_table a, dbimg_usrsec_rec b
      WHERE
         a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
         b.id = '470822' AND
         b.can_scan = 'Y') AND
   dm.scan_date >='01-01-2012' AND
   dm.scan_date <=today AND
   dm.verified ='N' AND
   i.id = dm.doc_key
UNION
SELECT
   dm.uid AS uid,
   0 AS id,
   di.ss_no AS ss_no,
```

```
   " AS fullname,
   di.id AS did,
   di.last_name AS last_name,
   di.first_name AS first_name,
   dm.doc_type AS doc_type,
   dm.verified AS verified,
   dm.usr_scnnr_id AS usr_scnnr_id,
   dm.scan_date AS scan_date,
   dm.usr_vrfy_id AS usr_vrfy_id,
   dm.vrfy_date AS vrfy_date,
   di.doc_name AS doc_name,
   dm.doc_path
FROM
   dbimg_id_rec di,
   dbimg_mstr_rec dm,
   dbimgdoctype_table dd
WHERE
   dm.doc_type = dd.doc_type AND
   dm.doc_type IN (
      SELECT
         a.doc_type
      FROM
         dbimgdoctype_table a, dbimg_usrsec_rec b
      WHERE
         a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
         b.id = '470822' AND
         b.can_scan = 'Y') AND
   dm.scan_date >='01-01-2012' AND
   dm.scan_date <=today AND
   dm.verified ='N' AND
   di.id = dm.doc_key
ORDER BY id DESC;
```

Figure 3. The initial SQL (SQL A) used in the database redesign

The SQL A is taken from the Search Screen (figure 4) of the imaging application. These parameters are chosen in such way so that they mimic what the everyday users do most frequently. Specifically, this SQL is used by the verification process, where the operators search for scanned documents that need to be approved or rejected. In this SQL example the

operator's id 470822 was used, and the search was conducted for all documents that have not been verified since the beginning of 2012.

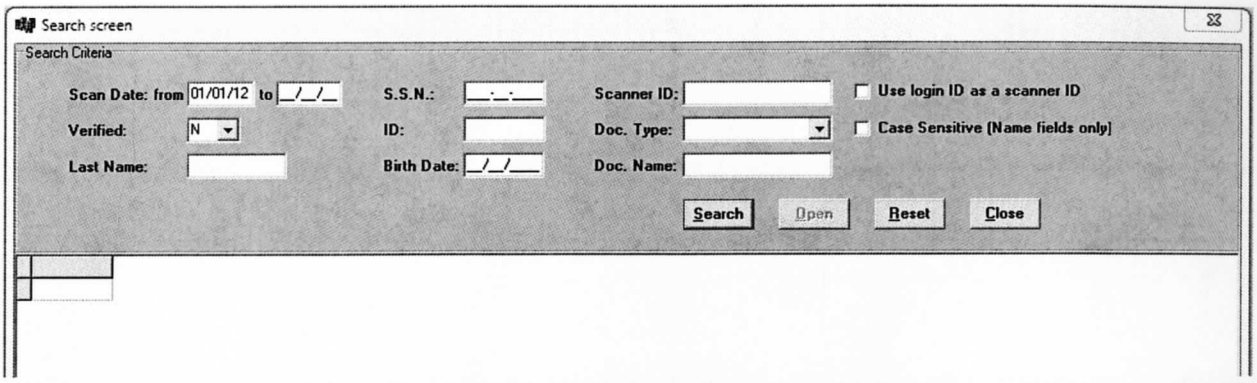


Figure 4. Document Imaging application – Search Screen

Analysis of the major SQL

In order to receive accurate analysis results, it is needed to update the query-related tables statistics first. Besides updating global table statistics (LOW mode), it is also needed to update statistics (HIGH mode) for the indexed columns used in the join or where clause of the query. Figure 5 shows the indexes for all the related tables. Figure 6 shows the specific statistics updates performed in preparation for the analysis.

```
table       dbimg_mstr_rec
index
   unique imgmstr_prim on (uid,doc_key,doc_type,doc_path)
   imgmstr_key2 on (doc_path)
   imgmstr_key on (doc_key)
   imgmstr_type on (doc_type)
   imgmstr_vd_uid on (vrfy_date, uid)


table       dbimgdoctype_table
```

```
index
   unique imgdoctype_prim on (doc_type_id, doc_type)
   imgdoctype_dt on (doc_type)


table       dbimg_id_rec
index
   unique imgid_prim on (id)
   unique imgidupers on (ss_no, first_name, last_name, birth_date, doc_name)


table       id_rec
index
   id_key1 on (name_sndx, fullname)
   id_ss_no on (ss_no)
   id_zip on (zip)
   id_fullname on (fullname)
   id_lastname on (lastname)
   id_firstname on (firstname)
   id_middlename on (middlename)
   id_prsp_no on (prsp_no)
   id_add_date on (add_date)
   id_fullname_search for id_rec on (namesearch(fullname))
   id_lastname_search for id_rec on (namesearch(lastname))
   id_firstname_search for id_rec on (namesearch(firstname))
   id_middlename_search for id_rec on (namesearch(middlename))


table       dbimg_usrsec_rec
index
   unique imgusrsec_prim on (id, application, doc_type)
```

Figure 5. Indexes for tables used in SQL A

```
UPDATE STATISTICS HIGH FOR TABLE dbimg_mstr_rec (doc_type, doc_key, verified,
scan_date);
UPDATE STATISTICS HIGH FOR TABLE dbimgdoctype_table (doc_type);
UPDATE STATISTICS HIGH FOR TABLE dbimg_id_rec (id);
UPDATE STATISTICS HIGH FOR TABLE dbimg_usrsec_rec (id, doc_type, can_scan);
UPDATE STATISTICS HIGH FOR TABLE id_rec (id);
```

Figure 6. The performed statistics updates

For the following query analysis DbVisualizer 7.0.8 application was used for SQL executions and timing, and Informix's SET EXPLAIN utility tool for collecting query plans used for tuning and optimizing of SQLs and database structures. The version of Informix Database Server at the time of the analysis was 11.50.FC6, running on HP-UX B.11.31 (HP-UX 11i v3).

All the test executions below were performed under the same conditions in order to provide as accurate results as possible.

As figure 7 shows, execution of SQL A took approximatelly 4 minutes, and returned 10 records. Figure 8 shows the results of the SET EXPLAIN utility.

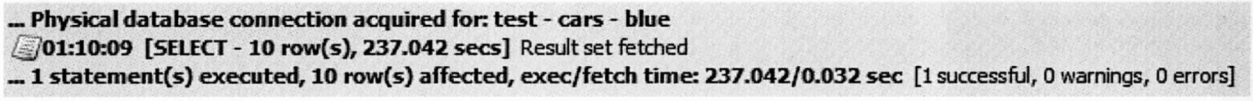


Figure 7. Statistics for SQL A

```
QUERY:
------
{SQL A}
SELECT
   dm.uid AS uid,
   i.id AS id,
   i.ss_no AS ss_no,
   i.fullname AS fullname,
   '' AS did,
   '' AS last_name,
   '' AS first_name,
   dm.doc_type AS doc_type,
   dm.verified AS verified,
```

```
    dm.usr_scnnr_id AS usr_scnnr_id,
    dm.scan_date AS scan_date,
    dm.usr_vrfy_id AS usr_vrfy_id,
    dm.vrfy_date AS vrfy_date,
    '' AS doc_name,
    dm.doc_path
FROM
    id_rec i,
    dbimg_mstr_rec dm,
    dbimgdoctype_table dd
WHERE
    dm.doc_type = dd.doc_type AND
    dm.doc_type IN (
        SELECT
            a.doc_type
        FROM
            dbimgdoctype_table a, dbimg_usrsec_rec b
        WHERE
            a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
            b.id = '470822' AND
            b.can_scan = 'Y') AND
    dm.scan_date >='01-01-2012' AND
    dm.scan_date <=today AND
    dm.verified ='N' AND
    i.id = dm.doc_key
UNION
SELECT
    dm.uid AS uid,
    0 AS id,
    di.ss_no AS ss_no,
    '' AS fullname,
    di.id AS did,
    di.last_name AS last_name,
    di.first_name AS first_name,
    dm.doc_type AS doc_type,
    dm.verified AS verified,
    dm.usr_scnnr_id AS usr_scnnr_id,
    dm.scan_date AS scan_date,
    dm.usr_vrfy_id AS usr_vrfy_id,
    dm.vrfy_date AS vrfy_date,
    di.doc_name AS doc_name,
    dm.doc_path
FROM
    dbimg_id_rec di,
    dbimg_mstr_rec dm,
    dbimgdoctype_table dd
```

```
WHERE
   dm.doc_type = dd.doc_type AND
   dm.doc_type IN (
      SELECT
         a.doc_type
      FROM
         dbimgdoctype_table a, dbimg_usrsec_rec b
      WHERE
         a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
         b.id = '470822' AND
         b.can_scan = 'Y') AND
   dm.scan_date >='01-01-2012' AND
   dm.scan_date <=today AND
   dm.verified ='N' AND
   di.id = dm.doc_key
ORDER BY id DESC

Estimated Cost: 178387
Estimated # of Rows Returned: 2
Temporary Files Required For: Order By

  1) juracem.dm: INDEX PATH

      Filters: ((juracem.dm.verified = 'N' AND juracem.dm.scan_date >= 01/01/2012 ) AND
juracem.dm.scan_date <= TODAY )

   (1) Index Name: informix.imgmstr_type
      Index Keys: doc_type   (Serial, fragments: ALL)
      Lower Index Filter: juracem.dm.doc_type = ANY <subquery>

  2) juracem.dd: INDEX PATH

   (1) Index Name: informix.imgdoctype_dt
      Index Keys: doc_type   (Key-Only)  (Serial, fragments: ALL)
      Lower Index Filter: juracem.dm.doc_type = juracem.dd.doc_type
NESTED LOOP JOIN

