Melissa B. Florio. The Development of a Conversion Model for Programmers Converting a VSAM File to Oracle Tables. A Master's paper for the M.S. in I.S. degree. April, 2002. 74 pages. Advisor: Stephanie W. Haas

This paper describes the development and evaluation of a conversion model for converting a VSAM file to Oracle tables. The conversion model was developed for programmers within the Student Information Group at the University of North Carolina at Chapel Hill. The purpose for the developing the conversion model was to provide step-by-step instructions to assist the programmers in the conversion process. To assist in the development of the conversion model, feedback from the programmers both before and after the development of the conversion model was gathered. Other methods used in the development of the conversion model included conducting a case study of previous conversions and implementing an actual VSAM to Oracle conversion.

Headings:

Database--Management--Systems

Databases

Database Conversion

Information Systems--Design

THE DEVELOPMENT OF A CONVERSION MODEL FOR PROGRAMMERS CONVERTING A VSAM FILE TO ORACLE TABLES

by
Melissa Florio

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April, 2002

Approved by:

_____

Advisor

Table of Contents

1.0 Introduction..... 3
2.0 Project Description..... 4
2.1 Purpose..... 4
2.2 Scope..... 4
2.3 Goals and Objectives..... 5
3.0 VSAM vs. Relational DB Models..... 6
3.1 What is VSAM?..... 6
3.2 Similarities and Differences..... 7
3.3 Advantages of Relational..... 9
4.0 Literature Review..... 11
4.1 Reengineering Non-relational to Relational..... 11
4.2 VSAM Conversion for COBOL..... 13
4.3 Transition of a Very Large Database..... 15
4.4 VSAM to Relational..... 17
5.0 Methodology..... 20
5.1 Developing the Conversion Model..... 20
5.2 Evaluating the Conversion Model..... 22
5.3 Testing the Graduate Online Application Conversion..... 23
6.0 Current System..... 27
6.1 Online System Overview..... 27
6.2 Online System Detailed Description..... 28
6.3 Batch System Description..... 28
7.0 Database Design..... 30
8.0 Results..... 33
8.1 Evaluation of Stakeholders' Wish List..... 33
8.2 Evaluation of the Conversion Model..... 34
8.3 Testing the Graduate Online Application Conversion..... 37
9.0 Conclusion..... 41
References..... 43
Appendices..... 44
Appendix A..... 45
Appendix B..... 48
Appendix C..... 49
Appendix D..... 50

1.0 Introduction

The Student Information Group (SI) within Administrative Information Services (AIS) at the University of North Carolina at Chapel Hill currently supports the graduate school online application for admission. The graduate online application was the first online admission application to be developed by the SI group. The application was developed in early 1997 and soon after eight more online admission applications followed, including online applications for undergraduate admissions, continuing studies, and summer school. In 1999, the online application was modified to handle credit card payments for application fees. Again, this was the first online application and online process to allow credit card payments over the web. The Graduate School has always been the “pioneer” and “Guinea pig” when it comes to new technology. So why stop now! This is why the graduate school online application has been chosen as the first online application to be converted from VSAM to Oracle.

2.0 Project Description

2.1 Purpose

The primary purpose of this project is to develop a conversion model to be used by applications programmers in the SI group who are converting applications from VSAM to Oracle. To aid in the development of the conversion model, another part of this project will be to convert the data storage method currently being used by the graduate online application from VSAM to an Oracle database. The goal is to convert the data storage method from a non-relational storage method to a relational database. There are several reasons why the decision was made to convert the graduate online application from VSAM to Oracle:

- The University is moving towards Oracle as the database standard.
- As a result, the Student database (commonly referred to as SIS) will be converted to Oracle in the near future. This is important because all applicant data gathered by the graduate online application eventually gets stored in the SIS database.
- Therefore, the goal of the SI group is to stop developing applications and systems using VSAM files and start developing applications in Oracle.

2.2 Scope

This project will focus on converting the VSAM file to Oracle tables and modifying the COBOL/CICS program to write to those tables. This project will not include any enhancements to the online application interface (form). Also no modifications will be done to any of the other processes or applications that currently use the VSAM file and as a result the actual deployment to production will not be included as part of this project.

2.3 Goals and Objectives

The primary goal of this project is to:

- Develop a conversion model for converting VSAM files (non-relational) to Oracle (relational) tables.

Other secondary goals of this project are to:

- Create tables in Oracle based on the current VSAM record layout of the graduate online application.
- Modify the COBOL/CICS program that currently writes to the VSAM file to write to the Oracle database.

Currently there are eight online admissions applications that use VSAM files as the data storage method. These applications will also need to be converted to Oracle. By converting the graduate online application to Oracle, a lot of insight can be gained about the conversion process and a conversion model can be developed that will help with future conversions.

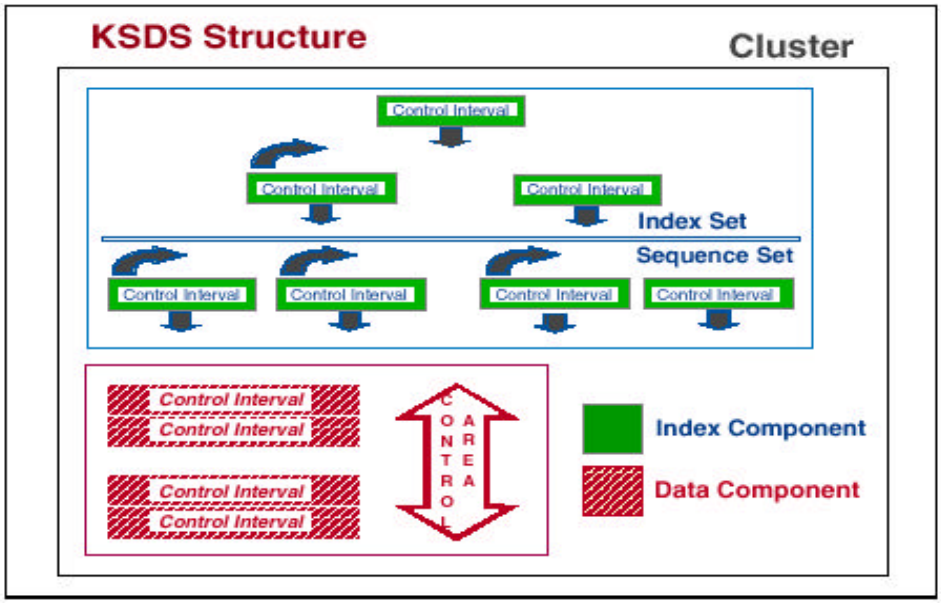
3.0 VSAM vs. Relational Database Models

3.1 What is VSAM?

VSAM (Virtual Storage Access Method) was developed by IBM in the 1970's to replace an existing much older storage technology called ISAM (IBM, 2001). VSAM is a file storage method for IBM's mainframe operating system called MVS (OS/390). "VSAM is one of several access methods that define technology by which data is stored and retrieved" (IBM, 2001, p. 1). VSAM supports several different data set organizations: entry-sequenced, fixed length relative record, variable length relative record, and key sequenced (IBM, 2001).

The most popular type of data set organization for VSAM is key sequenced or KSDS. This is the data set type that is being converted in this paper. In KSDS, records are inserted in ascending order by the primary key (IBM, 2001). The key is unique for each record and must be located in the same position of each record. Once the record is inserted the key cannot be changed, but the record can be updated or deleted. A KSDS data set can be accessed using three different methods: sequential, direct, or skip-sequential. Sequential access allows records to be retrieved in ascending or descending order by key. The key does not have to be specified because VSAM automatically obtains the next record. In direct access, the user must provide the key in order to retrieve records from the data set. The entire key does not have to be supplied to retrieve records. The beginning of the key (high order portion) can be supplied instead of the entire key to retrieve all of the records beginning with that high order portion. Direct

access saves a lot of processing overhead because only a small percentage of the records have to be retrieved. Skip-sequential allows the user to retrieve selected records in ascending order. The diagram below shows the KSDS structure (IBM, 1997).



3.2 Similarities and Differences

When converting VSAM files to the relational model, the user should know the basic similarities and differences between them. Knowing these similarities and differences will make the conversion process go more smoothly. Even though there are some differences between VSAM and the relational model, the relational model is capable of performing the same functions as VSAM (IBM, 1997).

One of the major differences between VSAM and relational is the way records are stored and retrieved. VSAM stores related data in the same record and allows this data to be retrieved at the same time as one single unit. There is generally little to no normalization of data in a VSAM file. There are also data redundancy problems associated with VSAM files. Data redundancy occurs when multiple VSAM files contain the same data fields. For example, the applicant's address may be stored in more than

one VSAM file. This is redundant and wastes storage space. On the other hand, the relational model uses normalization to break up related data into separate tables requiring the tables to be joined for retrieval. By using normalization and breaking up the data into separate tables, data redundancy is greatly reduced. Other important differences include: relationships, referential integrity, ordering, file and table format, one record at a time, and physical constraints.

In VSAM, relationships are not directly specified but rather are implemented through the application programs that access them (IBM, 1997). However, in the relational model relationships are directly specified based on the primary and foreign keys. In the relational model, referential integrity is also enforced by the values of the primary and foreign keys. As with relationships, VSAM enforces referential integrity through the application programs. VSAM orders the records in the file based on the primary key or alternate index and the records are guaranteed to always be in key order. In relational, there is no guarantee of order and the query must explicitly use the ORDER BY clause for ordering of the records. The format of a relational table is restricted in order to ensure searching of a table based on any column. The search column must be identified without ambiguity. VSAM record formats are not as restricted as the relational table and allow repeating fields and groups of data. Records are retrieved from a VSAM data set one record at a time. In relational, records are retrieved from a table as a set as opposed to one record at a time. However, a cursor can be used to retrieve records from a relational table that will provide the same functionality as one record at a time. Lastly there are some physical constraint differences between VSAM and relational. The record size of a VSAM record can be very large because of the ability to span the control

intervals, units of information that VSAM transfers between the storage device and the processor (IBM, 2001). This could be problematic with converting to relational because there is a size limit on the table size. The solution is to split the VSAM record into multiple relational tables.

3.3 Advantages of Relational

As with any conversion, the conversion from VSAM to relational can be a costly and complex process. However, there are many advantages of the relational model that greatly outweigh the cost and help to justify the conversion. Overall, the relational model allows the user's data and applications to be easily adaptable to an organization's changing business needs (IBM, 1997). The relational model provides many advantages over VSAM such as, data name standardization, enhanced data recovery and security, improved data sharing, improved data integrity, and portable applications (England, 1996).