  3) juracem.i: INDEX PATH

   (1) Index Name: informix. 1956_6496
      Index Keys: id   (Serial, fragments: ALL)
      Lower Index Filter: juracem.i.id = juracem.dm.doc_key
NESTED LOOP JOIN

   Subquery:
   ---------
```

```
  Estimated Cost: 11
  Estimated # of Rows Returned: 5

    1) juracem.b: INDEX PATH

        Filters: juracem.b.can_scan = 'Y'

      (1) Index Name: informix.imgusrsec_prim
        Index Keys: id application doc_type   (Serial, fragments: ALL)
        Lower Index Filter: juracem.b.id = '470822'

    2) juracem.a: INDEX PATH

        Filters: juracem.a.doc_type LIKE REPLACE ((juracem.b.doc_type || '%' ), ' ' , '' )

      (1) Index Name: informix.imgdoctype_dt
        Index Keys: doc_type   (Key-Only)  (Serial, fragments: ALL)
NESTED LOOP JOIN


Union Query:
------------

  1) juracem.dm: INDEX PATH

      Filters: ((juracem.dm.verified = 'N' AND juracem.dm.scan_date >= 01/01/2012 ) AND
juracem.dm.scan_date <= TODAY )

   (1) Index Name: informix.imgmstr_type
      Index Keys: doc_type   (Serial, fragments: ALL)
      Lower Index Filter: juracem.dm.doc_type = ANY <subquery>

  2) juracem.di: INDEX PATH

   (1) Index Name: informix.imgid_prim
      Index Keys: id   (Serial, fragments: ALL)
      Lower Index Filter: juracem.di.id = juracem.dm.doc_key
NESTED LOOP JOIN

  3) juracem.dd: INDEX PATH

   (1) Index Name: informix.imgdoctype_dt
      Index Keys: doc_type   (Key-Only)  (Serial, fragments: ALL)
      Lower Index Filter: juracem.dm.doc_type = juracem.dd.doc_type
NESTED LOOP JOIN
```

```
Subquery:
---------
Estimated Cost: 11
Estimated # of Rows Returned: 5

  1) juracem.b: INDEX PATH

      Filters: juracem.b.can_scan = 'Y'

    (1) Index Name: informix.imgusrsec_prim
      Index Keys: id application doc_type   (Serial, fragments: ALL)
      Lower Index Filter: juracem.b.id = '470822'

  2) juracem.a: INDEX PATH

      Filters: juracem.a.doc_type LIKE REPLACE ((juracem.b.doc_type || '%' ), ' ' , '' )

    (1) Index Name: informix.imgdoctype_dt
      Index Keys: doc_type   (Key-Only)  (Serial, fragments: ALL)
NESTED LOOP JOIN
```

Figure 8. SET EXPLAIN output for SQL A

It becomes immediatelly apparent how inefficient the processing in this specific area is since every time an operator needs to search it takes aprroximately 4 minutes to get back results. The results of the SQL A query will serve a vital role in the system redesign.

From the SQL A results it is clear that the system needed 178,387 operations (estimated cost provided by the SQL Explain utility) to retrieve the matching records. From the practical point of view it took about 4 minutes. In the following steps an attempt will be made to modify the database structure as well as the SQL A query to get a better performance. The estimated cost will be monitored as well as the average time of each subseqent modification. Additionally, an attempt will be made to minimize both of the outcomes until an acceptable performance is achieved.

Transactional database restructuring

The first evident fact about the SQL A related tables is that they are not normilized. However, taking into consideration the extent the normalization would have on all data processing throughtout all the applications and reports, and not even knowing whether the normalization would have a possitive effect on queries (on the opposite, it's very likely that the normalization would worsen the performance), the focus will be geared more towards the database operations and indexing.

To make the research more efficient, the dbimg_mstr_rec table will be duplicated into dbimg_mstr2_rec (see Appendix C for the data import SQL) with the exception of doc_type column. In the new dbimg_mstr2_rec table, the doc_type column will be substituted with doc_type_id column (join to doc_type_id of dbimgdoctype_table), moving from char(20) to integer data type, from 20 bytes to only 4 bytes. The related indexes of dbimg_mstr2_rec table will also need to be modified. The column change should have a huge impact on the new index as we are decreasing the index record size by 16 bytes, which means more index records per block (block size is 2048 bytes), which in turn means less blocks to access and less I/O operations to retriveve all the records. The figure 9 reflects the table changes and adjustments to the original query SQL A. The resulting query is named SQL A2.

```
SELECT
   dm.uid AS uid,
   i.id AS id,
   i.ss_no AS ss_no,
   i.fullname AS fullname,
   '' AS did,
```

```
   " AS last_name,
   " AS first_name,
   dd.doc_type AS doc_type,
   dm.verified AS verified,
   dm.usr_scnnr_id AS usr_scnnr_id,
   dm.scan_date AS scan_date,
   dm.usr_vrfy_id AS usr_vrfy_id,
   dm.vrfy_date AS vrfy_date,
   " AS doc_name,
   dm.doc_path
FROM
   id_rec i,
   dbimg_mstr2_rec dm,
   dbimgdoctype_table dd
WHERE
   dm.doc_type_id = dd.doc_type_id AND
   dd.doc_type IN (
      SELECT
         a.doc_type
      FROM
         dbimgdoctype_table a, dbimg_usrsec_rec b
      WHERE
         a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
         b.id = '470822' AND
         b.can_scan = 'Y') AND
   dm.scan_date >='01-01-2012' AND
   dm.scan_date <=today AND
   dm.verified ='N' AND
   i.id = dm.doc_key
UNION
SELECT
   dm.uid AS uid,
   0 AS id,
   di.ss_no AS ss_no,
   " AS fullname,
   di.id AS did,
   di.last_name AS last_name,
   di.first_name AS first_name,
   dd.doc_type AS doc_type,
   dm.verified AS verified,
   dm.usr_scnnr_id AS usr_scnnr_id,
   dm.scan_date AS scan_date,
   dm.usr_vrfy_id AS usr_vrfy_id,
   dm.vrfy_date AS vrfy_date,
   di.doc_name AS doc_name,
   dm.doc_path
```

```
FROM
   dbimg_id_rec di,
   dbimg_mstr2_rec dm,
   dbimgdoctype_table dd
WHERE
   dm.doc_type_id = dd.doc_type_id AND
   dd.doc_type IN (
      SELECT
         a.doc_type
      FROM
         dbimgdoctype_table a, dbimg_usrsec_rec b
      WHERE
         a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
         b.id = '470822' AND
         b.can_scan = 'Y') AND
   dm.scan_date >='01-01-2012' AND
   dm.scan_date <=today AND
   dm.verified ='N' AND
   di.id = dm.doc_key
ORDER BY id DESC;
```

Figure 9. SQL A2 - after the move to doc_type_id column

Before running and collecting the analysis data we have to update statistics for the new dbimg_mstr2_rec table, as well as for the new column being used from dbimgdoctype_table. Figure 10 shows the specific statistics updates performed in preparation for the analysis of SQL A2.

```
UPDATE STATISTICS HIGH FOR TABLE dbimg_mstr2_rec (doc_type_id, doc_key,
verified, scan_date);
UPDATE STATISTICS HIGH FOR TABLE dbimgdoctype_table (doc_type, doc_type_id);
```

Figure 10. The performed statistics updates before analysis of SQL A2

Table 13 shows the results of the SQL 2 exceution. All the subsequent results will be recorded in this table as well so that the results can be easily compared.

By comparing the SQL A and SQL 2 results it is immediatelly apparent that the first set of modifications was successful. As shown in figure 11, the database optimizer used the new index imgmstr_type_2, reducing the total esimated cost down to 95,030 operations, and reducing the query execution time from 4 minutes down to an average of 1 minute. While this is a huge improvement, the performance goal has not been achieved yet.

```
...

Estimated Cost: 95030
Estimated # of Rows Returned: 13680
Temporary Files Required For: Order By

  1) juracem.dd: INDEX PATH

    (1) Index Name: informix.imgdoctype_dt
        Index Keys: doc_type   (Serial, fragments: ALL)
        Lower Index Filter: juracem.dd.doc_type = ANY <subquery>

  2) juracem.dm: INDEX PATH

      Filters: ((juracem.dm.verified = 'N' AND juracem.dm.scan_date >= 01/01/2012 ) AND
juracem.dm.scan_date <= TODAY )

    (1) Index Name: informix.imgmstr_type_2
        Index Keys: doc_type_id   (Serial, fragments: ALL)
        Lower Index Filter: juracem.dm.doc_type_id = juracem.dd.doc_type_id
NESTED LOOP JOIN

  3) juracem.i: INDEX PATH

    (1) Index Name: informix. 1956_6496
        Index Keys: id   (Serial, fragments: ALL)
        Lower Index Filter: juracem.i.id = juracem.dm.doc_key
NESTED LOOP JOIN
...
```

Figure 11. Excerpt of SET EXPLAIN output for SQL A2

In the next step, an attempt will be made to modify the join columns of the main query and the 2 sub-queries. Currently, as shown in fugure 9, the SQL A2 joins the sub-queries on doc_type column. Using join of doc_type_id should theoretically produce better results. The figure 12 shows the changes, from now on refered to as SQL A3.

```
...
 dbimgdoctype_table dd
WHERE
  dm.doc_type_id = dd.doc_type_id AND
  dd.doc_type_id IN (
    SELECT
      a.doc_type_id
    FROM
      dbimgdoctype_table a, dbimg_usrsec_rec b
    WHERE
      a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
      b.id = '470822' AND
      b.can_scan = 'Y') AND
  dm.scan_date >='01-01-2012' AND
...
```

Figure 12. Excerpt of SQL A3 modification

The table 13 shows mixed results from the comparison of SQL A2 and SQL A3. The estimated cost has been brought down to 95,028, but the execution has increased in time by almost 4 seconds. A further research would have to be conducted to find out what exactly happens during the execution of SQL A3, however, this is outside of the scope of this project. Since the main goal is to improve system's performance from the operator's point of view, the SQL A2 will be used for all subsequent modifications.

On the next step, the focus will be on indexes of the related tables. First, it is apparent that the filter operation "dm.verified ='N' " must be very memory and time consuming as there is not any related index in dbimg_mstr2_rec table. Figure 13 shows the newly added index.