Naming conventions of data fields in VSAM files have been found to not follow any standards in the past (England, 1996). Many times across multiple VSAM files the same data item may have several different names. Since the relational model reduces data redundancy, the same data item is not stored in multiple places, which resolves the problem of inconsistencies in data names. VSAM files do have a backup system in place, but the relational model has a more robust and sophisticated data recovery and security management (England, 1996). Relational tables also have security in place at the field-level, data content level, and encryption level. One of the major advantages of relational is improved data sharing. One problem with VSAM files is the limited ability to share the data between batch and online applications (England, 1996). When batch jobs run,

the VSAM file has to be closed and therefore prevents online applications from reading or updating the file. The relational model eliminates the batch and online sharing problem because tables are locked at the record level (England, 1996). Data integrity is also improved by converting VSAM to the relational database model. By converting to the relational model, the applications are now portable since the applications will be using SQL to access the tables. Portability is important especially if the SIS group decides to stop using COBOL and use newer technologies, such as JAVA.

Converting VSAM to relational also gives flexibility, portability, and the capability for organizations to build:

- Data warehouses,
- Executive information systems,
- Data marts,
- Client/server applications,
- Network computing applications, and
- Mobile computing applications (IBM, 1997).

The relational model also helps reduce maintenance costs, reduce operational effort, and provides high productivity for end users (IBM, 1997).

There are many advantages to relational over VSAM, but there are also many reasons why a business organization may choose not to convert to relational. One problem with converting from VSAM to relational is that the conversion requires a huge investment before any benefits are felt by the organization (England, 1996). This can deter many business organizations from attempting a conversion (England, 1996). Many times it is hard for a business to justify the conversion if the new system does not provide any new functionality. Additional reasons an organization may not convert from VSAM to relational include cost of the conversion, lack of skilled workers, uncertainty about costs and results, and resistance from personnel and executives (IBM, 1997).

4.0 Literature Review

4.1 Reengineering Non-relational to Relational

For purposes of this paper, reengineering VSAM to relational includes the conversion of the actual file, conversion of the programs, and conversion of the processes that utilize the VSAM file. In 1996, Rob England described a methodology for reengineering non-relational databases to relational databases. In particular, this methodology focused on reengineering VSAM, IMS, and DL/1 applications to a relational database. For purposes of this paper, methods related to reengineering VSAM applications to relational will be examined more closely.

England's (1996) approach was a combination of step-by-step instructions and the utilization of a software construct called "transparency". England recommended having one relational database that provided both relational and navigational access methods. By doing this, existing applications would not have to be modified right away to read the relational database and developers could rewrite these applications at their leisure. At the same time, developers could begin developing new applications to read the relational database. England (1996) discussed achieving this by using a software construct called "transparency". "A transparency is a layer of software that translates a data manipulation language (DML) call to a legacy database into a call that is understood by the relational database, and converts the returned data and codes back to the format of the legacy DML" (England, 1996, p. 58). This software allowed the existing applications to remain unchanged while reading the new relational database by using the navigational access

method. A transparency is usually only used as an interim measure until the existing applications can be reengineered. Unfortunately, in order to utilize this method business organizations must purchase the transparency software and a database that supports both navigational and relational access. This can be a costly approach for any business organization, but may be worth the investment when a very large application that will take years to convert is being reengineered. This approach is not feasible for the conversion being discussed in this paper. The graduate online application along with our other VSAM applications are not large enough applications to justify the cost of the transparency software and a relational database that supports both navigational and relational access.

Another part of the method England described is a step-by-step approach to reengineering applications. England's approach consisted of four major steps: "1) reverse engineer a high-level data model; and 2) add supporting data integrity rules to the data model; and 3) remove data oriented logic from programs; and 4) begin process reengineering" (1996, p. 61). Within each of these four major steps, England listed instructions to help in reengineering applications. Even though these steps include instructions on how to utilize the transparency software, they can still be used towards the reengineering of an application that does not use transparency. The steps that England discussed are very general and can be applied to any conversion. Therefore some of the instructions of the approach will be useful to developing the conversion model being discussed in this paper. For example, one of the instructions stated the developer should develop an entity relationship (ER) diagram for the legacy system. Creating an ER diagram will be done when converting an application from non-relational to relational

regardless if the transparency software is used. There were definitely some useful steps and instructions in England's method that can be utilized in the development of the conversion model discussed in this paper.

4.2 VSAM Conversion for COBOL

Huang and Tsai (1999) described a semi-automatic approach to converting a VSAM file to relational tables along with the corresponding COBOL application code. In this paper, Huang and Tsai addressed database schema conversion and SQL query generation. Huang and Tsai's (1999) approach consisted of four steps: "1) identify VSAM data set schema and convert them to SQL relations; and 2) identify input-output (I/O) operations that manipulate the VSAM data sets; and 3) generate functionality equivalent SQL statements; and 4) restructure the code" (Huang & Tsai, 1999, p. 313). This approach was very specific and proved to be very useful in developing the conversion model described in this paper.

Step one of the approach discussed how to convert the VSAM data set schema to SQL relations or tables (Huang & Tsai, 1999). In developing this step of the approach, Huang and Tsai assumed the VSAM file being converted was properly designed to avoid data redundancy. This may be an unreasonable assumption to make because data redundancy is one of the major problems with VSAM that is solved by converting the file to relational. In making this assumption, Huang and Tsai created one step that stated, "Create one relation for one record type and convert each field in the record format to an attribute of the relation" (Huang &Tsai, 1999, p. 316). By doing this, Huang and Tsai only guarantee the resulting relational will be in first normal form (1NF). They defend this approach by stating that if the relations have at least the same degree of

normalization as the original VSAM file, the logical view of the database remains the same and the flow of the COBOL program will only need minor changes. This step of the approach appeared to defeat the purpose of converting the VSAM file to relational in the first place. For the development of the conversion model in this paper, this step of the approach will not be useful. If the programmers do not normalize the VSAM file and leave it as one record when they convert it to relational, then nothing is gained by doing the conversion. The programmers also do not solve any of the design problems that are associated with VSAM by leaving it in first normal form. Therefore, the SIS staff cannot utilize this step of the approach because it would be in our best interests to at least normalize the VSAM file to third normal form (3NF). However, Huang and Tsai probably felt it would be difficult to describe how to convert a VSAM file in first normal form to relational tables in third normal form. There has to be an assumption that the programmers have some background in relational theory when developing any conversion model.

Step two of this approach stated that each I/O statement in the COBOL program should be identified (Huang & Tsai, 1999). Huang and Tsai list nine types of I/O statements the programmer should be looking for in the program. Examples of I/O statements are OPEN, CLOSE, READ, and WRITE. According to Huang and Tsai, each I/O statement should be easy to identify automatically because the statement must specify the data set it operates on (1999).

Step three of the approach described in detail how to convert the I/O statements identified in step two to SQL statements (Huang & Tsai, 1999). Each I/O statement was broken down and explained according to the access mode of the VSAM file. VSAM files

have three different access modes: random, sequential, and dynamic. SQL cursor declarations were also discussed as part of the I/O statement conversions. Huang and Tsai discussed in great detail how to handle processing loops efficiently to help avoid performance degradation.

The last step of the approach explained how to restructure the code by adding the SQL statements to the COBOL program and removing the VSAM I/O statements (Huang & Tsai, 1999). Huang and Tsai (1999) recommended not deleting the VSAM I/O statements until all of the SQL statements had been added to ensure that the original control sequence of the COBOL program is not destroyed.

Overall, this approach described by Huang and Tsai will be very useful in the development of the conversion model described in this paper. The steps of the approach were very detailed and specific to VSAM and COBOL. The first step did have some weaknesses, but on the other hand it did help emphasize the importance of normalization.

4.3 Transition of a Very Large Database

In 1997, Finken described a plan for the transition of the Identification Automated Services (IDAS) system to the new Interstate Identification Index/Federal Bureau of Investigation (III/FBI) segment database. The plan consisted of two steps: "1) transferring the data from one hardware platform to another; and 2) checking the integrity of the moved data" (Finken, 1997, p. 33).

The first step of the plan included several different sub-steps, but the steps of interest for this paper are conversion of the data to fit into the new database structure and loading it into the new database (Finken, 1997). The other steps in the plan were specific to IDAS and did not prove useful to the conversion model in this paper. In converting the

data, the IDAS VSAM files were converted to IDAS flat files to transition the data to the new III/FBI database. Customized software was developed to read the IDAS flat files and convert the data to the format of the III/FBI database by creating III/FBI flat files. After the III/FBI flat files were created, the data in these files were loaded into the III/FBI database using two ORACLE tools, PRO*C and SQL*Loader. "PRO*C is an extension of the library of C language used to interface with ORACLE and SQL*LOADER is a utility for bulk-loading data into ORACLE tables" (Finken, 1997, p. 36).

The second step of the plan consisted of checking the integrity of the data in the new III/FBI database (Finken, 1997). The integrity of the data was checked and verified according to six rules: mapping rules, column integrity rules, translation rules, substitution rules, context rules, and special rules (Finken, 1997). Mapping rules consisted of one-to-one mapping of IDAS data elements to III/FBI data elements, mapping a single IDAS data element to multiple III/FBI data elements, and mapping several IDAS data elements to a single III/FBI data element (Finken, 1997). The following items were checked for the column integrity rules: data type, data length, nulls, range, and format of the data (Finken, 1997). Column integrity rules checked for problems such as, wrong data types, correct length of data, and a valid range of values for a field. Translation rules involved translating IDAS data value codes to different codes in III/FBI. For example, In IDAS '1' may represent blue eyes while in III/FBI 'BLU' represents blue eyes (Finken, 1997). To do the translating, translation software is implemented to convert IDAS data values to III/FBI data values (Finken, 1997). "Substitution rules provide an alternate, legal (approved) value to substitute (by the translation software) for an original IDAS value" (Finken, 1997, p. 39). Substitution

rules were used when the data element values failed other data translation rules. Context rules checked the relationships among the data and were broken up in two different types: single entity context checking and referential integrity rules (Finken, 1997). "Single Entity Context Checking verifies that the data contained in an element conforms to values in another element in the same logical record. For example, zip code must match state and city" (Finken, 1997, p. 39). Referential and integrity rules consisted of relationships between two or more data elements. Special rules were set aside for any other rules that do not fit into one of the other five data integrity rules. The data was checked against these six rules to ensure the highest level of data integrity.