```
table       dbimg_mstr2_rec
index
   unique imgmstr_prim on (uid,doc_key,doc_type,doc_path)
   imgmstr_key2 on (doc_path)
   imgmstr_key on (doc_key)
   imgmstr_type on (doc_type)
   imgmstr_vd_uid on (vrfy_date, uid)
   imgmstr_vrfy on (verified)
```

Figure 13. Indexes of dbimg_mstr2_rec table

The results of SQL A2 (shown as SQL A2i1 in table 13) are great. The database optimizer indeed used the new index as shown in figure 13. The total estimated cost has been lowered from 95,030 to only 17,999 operations, and most significantly, the query execution time has been lowered from 1 minute to under 1 second. This result, from operator's point view, is surely satisfactory, which means perfomance goals have been met.

```
...

Estimated Cost: 17999
Estimated # of Rows Returned: 13680
Temporary Files Required For: Order By

  1) juracem.dm: INDEX PATH

       Filters: (juracem.dm.scan_date >= 01/01/2012 AND juracem.dm.scan_date <= TODAY
)
```

```
  (1) Index Name: informix.imgmstr_vrfy
      Index Keys: verified   (Serial, fragments: ALL)
      Lower Index Filter: juracem.dm.verified = 'N'

 2) juracem.dd: INDEX PATH

  (1) Index Name: informix.imgdoctype_prim
      Index Keys: doc_type_id doc_type   (Key-Only)  (Serial, fragments: ALL)
      Lower Index Filter: (juracem.dm.doc_type_id = juracem.dd.doc_type_id AND
juracem.dd.doc_type = ANY <subquery> )
NESTED LOOP JOIN

 3) juracem.i: INDEX PATH

  (1) Index Name: informix. 1956_6496
      Index Keys: id   (Serial, fragments: ALL)
      Lower Index Filter: juracem.i.id = juracem.dm.doc_key
NESTED LOOP JOIN
...
```

Figure 14. Excerpt of SET EXPLAIN output for SQL A2i1

Even though the goal of the database structure and SQL redesign has been met, an attempt to even further reduce the estimated cost of the query and thus reduce the overall usage of the server resources will be made. Since the filter in the SQL A2 also references the column scan_date from dbimg_mstr2_rec, another index will be added to dbimg_mstr2_rec for column scan_date as shown in figure 15.

```
table        dbimg_mstr2_rec
index
   unique imgmstr_prim on (uid,doc_key,doc_type,doc_path)
   imgmstr_key2 on (doc_path)
   imgmstr_key on (doc_key)
   imgmstr_type on (doc_type)
   imgmstr_vd_uid on (vrfy_date, uid)
   imgmstr_vrfy on (verified)
   imgmstr_scndt on (scan_date)
```

Figure 15. Indexes of dbimg_mstr2_rec table

As expected, the results of SQL A2 with the new index imgmstr_scndt provide even better estimated cost as shown in table 13 under SQL A2i2. Figures 16 and 17 graphically depict the performance improvements.

Table 13. SQL performance results

SQL Performance	SQL A	SQL A2	SQL A3	SQL A2i1	SQL A2i2	SQL A4	SQL A5
Estimated Cost	178,387	95,030	95,028	17,999	17,930	16,850	1,179
Execution Time 1 (s)	237.042	60.317	63.991	0.115	0.119	0.088	0.087
Execution Time 2 (s)	233.467	60.204	64.388	0.120	0.129	0.119	0.094
Execution Time 3 (s)	235.525	60.401	63.268	0.133	0.123	0.094	0.086
Execution Time 4 (s)	231.297	60.022	63.737	0.123	0.113	0.082	0.105
Execution Time 5 (s)	225.743	60.786	64.445	0.118	0.115	0.092	0.089
Execution Time 6 (s)	235.967	59.588	63.902	0.115	0.119	0.097	0.082
Average Ex. Time (s)	233.174	60.220	63.955	0.121	0.120	0.095	0.091

Table 14. SQL descriptions

SQL	Description
SQL A	Original SQL and db tables
SQL A2	Modified dbimg_mstr_rec schema - use doc_type_id instead of doc_type; modified the index imgmstr_type to use the new column
SQL A3	Modified the subquery join to use doc_type_id instead of doc_type
SQL A2i1	The SQL A2 results after the added imgmstr_vrfy index
SQL A2i2	The SQL A2 results after the added imgmstr_scndt index
SQL A4	Modified SQL A2 to use id_rec only
SQL A5	Modified SQL A2 to use dbimg_id_rec only

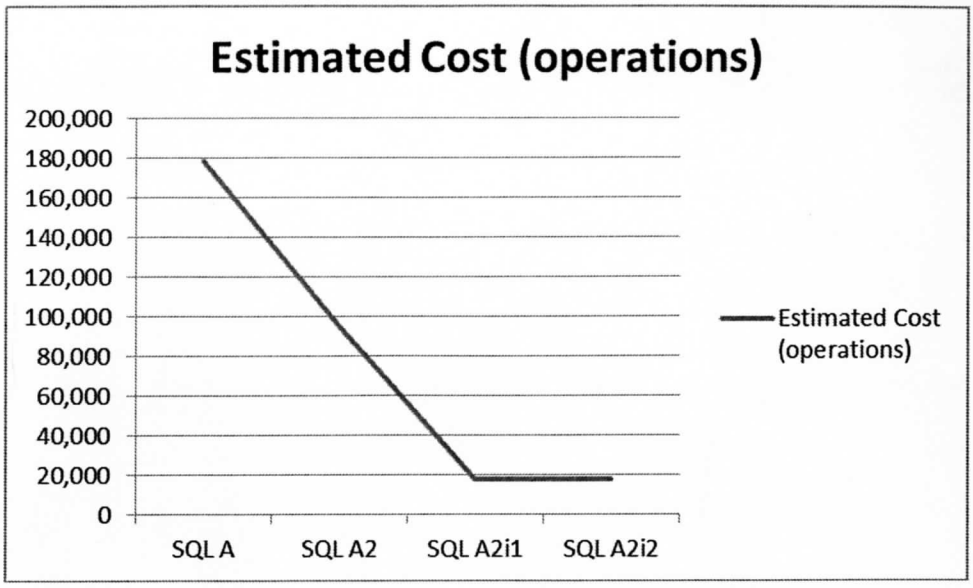


Figure 16. Graphical representation of the estimated cost results

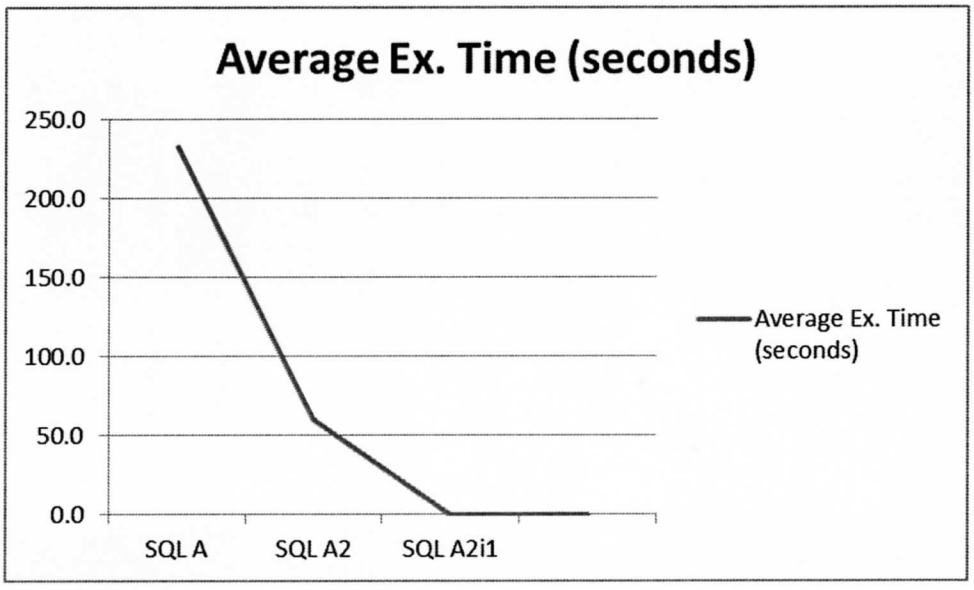


Figure 17. Graphical representation of the average execution time results

Again, the performance goal has been reached. The new design will significantly reduce the operator's wait time, from 4 minutes to under 1 second, and it is saving a

significant amount of server resources with the estimated cost droping from 178,387 down to 17,930.

While the main focus has been on addressing the server side modifications for achieving the performance goal, the scanning application could also be modified, in this case the search screen, and provide additional screen parameters that would narrow down the search. For example, if the operator specifies which entity table to use, id_rec versus dbimg_id_rec, than it could avoid using the SQL union clause and instead execute only one of the parts of the union. This would mainly further reduce the estimated cost of operations, as shown in table 13 in SQL A4 and SQL A5, to as low as 1,179 if dbimg_id_rec was selected. Figures 18 and 19 show the split of the final SQL A2 query.