The objective of this plan was to document the steps taken during the transition of a very large database. For the purposes of this paper, there were only two steps of this plan that would be helpful in the development of the conversion model. Those two steps talked in great detail about the transition of the data to a new database and checking the integrity of the data once in the database. Both of these two steps need to be included in the conversion of the VSAM file and in the conversion model described in this paper.

4.4 VSAM to Relational

A programmer can review all the literature that is out there, but no one knows the concepts of VSAM better than IBM. This is why IBM (1997) has an entire book on how to convert VSAM to DB2. The conversion described in this paper converts a VSAM file to Oracle, but the overall concepts of converting VSAM to relational described in the book can be applied to this conversion. The book described converting VSAM to DB2 using DB2 transparency that will not be needed in the conversion described in this paper

(IBM, 1997). Therefore, only a few chapters of the book will be reviewed for purposes of this paper: 1) planning for conversion, 2) database design, and 3) testing.

The planning for conversion chapter gave an overview of how to plan for a successful conversion by including concepts such as conversion phases, conversion methods, inventory of files and applications, and conversion tools (IBM, 1997). The book explained that a conversion was more successful if done in phases. Several different conversion methods were also compared: translation, transparency, re-engineering, reverse engineering, redevelopment, and the corporate model. Translation takes the VSAM calls and translates them to DB2 calls. Transparency intercepts VSAM calls and re-routes them to DB2 calls. Re-engineering consists of changing enough to remove dependencies on the old structures (VSAM) to become compatible with good DB2 design without rethinking the whole data design. Reverse engineering consists of capturing the old designs into models and generating new programs and designs. Redevelopment is usually needed if old code is in poor shape and therefore models are redesigned from scratch. The corporate model involves developing a completely new structure of the data and applications (IBM, 1997). Issues such as time, costs, and quality of results were compared among all of the conversion methods. The application developer or manager must decide on which conversion method best fulfills their needs. Guidelines were discussed on how to conduct inventory on files and applications that would be affected by the conversion. Several programs that would help in automating the conversion process were discussed, including database design, transparency, and data movement programs. However, no specific programs were discussed in the book.

The second chapter of interest from the book was the chapter on database design. Steps were discussed on how to convert VSAM data to DB2 data (IBM, 1997). Even though the steps were specific to DB2, some of the concepts can also be applied to the conversion discussed in this paper. The first step stated the user should verify that each VSAM record type is in third normal form. This step would prove useful unlike the step described by Huang and Tsai (1999) that did not include normalizing the data. Once each record type is in third normal form, a table can be created for each record type. Each field within the record type would become an attribute of that table. Then a primary key would be established for each table created. Many times VSAM files are not in third normal form and need more steps to be converted to relational tables. This chapter went into great detail to describe the differences between VSAM and relational and how to convert VSAM to relational tables.

Another very important issue in a conversion is testing. The third chapter of interest in the book was the chapter on testing. This chapter discussed in great detail a testing methodology to be followed after a conversion (IBM, 1997). The chapter recommended that the following issues be planned for and agreed to by the developer and the senior analyst before the actual conversion takes place: measurable tests, acceptance criteria, test exit criteria, schedule dependencies, and task relationships (IBM, 1997).

This book by IBM went into great detail about converting VSAM to DB2. Even though the book concentrates on DB2, the concepts can be used to convert the graduate online application and to develop the conversion model discussed in this paper.

5.0 Methodology

5.1 Developing the Conversion Model

For this project several methods were combined in order to develop the conversion model. The first two methods were ones I chose to conduct while the third one was drawn from the literature. Those methods were: 1) to obtain a "wish list" from the stakeholders of what they would like to see documented in the conversion model; 2) to conduct a case study of other conversions and conversion models to evaluate their usefulness towards developing the conversion model; 3) to implement a conversion of the graduate online application in order to document the steps taken and to create a working example of the model.

The first method was to obtain a "wish list" from the stakeholders of what they would like to see documented in the conversion model. This was done in order to gain insight and information from the stakeholders who will be using the conversion model. Getting feedback from the stakeholders early in the process was important because if the stakeholders did not find the conversion model useful they may not use it. For this project, the stakeholders were applications programmers in the student information (SI) group. Five programmers with different levels of experience with VSAM and Oracle were chosen. The programmers should have developed at least one application using VSAM. This requirement was needed to ensure the programmers understood the VSAM concepts contained in the conversion model. Once the set of five programmers had been established, an email was sent to each of the programmers asking for their participation in

the evaluation of the conversion model. It was explained to the programmers that their participation was needed for two phases. The first phase was to gather a "wish list" from the programmers for the conversion model and in the second phase the programmers were asked to evaluate the conversion model once it was complete. The programmers had to agree to participate in both phases (the second phase of evaluating the conversion model is described in more detail in section 5.2). Once the programmers agreed to participate in both phases, they were sent an email containing the following questions for the first phase:

1. What topics would you find to be useful to include in the conversion model?
2. What topics would you find not useful to include in the conversion model?
3. Can you think of any good and/or bad examples of a conversion model you have used before (work related or non-work related)?
4. For the conversion model you mentioned in question 3, what aspects made the conversion model good and/or bad?
5. What format would you like the conversion model to be created in? (Examples include: Web based, a Word document, a COBOL copy member, a printed version)
6. Would it be helpful if the conversion model were created in a format that you could update?
7. If you answered yes to question 6, would you update the conversion model if you came across any issues that were not documented?