```
{SQL A4}
SELECT
   dm.uid AS uid,
   i.id AS id,
   i.ss_no AS ss_no,
   i.fullname AS fullname,
   '' AS did,
   '' AS last_name,
   '' AS first_name,
   dd.doc_type AS doc_type,
   dm.verified AS verified,
   dm.usr_scnnr_id AS usr_scnnr_id,
   dm.scan_date AS scan_date,
   dm.usr_vrfy_id AS usr_vrfy_id,
   dm.vrfy_date AS vrfy_date,
   '' AS doc_name,
   dm.doc_path
FROM
   id_rec i,
   dbimg_mstr2_rec dm,
   dbimgdoctype_table dd
WHERE
   dm.doc_type_id = dd.doc_type_id AND
   dd.doc_type IN (
```

```
      SELECT
        a.doc_type
      FROM
        dbimgdoctype_table a, dbimg_usrsec_rec b
      WHERE
        a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
        b.id = '470822' AND
        b.can_scan = 'Y') AND
   dm.scan_date >='01-01-2012' AND
   dm.scan_date <=today AND
   dm.verified ='N' AND
   i.id = dm.doc_key
ORDER BY id DESC;
```

Figure 18. SQL A4 – using id_rec only

```
{SQL A5}
SELECT
   dm.uid AS uid,
   0 AS id,
   di.ss_no AS ss_no,
   '' AS fullname,
   di.id AS did,
   di.last_name AS last_name,
   di.first_name AS first_name,
   dd.doc_type AS doc_type,
   dm.verified AS verified,
   dm.usr_scnnr_id AS usr_scnnr_id,
   dm.scan_date AS scan_date,
   dm.usr_vrfy_id AS usr_vrfy_id,
   dm.vrfy_date AS vrfy_date,
   di.doc_name AS doc_name,
   dm.doc_path
FROM
   dbimg_id_rec di,
   dbimg_mstr2_rec dm,
   dbimgdoctype_table dd
WHERE
   dm.doc_type_id = dd.doc_type_id AND
   dd.doc_type IN (
      SELECT
        a.doc_type
      FROM
        dbimgdoctype_table a, dbimg_usrsec_rec b
```