Getting feedback from the programmers helped ensure that a useful conversion model was developed.

The second method conducted was a case study of other conversions and conversion models. The conversions' usefulness in developing the conversion model was evaluated. This was part of reviewing the relevant literature about other conversions implemented that converted non-relational to relational. By reviewing other conversions, insight was gained about the methods, tools, and other means used to implement the

conversions. The methods and tools used in previous conversions did prove useful in the implementation and creation of the conversion model.

The third method was to implement a conversion of the graduate online application to document the steps taken and to create a working example of the model. The conversion consisted of converting a VSAM file to Oracle tables. Steps were documented throughout the conversion process based on the programmers' feedback in the first method described above. The programmers' feedback helped determine what needed to be documented. As future conversions are implemented, programmers can use the working model as an example along with the conversion model. During the conversion process the following areas were documented:

- Setting up the test environment
- Data type conversion recommendations (VSAM → Oracle)
- COBOL/CICS code changes
- Converting existing data
- Implementation/Deployment Issues
- Monitoring the system once in production

The information gathered from these three methods was combined to create the conversion model presented in Appendix D.

5.2 Evaluating the Conversion Model

The second phase involved the participation of the programmers to evaluate the conversion model (the documentation) once it was developed. The conversion model was evaluated by the same set of five programmers who submitted their "wish list" in the first phase. When the conversion model was complete, a second email was sent to the programmers that contained the conversion model to be evaluated and the following questions:

1. Name at least one thing you liked about the conversion model?

2. Name at least one thing you disliked about the conversion model?
3. Can you think of any ways this conversion model could be improved?
4. If you had to do a conversion, do you think you would use this conversion model? Would you find it helpful?
5. Are there any issues this conversion model does not address?
6. Are there any aspects of the conversion model that need to be made clearer?
7. Do you find the conversion model easy to understand?
8. Does this conversion model appear easy to update?

Once the programmers returned the feedback, the feedback was analyzed to see if there were any major themes or trends that stood out. Some of the major themes I looked for were: 1) did the programmers agree or disagree on particular issues; 2) did the programmers tend to focus on the same things that were needed to improve the conversion model; 3) did the majority of the programmers find the conversion model useful.

5.3 Testing the Graduate Online Application Conversion

Once the conversion of the graduate online application was completed, there were several issues that needed to be tested before the new system moved into production. The main goal for testing was to make sure that the new system (Oracle) was equivalent to the current system (VSAM). In order to check the equivalency, testing was done to check the integrity and accuracy of the data as well as the performance of the new system.

To check the integrity and accuracy of the data, a sample graduate application was entered in the current system and in the new system. Once the applications were entered, the data was compared between the two systems. Comparing the data allowed close examination of how the new system stored the data. The goal was to make sure the new system had not lost any data as a result of the conversion. Another goal was to verify the data type conversion was done properly. If the data was a text field in VSAM, it should

also be a text field in Oracle. This was also true for numeric fields. Checks were made to ensure that all date fields were stored in the correct format in Oracle. If the date was stored as CCYYMMDD in VSAM, then the date was stored in that same format in Oracle. Any decimal fields were also checked to verify the decimal was stored in the correct place. Verification of the integrity and accuracy of the data was essential in the success of the conversion and other processes that depend on the data stored in the new system.

The equivalency testing described above was done to ensure the new system handled the data correctly when good data was received. Testing was also done to ensure the new system handled data correctly when bad data was received. Bad data was handled by making sure there were appropriate validation routines in place to capture the bad data before it was written to the database. The validation routines checked incoming data, validated the data, and sent an error message to the applicant if the data was not valid. The first type of validation performed was checks for required fields. There are certain fields that were required in order for an applicant's application to be considered complete. Without all of the necessary information, a decision cannot be made on whether to admit the applicant to the University. Examples of required fields are the applicant's name, address, term applying for, program applying for, and citizenship status. There were also checks for wrong data types, for example if a certain field is numeric but the user enters text instead. Checks were also made to make sure dates were in the correct format needed. For example, the date the applicant took the GRE should be in the format of MM/CCYY. Other checks needed to be made for leading blanks and punctuation, trailing blanks and punctuation, too much data in the field, and out of range

data. An example of out of range data was when the attended college from-date was greater than the attended college to-date.

Another part of equivalency testing was performance testing. The main component to test was the response time of the new system during non-peak and peak hours. For our purposes, non-peak hours were during the hours between midnight and seven o'clock a.m. and peak hours are during the hours of eight o'clock a.m. and five o'clock p.m. In order to test the performance, a benchmark was established based on the current system's response time. The benchmark was determined by entering an application in the current system during non-peak and peak hours and recording the time it takes to receive the confirmation page. For consistency purposes, a sample graduate application was entered in the new system and the response time was recorded. The response times of the current and new system were compared. The goal was for the response time of the new system to be the same if not better than the current system. Another part of performance testing was stress testing. Stress testing consisted of several people entering an application in the new system at the same time. The purpose of stress testing was to make sure the new system could handle the load during high traffic times. High traffic times occur approximately one month before and up to the application deadline.

The current system already had some validation routines in place. These routines did not have to be changed and can be used for the new system. The application also used drop down boxes for some fields, which helped in minimizing the amount of bad data being passed from the application. By using drop down boxes, the ability was given to control some of the data that was passed to the COBOL/CICS program. Since the

current system already had validation routines in place to handle bad data, the emphasis of the testing was on checking if the new system handled good data properly and on the performance of the new system.

Most of the testing was performed when the new system was in the test environment. However, a final test should be performed once the new system moves to production. This final test is needed because many times there is a small difference between test and production environments that may result in errors that were not encountered in the test environment. The purpose of the final test is to ensure that moving the new system into production is successful.

6.0 Current System

6.1 Online System Overview

The graduate online application, according to the graduate school, is the preferred means of receiving an application for admission. While the admissions process is still not completely paperless, the online application is a step in that direction. The graduate online application can be found at https://www-s2.ais.unc.edu/sis/adm/gradapp.html. Here the applicant can find the application and the directions for applying. Both United States citizens and international applicants can use the online application.

The process starts out when an applicant fills out the application and clicks the submit button (see Appendix A - Current Online System Overview Flow Chart). Filling out the application occurs in a single session and there is currently no way to fill out the application in parts. Data from the application is then passed to a Perl script along with the name of the COBOL/CICS program that processes the data. The Perl script calls the COBOL/CICS program and passes the application data to the program to be parsed. The COBOL/CICS program writes the data to the VSAM file and sends a confirmation page back to the applicant. The confirmation page contains an option to pay by credit card. If the applicant decides not to pay by credit card the application is complete and the applicant is instructed to print the confirmation page and mail it to the graduate school. If the applicant chooses to pay by credit card, then a CGI script (Perl) is called to process the credit card transaction. Once the transaction has been processed, another COBOL/CICS program is called to update the VSAM file with the new fee type

(indicating fee paid by credit card) and to add an orderID to the applicant's record (which enables reconciliation with reports from the vendor who handles the credit card transactions). After the credit card transaction is complete, the COBOL/CICS program also sends a confirmation page to the applicant to be printed and mailed to the graduate school.

6.2 Online System Detailed Description

This section describes how the COBOL/CICS program processes the application data in more detail (see Appendix A – Current Online System Detailed Flow Chart). Once the applicant clicks the submit button, data is passed to the COBOL/CICS program and is verified for completeness. If any required field is left blank, an error message is sent back to the applicant indicating the field that needs to be completed. If all required fields are complete, then the COBOL/CICS program validates certain data such as the birth date. If all data is valid, then the VSAM record is built. If data is not valid, an error message is sent to the applicant indicating the field that needs attention. Once the program has verified and validated all of the data, the program proceeds to write the data to the VSAM file. Before the data is written to the VSAM file, the program checks to make sure the application is not already in the file. If the application already exists, a duplicate application error is sent back to the applicant. If the application does not exist, then the application data is written to the VSAM file. A confirmation page is then sent back to the applicant.

6.3 Batch System Description

Even though the graduate application is an online process, there is some batch processing that takes place on a nightly basis (see Appendix A – Current Batch System

Overview Flow Chart). There are several programs that run during batch processing. As mentioned earlier, the graduate admissions process is not totally paperless. One batch process prints out a "profile" sheet containing the application information on each new applicant for that day. These "profile" sheets get sent to the graduate school to be considered for admission. The "profile" sheets become the official application and will be filed in the applicants' folder. Another major batch process converts the social security numbers (SSN) stored in the VSAM file to personal id numbers (PID). For legal and privacy reasons the University cannot store data in the student database using the SSN, therefore a PID has to be assigned for each new applicant. This PID becomes the unique identification for the applicant/student at the university. Once all of the SSNs are converted to PIDs, another batch process runs to post each application to the student information database (SIS). There are also other processes that read the VSAM file such as a web-enabled recommendation form for admittance and reconciliation of credit card transactions.

7.0 Database Design

As discussed earlier in this paper, there are several problems with VSAM files that can be resolved by converting the file to the relational model. One of the problems with VSAM is the lack of normalized data. For example, in the graduate online application VSAM file, all of the data are stored in one record. The applicant's address, biographical data, college data, next of kin data, etc. are stored in the same record. The college data will serve as an example. On the online application the applicant can enter up to three different colleges, but if the applicant has only attended one college the fields for the other two colleges are blank. In relational, this problem is solved because only one college record would be in the college table.