```
    WHERE
      a.doc_type LIKE replace((b.doc_type || '%'), ' ','') AND
      b.id = '470822' AND
      b.can_scan = 'Y') AND
  dm.scan_date >='01-01-2012' AND
  dm.scan_date <=today AND
  dm.verified ='N' AND
  di.id = dm.doc_key
ORDER BY id DESC;
```

Figure 19. SQL A5 – using dbimg_id_rec only

Data warehouse design

As stated in the project requirements section, the main purpose of this data warehouse will be to permanently record and store system activities per individual operator. The next section will first analyze all the data requirements for extraction and storage, and then design the actual database structure of the new date warehouse.

It is already known that the management needs to be able to track each operator's activity, so a dimension for operators is needed. They need to be able to track the activities on daily basis, thus a dimension for time will be needed as well. Next, there is a need to track the type of operation, thus an operation dimension will be used. The management will also need to be able to have a granular access to the various document types, thus a document type dimension will be needed. Lastly, a document dimension to give the management capabilities to track action on any particular document entity will also be necessary. Table 15 summarizes the dimension needs, as well the specific column requirements.

Table 15. Dimensions needed for the data warehouse

Dimension type	Dimension name	Description/Reason	Fields needed	Field type
Time	time_dimension	To be able to track activities on daily basis, as well as by fiscal year and fiscal month	time_key	date
			fiscal_year	smallint
			fiscal_month	char(10)
Operation	operation_dimension	To be able to report by operation type	operation_key	char(1)
			operation_name	char(20)
Operator	operator_dimension	To be able to report by operator name	operator_id	char(20)
			operator_name	char(50)
Document Type	doc_type_dimension	To be able to report by specific document type	document_type_key	smallint
			document_type	char(20)
Document Entity	document_dimension	To be able to report all activities by specific document entity	document_key	char(12)
			document_id	char(25)

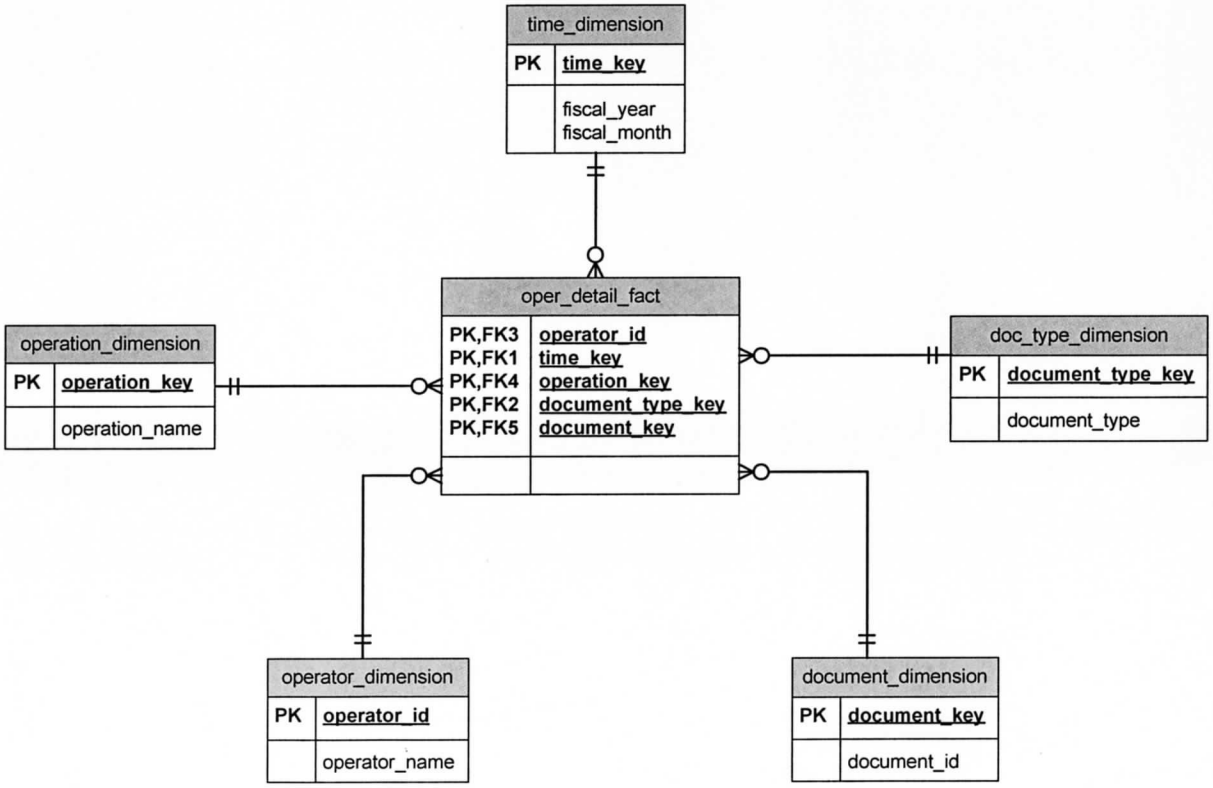


Figure 20. Data Warehouse Diagram

It's apparent from figure 20 that the suggested database structure for the data warehouse is designed to save storage resources, but at the same time the simple design will still ensure efficient processing and instant reporting capabilities.

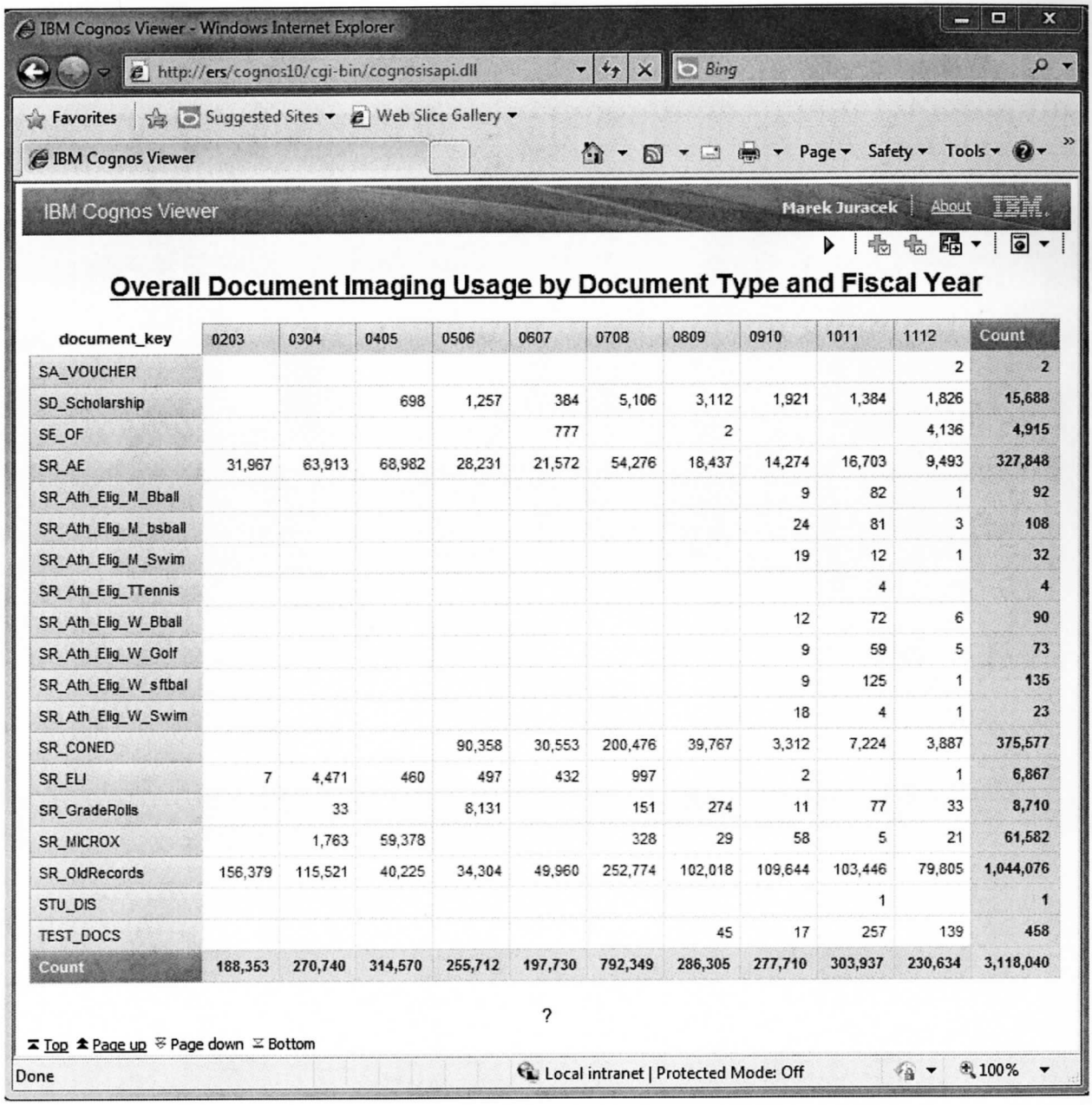
Appendix C shows all the queries used for creating the new tables within the new "docimg" database stored in the MYSQL database engine. Every major column in all tables has been indexed. This may slow down that nightly imports, but will speed up all subsequent reporting requests.

Appendix C also shows all the scripts used for the initial extraction of the data from the transactional database and import into the new data warehouse database. The extract/import was straight forward for all the dimension tables. The data gathering process for the fact table required more analysis though since the information is being imported from 2 different tables. Both tables, dbimg_mstr_rec and dbimg_stat_rec, have their own unique serial key. In order to distinguish in the data warehouse where the document entry came from, all the document keys from the dbimg_stat_rec will be prefixed with 'S'. Another decision that had to be made was to choose how to import documents that were processed and verified by the same operator. To provide extra reporting flexibility, 2 separate fact records for each of these activities will be used. One will be based on the scan_date and verified status of N, and the other will be based on the vrfy_date and verified status of either D, Y, or R. A user documentation and training will be needed to ensure that the reports based on this fact table get translated correctly.

The next section will describe all the reports that were created to address the reporting needs as outlined in the objectives of this project. In addition, in order to further free the

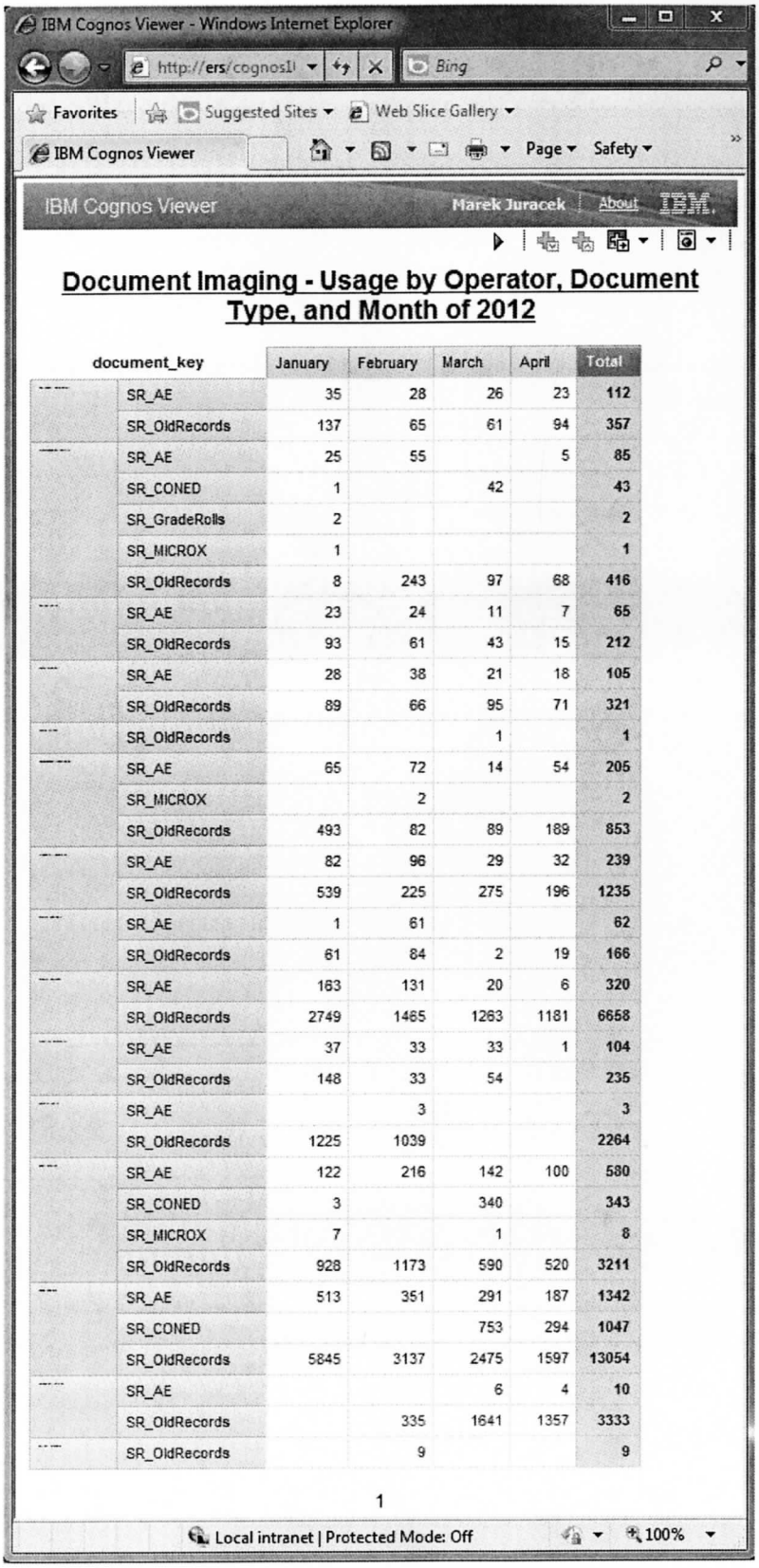
system resources of the transactional database server, the existing reports that were based on the transactional database were migrated into the data warehouse.

All of the reports that involve operator names are purposely modified to not to display the actual operator name.



document_key	0203	0304	0405	0506	0607	0708	0809	0910	1011	1112	Count
SA_VOUCHER										2	2
SD_Scholarship			698	1,257	384	5,106	3,112	1,921	1,384	1,826	15,688
SE_OF					777		2			4,136	4,915
SR_AE	31,967	63,913	68,982	28,231	21,572	54,276	18,437	14,274	16,703	9,493	327,848
SR_Ath_Elig_M_Bball								9	82	1	92
SR_Ath_Elig_M_bsball								24	81	3	108
SR_Ath_Elig_M_Swim								19	12	1	32
SR_Ath_Elig_TTennis									4		4
SR_Ath_Elig_W_Bball								12	72	6	90
SR_Ath_Elig_W_Golf								9	59	5	73
SR_Ath_Elig_W_sftbal								9	125	1	135
SR_Ath_Elig_W_Swim								18	4	1	23
SR_CONED				90,358	30,553	200,476	39,767	3,312	7,224	3,887	375,577
SR_ELI	7	4,471	460	497	432	997		2		1	6,867
SR_GradeRolls		33		8,131		151	274	11	77	33	8,710
SR_MICROX		1,763	59,378			328	29	58	5	21	61,582
SR_OldRecords	156,379	115,521	40,225	34,304	49,960	252,774	102,018	109,644	103,446	79,805	1,044,076
STU_DIS									1		1
TEST_DOCS							45	17	257	139	458
Count	188,353	270,740	314,570	255,712	197,730	792,349	286,305	277,710	303,937	230,634	3,118,040

Figure 21. Cognos Report – Overall document imaging usage



document_key		January	February	March	April	Total
	SR_AE	35	28	26	23	112
	SR_OldRecords	137	65	61	94	357
	SR_AE	25	55		5	85
	SR_CONED	1		42		43
	SR_GradeRolls	2				2
	SR_MICROX	1				1
	SR_OldRecords	8	243	97	68	416
	SR_AE	23	24	11	7	65
	SR_OldRecords	93	61	43	15	212
	SR_AE	28	38	21	18	105
	SR_OldRecords	89	66	95	71	321
	SR_OldRecords			1		1
	SR_AE	65	72	14	54	205
	SR_MICROX		2			2
	SR_OldRecords	493	82	89	189	853
	SR_AE	82	96	29	32	239
	SR_OldRecords	539	225	275	196	1235
	SR_AE	1	61			62
	SR_OldRecords	61	84	2	19	166
	SR_AE	163	131	20	6	320
	SR_OldRecords	2749	1465	1263	1181	6658
	SR_AE	37	33	33	1	104
	SR_OldRecords	148	33	54		235
	SR_AE		3			3
	SR_OldRecords	1225	1039			2264
	SR_AE	122	216	142	100	580
	SR_CONED	3		340		343
	SR_MICROX	7		1		8
	SR_OldRecords	928	1173	590	520	3211
	SR_AE	513	351	291	187	1342
	SR_CONED			753	294	1047
	SR_OldRecords	5845	3137	2475	1597	13054
	SR_AE			6	4	10
	SR_OldRecords		335	1641	1357	3333
	SR_OldRecords		9			9

1

Figure 22. Cognos Report – Usage by operator, document type, and month of 2012

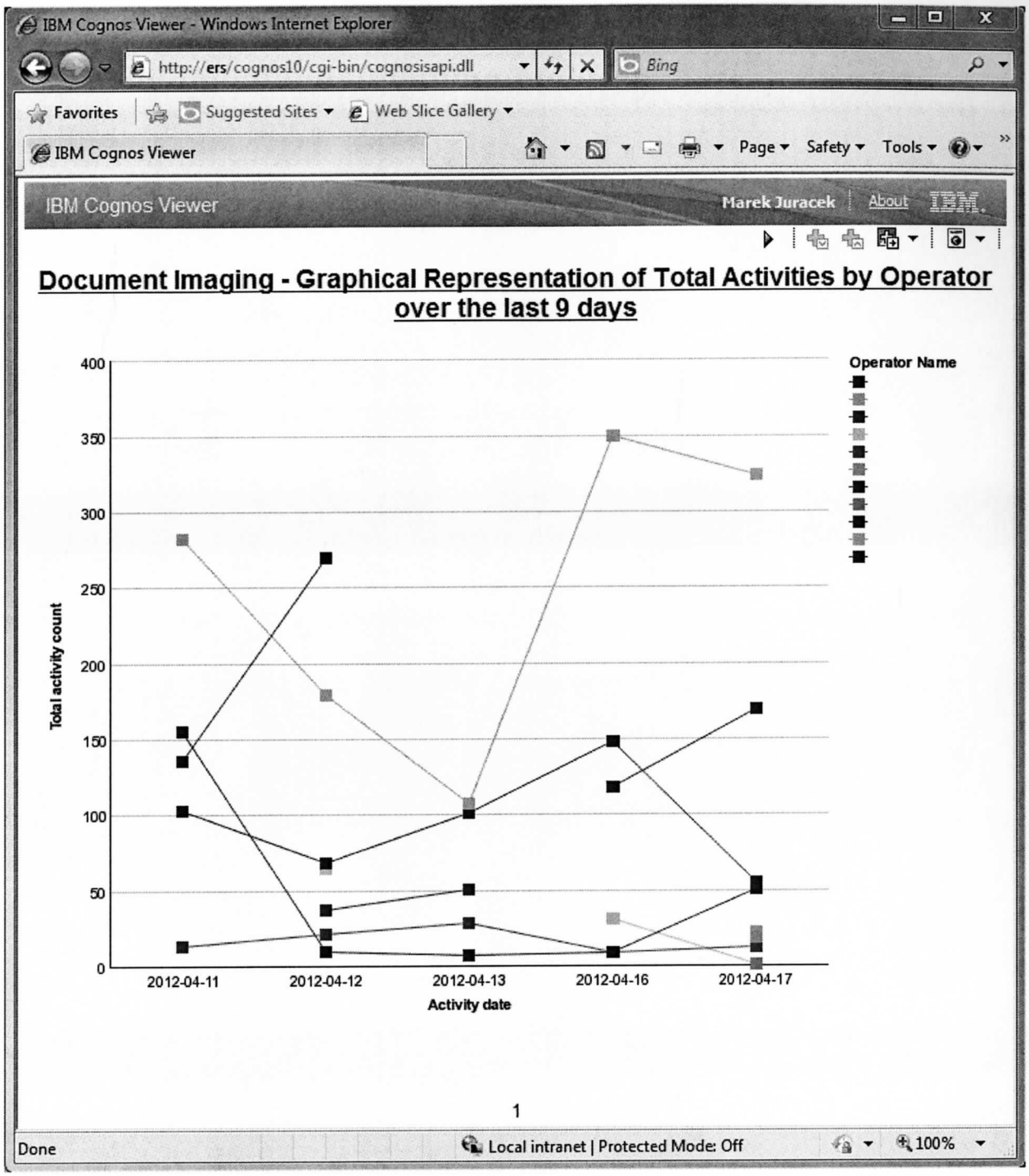
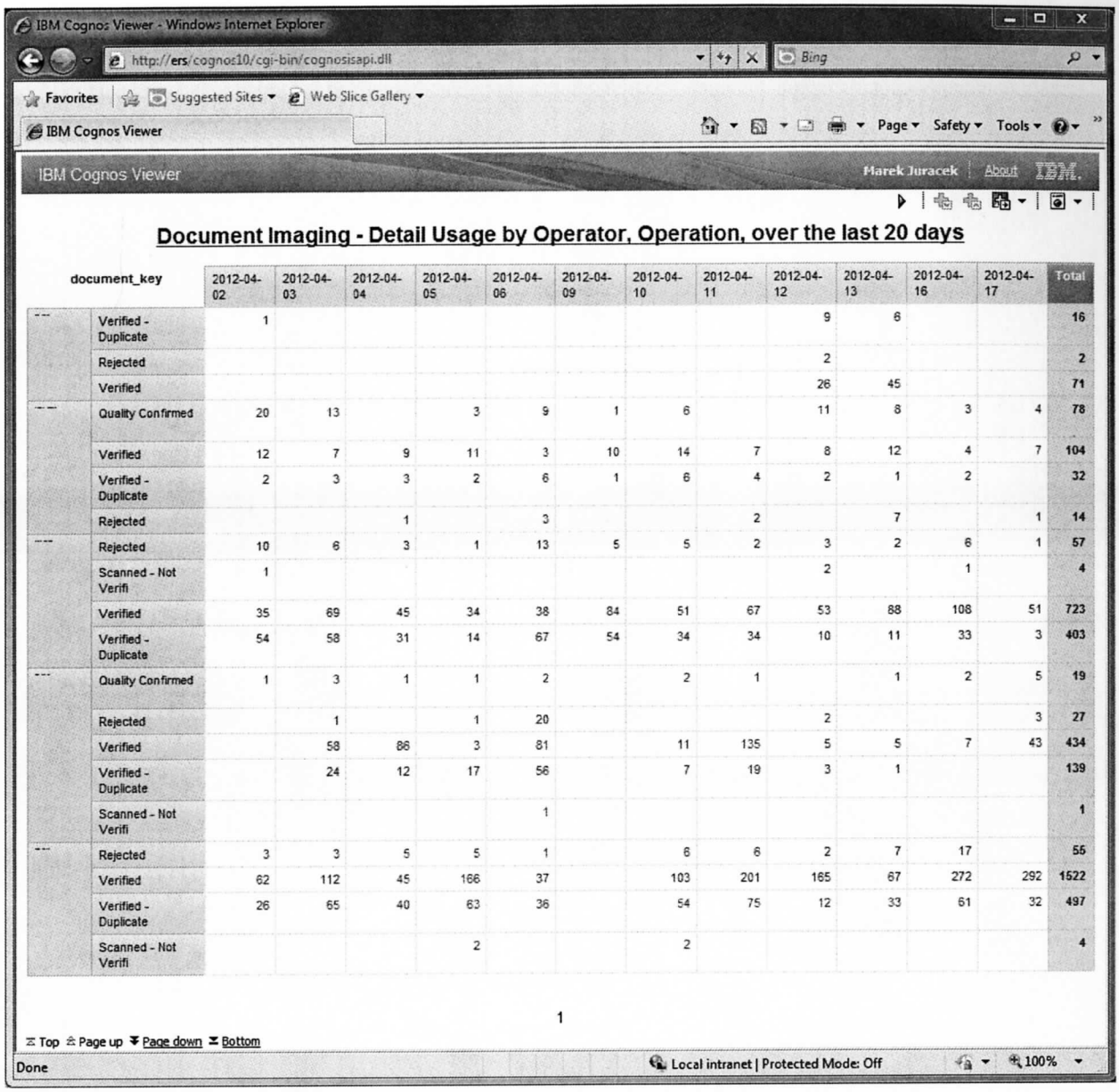


Figure 23. Cognos Report – Graphical representation of total activities by operator over the last 9 days



Document Imaging - Detail Usage by Operator, Operation, over the last 20 days

document_key		2012-04-02	2012-04-03	2012-04-04	2012-04-05	2012-04-06	2012-04-09	2012-04-10	2012-04-11	2012-04-12	2012-04-13	2012-04-16	2012-04-17	Total
---	Verified - Duplicate	1								9	6			16
	Rejected									2				2
	Verified									26	45			71
---	Quality Confirmed	20	13		3	9	1	6		11	8	3	4	78
	Verified	12	7	9	11	3	10	14	7	8	12	4	7	104
	Verified - Duplicate	2	3	3	2	6	1	6	4	2	1	2		32
	Rejected			1		3			2		7		1	14
---	Rejected	10	6	3	1	13	5	5	2	3	2	6	1	57
	Scanned - Not Verifi	1								2		1		4
	Verified	35	69	45	34	38	84	51	67	53	88	108	51	723
	Verified - Duplicate	54	58	31	14	67	54	34	34	10	11	33	3	403
---	Quality Confirmed	1	3	1	1	2		2	1		1	2	5	19
	Rejected		1		1	20				2			3	27
	Verified		58	86	3	81		11	135	5	5	7	43	434
	Verified - Duplicate		24	12	17	56		7	19	3	1			139
	Scanned - Not Verifi					1								1
---	Rejected	3	3	5	5	1		6	6	2	7	17		55
	Verified	62	112	45	166	37		103	201	165	67	272	292	1522
	Verified - Duplicate	26	65	40	63	36		54	75	12	33	61	32	497
	Scanned - Not Verifi				2			2						4

1

Figure 24. Cognos Report – Detail usage by operator, operation, over the last 20 days

CHAPTER 4

CASE STUDY (RESULTS AND DISCUSSION)

One of the main goals of this project was to improve the efficiency of the document imaging system, which would lead to a higher imaging processing, and ultimately to a more cost effective operations. As demonstrated, some of the system queries could take about 4 minutes to execute. By redesigning the current imaging system the execution time was brought down to less than 1 second. The next section will estimate the average time savings per day, month, and year. The estimation will be based on the data processed from the period of 2008 to 2011.

In 2008, out of 362,131 processed documents, there were 144,184 documents that were scanned and verified by different operators. This means, that the documents were most likely processed in a batch mode, where one operator's daily task was to only scan, and other operator's daily task was to verify only. Under the assumption that the operator that verified documents had to wait an average of 1 minute for a document to be found and loaded, the total wait time just for this operation that year would be 144,184 minutes, which is 2,403 hours. With 10 operators verifying during 246 days during the year of 2008, the daily average time spent on the documents to be found and loaded was about 1 hour (20 hours a month).

Going down from 4 minutes of wait time to less than 1 second truly presents huge operation savings, and is meeting the performance goals set in this project.

In 2011, the number of end users that used the document imaging for retrieval purposes nearly doubled to about 400 total users. Using a rough estimate of 10 document retrievals per day per end user (the usage is much higher for departments such as Financial Aid, but much lower for users that only occasionally access archives, etc.), using 230 working day calendar, and assuming each search and retrieval takes about 20 seconds, we total to about 511 wasted hours per year. Again, there are huge time savings by driving the system to retrieve the documents within a second.

Not only time and operation cost will be saved by improving the efficiency of the system, but also an increment in student satisfaction will be seen by providing faster services to our students.

The new data warehouse more than enough satisfies all the current reporting needs. The managers will no longer lose track of all their operation statistics, allowing them to further analyze the operations and perfect their document imaging plans to fit immediate or future needs.

In addition to the needed reports, operation managers of other departments (such as Facilities department) can utilize the available data to estimate, for example, when will the buildings and rooms that are currently used for storing various document archives be available again for further usage (office space, classrooms, etc.).

CHAPTER 5

CONCLUSIONS

The main objectives of this project were to improve the efficiency of the Document Imaging system and to provide the additional reporting capabilities. The project achieved both of the objectives. As discussed in the case study section, the college will be able to save many hours every month due to the improved efficiency of the imaging system. Also, the new document imaging data warehouse and the related Cognos reports fully fulfill the reporting needs as well as prepare the reporting environment for any future reporting needs.

Due to the time limitation of the project, only the most critical areas of the system that contribute to low efficiency have been addressed. This project, the research and the implementation, will serve as a guide for any subsequent modifications and thus will simplify the future system development.

This project proves what a big role software maintenance plays in Information Technology. Due to the limited IT resources, the IT staff cannot evaluate every application that was written internally on regular basis. Instead, there is a need to rely on continuous feedback from the end users to help assess the application and maintenance performance needs. The college has recognized the need for improvement in this area and started working on setting up an environment that would allow IT to automatically handle such communication. It has been successfully implemented and deployed through the Microsoft's System Center Configuration Manager and System Center Operations Manager throughout the college (with the exception of the MAC environment), and are looking at setting up the

application performance monitoring for the desktop systems (servers are already monitored) in the near future.

The new reporting environment provides an excellent way of keeping up to date with the daily usage of the Document Imaging system, as well as provides many ways to analyze the overall growth of the system. The reporting will no longer affect, and be affected (in case of deleted records) by the transactional database used for the system operations.

In conclusion, this project has allowed, and will continue to allow shaping the systems operation environment within the college. It has made possible to identify other system deficiencies, thus setting up a stepping stone for future system modifications and report management.

REFERENCES

Elmasri, R., Navathe, S. (2007). *Fundamentals of database systems*. 5th ed. Boston: Pearson Education.

Giovinazzo, W. (2000). *Object-oriented data warehouse design: Building a star schema*. New Jersey: Prentice Hall.

IBM Corporation. (2010). *IBM cognos framework manager: Version 10.1.0*. Retrieved on March 1st, 2012, from http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_fm.pdf

IBM Corporation. (2010). *IBM cognos report studio: Version 10.1.0*. Retrieved on March 1st, 2012, from http://public.dhe.ibm.com/software/data/cognos/documentation/docs/en/10.1.0/ug_cr_rptstd.pdf

IBM Corporation. (2010). *IBM Informix version 11.50: IBM Informix dynamic server performance guide*. Retrieved on March 1st, 2012, from http://publibfp.dhe.ibm.com/epubs/pdf/c2736180.pdf

Wilson, Ch. (2011). *MySQL 5.5 Reference manual.* Retrieved on March 1st, 2012, from http://dev.mysql.com/doc/refman/5.5/en/index.html

APPENDICES

APPENDIX A: WORK BREAKDOWN STRUCTURE (WBS)

1	**Initiating**
1.1	Problem definition
1.2	Initiating completed
2	**Planning**
2.1	Develop scope statement
2.2	Review current system
2.3	Create WBS
2.4	Select Tools
2.4.1	Select data warehouse database
2.4.2	Select reporting tools
2.5	Create schedule
2.5.1	Refine timeline
2.5.2	Gantt Chart
2.5.3	Planning completed
3	**Executing**
3.1	Redesign Document Imaging
3.1.1	Redesign Tables Structure
3.1.2	Redesign SQLs
3.1.3	Measure execution times
3.1.4	Document Results
3.1.5	Document new design
3.1.6	Redesign Completed
3.2	Design warehouse database
3.2.1	Identify data to extract
3.2.2	Design warehouse tables

APPENDIX B: GANTT CHART

ID	WBS	Task Name	Duration	Start	Finish
1	1	**Initiating**	**2 days**	**on 10/10/11**	**Tue 10/11/11**
2	1.1	Problem definition	2 days	Mon 10/10/11	Tue 10/11/11
3	1.2	Initiating completed	0 days	Tue 10/11/11	Tue 10/11/11
4	2	**Planning**	**40 days**	**Wed 10/12/11**	**Tue 12/6/11**
5	2.1	Develop scope statement	10 days	Wed 10/12/11	Tue 10/25/11
6	2.2	Review current system	5 days	Wed 10/26/11	Tue 11/1/11
7	2.3	Create WBS	5 days	Wed 11/2/11	Tue 11/8/11
8	2.4	**Select Tools**	**5 days**	**Wed 11/9/11**	**Tue 11/15/11**
9	2.4.1	Select data warehouse database	3 days	Wed 11/9/11	Fri 11/11/11
10	2.4.2	Select reporting tools	2 days	Mon 11/14/11	Tue 11/15/11
11	2.5	**Create schedule**	**15 days**	**Wed 11/16/11**	**Tue 12/6/11**
12	2.5.1	Refine timeline	5 days	Wed 11/16/11	Tue 11/22/11
13	2.5.2	Gantt Chart	10 days	Wed 11/23/11	Tue 12/6/11
14	2.5.3	Planning completed	0 days	Tue 12/6/11	Tue 12/6/11
15	3	**Executing**	**65 days**	**Wed 12/7/11**	**Tue 3/6/12**
16	3.1	**Redesign Document Imaging**	**35 days**	**Wed 12/7/11**	**Tue 1/24/12**
17	3.1.1	Redesign Tables Structure	25 days	Wed 12/7/11	Tue 1/10/12
18	3.1.2	Redesign SQLs	5 days	Wed 1/11/12	Tue 1/17/12
19	3.1.3	Measure execution times	1 day	Wed 1/18/12	Wed 1/18/12
20	3.1.4	Document Results	2 days	Thu 1/19/12	Fri 1/20/12
21	3.1.5	Document new design	2 days	Mon 1/23/12	Tue 1/24/12
22	3.1.6	Redesign Completed	0 days	Tue 1/24/12	Tue 1/24/12
23	3.2	**Design warehouse database**	**29 days**	**Wed 1/25/12**	**Mon 3/5/12**
24	3.2.1	Indetify data to extract	2 days	Wed 1/25/12	Thu 1/26/12
25	3.2.2	Design warehouse tables	10 days	Fri 1/27/12	Thu 2/9/12
26	3.2.3	Release data warehouse	1 day	Fri 2/10/12	Fri 2/10/12
27	3.2.4	Database warehouse completed	0 days	Fri 2/10/12	Fri 2/10/12
28	3.2.5	**Create reports**	**13 days**	**Mon 2/13/12**	**Wed 2/29/12**
29	3.2.5.1	Setup Cognos framework	5 days	Mon 2/13/12	Fri 2/17/12
30	3.2.5.2	Desing Cognos Reports	5 days	Mon 2/20/12	Fri 2/24/12
31	3.2.5.3	Test reports	.2 days	Mon 2/27/12	Tue 2/28/12
32	3.2.5.4	Deploy reports	1 day	Wed 2/29/12	Wed 2/29/12
33	3.2.6	Document the data warehouse	3 days	Thu 3/1/12	Mon 3/5/12
34	3.3	Training	1 day	Tue 3/6/12	Tue 3/6/12
35	3.4	Executing completed	0 days	Tue 3/6/12	Tue 3/6/12
36	4	**Closing**	**21 days**	**Wed 3/7/12**	**Wed 4/4/12**
37	4.1	Lessons Learned	1 day	Wed 3/7/12	Wed 3/7/12
38	4.2	Prepare Final Report	10 days	Thu 3/8/12	Wed 3/21/12
39	4.3	Prepare Project Presentation	10 days	Thu 3/22/12	Wed 4/4/12
40	4.4	Project completed	0 days	Wed 4/4/12	Wed 4/4/12

APPENDIX C: SYSTEM TECHNICAL DOCUMENTATION

SQLs used for transactional database restructuring

SQL to alter the dbimg_mstr_rec for no-append mode – add doc_description column:

```
ALTER
TABLE dbimg_mstr_rec
ADD doc_decsription char(25) default " "
BEFORE doc_path;
```

SQL to alter the dbimgdoctype_table for no-append mode – add retention_prd column for the newly added retention management functionality:

```
ALTER
TABLE dbimgdoctype_table
ADD retention_prd smallint default 0;
```

SQL to populate dbimg_mstr2_rec:

```
SELECT
        a.doc_key,
        b.doc_type_id,
        a.doc_path,
        a.verified,
        a.usr_scnnr_id,
```

```
        a.usr_vrfy_id,
        a.scan_date,
        a.vrfy_date,
        a.previous_path,
        a.temp_doc_key
FROM
        dbimg_mstr_rec a,
        dbimgdoctype_table b
WHERE
        a.doc_type = b.doc_type
INTO TEMP temp_a WITH NO LOG;

INSERT INTO        dbimg_mstr2_rec
        (doc_key,
        doc_type_id,
        doc_path,
        verified,
        usr_scnnr_id,
        usr_vrfy_id,
        scan_date,
        vrfy_date,
        previous_path,
        temp_doc_key)
SELECT * FROM temp_a;
```

SQL creation statements for the Document Imaging repository database (MYSQL):

```
CREATE
  TABLE docimg.oper_detail_fact
  (
    operator_id CHAR(20) NOT NULL,
    time_key DATE NOT NULL,
    operation_key CHAR(1) NOT NULL,
    document_type_key SMALLINT NOT NULL,
    document_key CHAR(12) NOT NULL,
   PRIMARY KEY (operator_id, time_key, operation_key, document_type_key,
document_key),
    INDEX ix1 UNIQUE (operator_id),
    INDEX ix2 UNIQUE (time_key),
    INDEX ix3 UNIQUE (operation_key),
    INDEX ix4 UNIQUE (document_type_key),
    INDEX ix5 UNIQUE (document_key)\
  )
  ENGINE=MyISAM DEFAULT CHARSET=latin1
```

```
CREATE
  TABLE docimg.operation_dimension
  (
    operation_key CHAR(1) NOT NULL,
    operation_name CHAR(20) NOT NULL,
    PRIMARY KEY (operation_key)
  )
```

```
  ENGINE=MyISAM DEFAULT CHARSET=latin1


CREATE
  TABLE docimg.time_dimension
  (
    time_key DATE NOT NULL,
    fiscal_year SMALLINT NOT NULL,
    fiscal_month CHAR(10) NOT NULL,
    PRIMARY KEY (time_key),
    INDEX ix1 (fiscal_year, fiscal_month)
  )
  ENGINE=MyISAM DEFAULT CHARSET=latin1


CREATE
  TABLE docimg.doc_type_dimension
  (
    document_type_key SMALLINT NOT NULL,
    document_type CHAR(20) NOT NULL,
    PRIMARY KEY (document_type_key),
    CONSTRAINT ix1 UNIQUE (document_type)
  )
  ENGINE=MyISAM DEFAULT CHARSET=latin1


CREATE
  TABLE docimg.document_dimension
```

```
(
  document_key CHAR(12) NOT NULL,
  document_id CHAR(25) NOT NULL,
  PRIMARY KEY (document_key),
  INDEX ix1 (document_id)
)
ENGINE=MyISAM DEFAULT CHARSET=latin1
```

```
CREATE
  TABLE docimg.operator_dimension
  (
    operator_id CHAR(20) NOT NULL,
    operator_name CHAR(50) NOT NULL,
    PRIMARY KEY (operator_id),
    INDEX ix1 (operator_name)
  )
  ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Scripts used to extract the data from the Document Imaging transaction database into the data warehouse database:

```
#!/bin/sh
#####################################################################
## This script is used to extract document master table          ##
## from cars/informix database and import the data into          ##
## docimg/MYSQL database                                         ##
## --                                                            ##
## ./dw_docimg_doc                                               ##
##                                                               ##
#####################################################################

datfil=/work/juracem/dw_docimg/doc_table.dat
mysqltbl=document_dimension

echo "Extracting Document Master into " $datfil
##### Now extract the information
echo "
SET ISOLATION TO DIRTY READ;
unload to $datfil
select
 trim(to_char(uid)),
 doc_key
from dbimg_mstr_rec
union
select
 'S' || uid,
 doc_id
from dbimg_stat_rec
;
" | isql cars

#### End of the extraction sql and execution ####

#########################################################
## Now we load the extracted information into mysql ##
#########################################################

echo "
load data LOCAL infile '$datfil'
  REPLACE into table $mysqltbl fields terminated by '|';
  " | /mysql/mysql/bin/mysql docimg
```

```
######## End of MYSQL data load    ###########
#################################################
```

```
#!/bin/sh
#########################################################################
## This script is used to extract document type table          ##
## from cars/informix database and import the data into         ##
## docimg/MYSQL database                                  ##
## --                                          ##
## ./dw_docimg_doctype                                    ##
##                                             ##
#########################################################################

datfil=/work/juracem/dw_docimg/doctype_table.dat
mysqltbl=doc_type_dimension

echo "Extracting Document Type into " $datfil
##### Now extract the information
echo "
SET ISOLATION TO DIRTY READ;
unload to $datfil
select
 doc_type_id
 ,doc_type
from dbimgdoctype_table
;

" | isql cars


#### End of the extraction sql and execution ####

#########################################################
## Now we load the extracted information into mysql ##
#########################################################


echo "
load data LOCAL infile '$datfil'
  REPLACE into table $mysqltbl fields terminated by '|';
  " | /mysql/mysql/bin/mysql docimg

########  End of MYSQL data load    ############
#############################################
#!/bin/csh -f | more
```

```
#!/bin/sh
#########################################################################
## This script is used to extract fact info                    ##
## from cars/informix database and import the data into        ##
## docimg/MYSQL database                                       ##
## --                                                          ##
## ./dw_docimg_fact                                            ##
##                                                             ##
#########################################################################

set datfil=/work/juracem/dw_docimg/fact_table.dat
set mysqltbl=oper_detail_fact

goto loaddata

echo "Extracting fact info" $datfil
##### Now extract the information

isql cars << END_SQL  > & ztmp.err

SET ISOLATION TO DIRTY READ;
unload to $datfil
select a.usr_scnnr_id, to_char(a.scan_date, '%Y-%m-%d'), a.verified, b.doc_type_id,
to_char(a.uid)
from
 dbimg_mstr_rec a, dbimgdoctype_table b
where
  a.verified in ('N') and
  a.doc_type = b.doc_type
union
select a.usr_scnnr_id, to_char(a.scan_date, '%Y-%m-%d'), a.verified, b.doc_type_id,
to_char(a.uid)
from
 dbimg_mstr_rec a, dbimgdoctype_table b
where
  a.verified in ('R', 'Y', 'D') and
  a.doc_type = b.doc_type
union
select a.usr_id, to_char(a.perform_date, '%Y-%m-%d'), a.work_ctgr, a.doc_type_id, 'S' || a.uid
from
 dbimg_stat_rec a
where
  a.work_ctgr in ('Q', 'P')
;

END_SQL
```

```
## label here
loaddata:
#### End of the extraction sql and execution ####
echo "loading data"

#########################################################
## Now we load the extracted information into mysql ##
#########################################################

/mysql/mysql/bin/mysql docimg << ADD_MYSQL

load data LOCAL infile '$datfil'
  REPLACE into table $mysqltbl fields terminated by '|';

ADD_MYSQL
########  End of MYSQL data load    ############
################################################
```

```
#!/bin/sh
#########################################################################
## This script is used to extract operation table              ##
## from cars/informix database and import the data into        ##
## docimg/MYSQL database                                       ##
## --                                                          ##
## ./dw_docimg_operation                                       ##
##                                                             ##
#########################################################################

datfil=/work/juracem/dw_docimg/operation_table.dat
mysqltbl=operation_dimension

echo "Extracting Operation Info into " $datfil
##### Now extract the information
echo "
SET ISOLATION TO DIRTY READ;
unload to $datfil
select
  grouping
 ,txt
from db_lookup_table
where
  code = 'DOCIMG_VERIFIED'
;

" | isql cars


#### End of the extraction sql and execution ####

#########################################################
## Now we load the extracted information into mysql ##
#########################################################


echo "
load data LOCAL infile '$datfil'
  REPLACE into table $mysqltbl fields terminated by '|';
  " | /mysql/mysql/bin/mysql docimg

########  End of MYSQL data load   ###########
#############################################
```

```
#!/bin/sh
#########################################################################
## This script is used to extract operator table                ##
## from cars/informix database and import the data into         ##
## docimg/MYSQL database                                        ##
## --                                                           ##
## ./dw_docimg_operator                                         ##
##                                                              ##
#########################################################################

datfil=/work/juracem/dw_docimg/operator.dat
mysqltbl=operator_dimension

echo "Extracting Operator into " $datfil
##### Now extract the information
echo "
SET ISOLATION TO DIRTY READ;
unload to $datfil
select
 login_id
 ,fullname
from dbimg_security_rec
;

" | isql cars


#### End of the extraction sql and execution ####

#########################################################
## Now we load the extracted information into mysql ##
#########################################################


echo "
load data LOCAL infile '$datfil'
  REPLACE into table $mysqltbl fields terminated by '|';
  " | /mysql/mysql/bin/mysql docimg

########  End of MYSQL data load   ############
#################################################
```

```
#!/bin/sh
#########################################################################
## This script is used to extract time info                    ##
## from cars/informix database and import the data into        ##
## docimg/MYSQL database                                       ##
## --                                                ##
## ./dw_docimg_time                                        ##
##                                                   ##
#########################################################################

datfil=/work/juracem/dw_docimg/time_table.dat
mysqltbl=time_dimension

echo "Extracting Time info into " $datfil
##### Now extract the information
echo "
SET ISOLATION TO DIRTY READ;
unload to $datfil
select distinct(to_char(scan_date, '%Y-%m-%d')) as tday,
 case when month(scan_date)>6 then (to_char(scan_date, '%y') ||
substring(to_char(year(scan_date) + 1) from 3 for 2))
  else (substring(to_char(year(scan_date) - 1) from 3 for 2) || to_char(scan_date, '%y')) end as
fiscal_year,
   to_char(scan_date, '%B') as fiscal_month
from dbimg_mstr_rec
union
select distinct(to_char(vrfy_date, '%Y-%m-%d')) as tday,
 case when month(vrfy_date)>6 then (to_char(vrfy_date, '%y') ||
substring(to_char(year(vrfy_date) + 1) from 3 for 2))
   else (substring(to_char(year(vrfy_date) - 1) from 3 for 2) || to_char(vrfy_date, '%y')) end as
fiscal_year,
    to_char(vrfy_date, '%B') as fiscal_month
from dbimg_mstr_rec
union
select distinct(to_char(perform_date, '%Y-%m-%d')) as tday,
 case when month(perform_date)>6 then (to_char(perform_date,'%y') ||
substring(to_char(year(perform_date) + 1) from 3 for 2))
  else (substring(to_char(year(perform_date) - 1) from 3 for 2) || to_char(perform_date, '%y'))
end as fiscal_year,
   to_char(perform_date, '%B') as fiscal_month
from dbimg_stat_rec
;

" | isql cars
```

```
#### End of the extraction sql and execution ####

#######################################################
## Now we load the extracted information into mysql ##
#######################################################


echo "
load data LOCAL infile '$datfil'
  REPLACE into table $mysqltbl fields terminated by '|';
  " | /mysql/mysql/bin/mysql docimg

########  End of MYSQL data load    ###########
#################################################
```