After evaluating the current record layout of the online graduate application VSAM file, it was decided that twelve tables were needed in the new relational model in order to achieve the desired normalization level of third normal form (see Appendix B and Appendix C for the Entity Relationship Diagram and Database schema for this database). The SIS_GRAD_APPLICANT table was created to hold the applicant's biographical data. Biographical data includes fields such as the applicant's name, social security number, birth date, and ethnic origin. The SIS_GRAD_ADDRESS table was created to hold the applicant's address information. This table can hold both the permanent and local address for each applicant. To accommodate the different addresses, an ADDRESS_TYPE column is added to the table that will help distinguish between permanent and local addresses. The SIS_GRAD_PROGRAM table is created to hold the

actual application (program) data. This includes information such as the degree and major the applicant is applying for and the date the application was entered. The SIS_GRAD_CONTACT table was created to hold the information about the applicant's emergency contact (sometimes referred to as next of kin). This table will contain the contact's name, address, and the relationship to the applicant. The SIS_GRAD_CITIZENSHIP table was created to hold the citizenship information about the applicant such as the applicant's citizenship status. The SIS_GRAD_RECOMM table was created to hold the applicant's recommendation information. In order to apply to Graduate School, an applicant must provide recommendations from various people who can evaluate their academic and professional abilities. This table will contain the names and addresses of those people who provided recommendations for the applicant. The SIS_GRAD_HONORS table will contain information about any honors or awards the applicant wishes to be considered when applying to the school. The SIS_GRAD_LANGUAGE table will contain information regarding any languages the applicant knows and their degree of fluency in each language. The SIS_GRAD_SCORES table will contain information on self-reported Graduate Record Examination (GRE) test scores. This are scores the applicant reports on the application, but the Graduate School must receive official GRE scores from the Educational Testing Service (ETS) also. The GRE is a requirement of the Graduate School for admittance to the school. The SIS_GRAD_FINANCIAL table will contain any information regarding the financial needs of the applicant. For example, this table will contain whether or not the applicant is interested in being a teaching assistant. The SIS_GRAD_COLLEGE table will contain information about the previous colleges the applicant has attended.

There will at least be one college for each applicant in this table because each applicant must have a four-year degree in order to attend Graduate School. The SIS_GRAD_SKILLS table will contain any additional information about specific skills that the applicant may have and wishes to be considered in the admission decision.

8.0 Results

8.1 Evaluation of Stakeholders' Wish List

The first method I conducted to contribute in the development of the conversion model was obtaining the stakeholders "wish list". The purpose of obtaining the "wish list" was to find out what the stakeholders wanted documented in the conversion model.

After evaluating the results of the feedback, I determined that the majority of the programmers agreed on the topics that would be useful to include in the conversion model. There were three major topics I found the programmers agreed on including: COBOL code changes, how to test the changes, and how to create Oracle tables. Other topics the programmers mentioned including were: purpose and target audience of the conversion model, an introduction to Oracle, and screen shots of any software needed. The only thing the programmers mentioned they would not find useful to include was anything not pertaining to converting VSAM to Oracle. When creating the conversion model, I tried very hard to include all of the topics the programmers requested. I did however feel that the scope of the conversion model was not big enough to include an introduction to Oracle. I made the assumption that the programmers will get this introduction to Oracle by different means, such as attending a training class or by completing a online tutorial.

In the feedback, the programmers also gave examples of good and bad documentation along with the reasons why those examples were such. The programmers mentioned examples of documentation they have used at work and also some

documentation used outside of work. The programmers found the examples of documentation good for the following reasons: examples of code changes, table of contents, online and downloadable, and easily searchable. The bad examples of documentation were not very good because they were not easily searchable and were hard to follow. When creating the conversion model, I tried to incorporate the good features given of the example documentation and to make sure I addressed the bad features.

One of the important decisions in developing the conversion model was what format to create it in. As part of the feedback, I asked the programmers what format they preferred the conversion model to be created in. The majority of the programmers agreed the format of the conversion model should be either a Word document or a web-based document. Considering the time constraints of the master's paper and the needs of the programmers, I decided to create the conversion model as a Word document rather than web-based. If the conversion model proves to be useful to the programmers, then I may consider in the future making it web-based. All of the programmers said they would update the conversion model if they came across any issues they felt would be helpful to others. Therefore, I felt a Word document would be easier to update than a web-based document and I believed it was important for the conversion model to be updated by the programmers.

8.2 Evaluation of Conversion Model

In the methodology section of this paper, I mentioned that when analyzing the feedback from the programmers I would be looking for three major themes. I did indeed find all three of these themes within the feedback. The themes I looked for were: 1)

whether the programmers agree or disagree on particular issues; 2) whether the programmers tend to focus on the same things that are needed to improve the conversion model; 3) whether the majority of the programmers say they would find the conversion model useful.

The first theme I looked for was whether the programmers agreed or disagreed on particular issues. I definitely found that the programmers agreed on the same issues, and I did not find any issues that they disagreed on. The programmers agreed on the same likes and dislikes they had about the conversion model. Some of the items the programmers mentioned they liked about the conversion model were: 1) the screen prints of how to use PL/SQL Developer; 2) the side-by-side code examples of VSAM and SQL; 3) the glossary of terms; 4) the listing of other resources; 5) the table of contents and overall organization of the conversion model that made it easy to read and understand. Some of the following items were mentioned as dislikes of the conversion model: the primary and foreign key section was confusing and readers believed more SQL related terms should be added to the glossary. These dislikes concerning the conversion model will be taken into consideration and possibly added before the final version of the conversion model is made available to all of the SIS programmers.

The second theme I looked for was whether or not the programmers tended to focus on the same things that needed to be improved in the conversion model. As with the likes and dislikes, the programmers focused on the same issues and areas where the conversion model could be improved. Several programmers mentioned improvements that could be made to the glossary. First, the programmers felt that more SQL terms and even some VSAM terms used in the conversion model should be added to the glossary.

Even though most of the programmers had worked with VSAM before, they felt adding some VSAM terms to the glossary would be helpful for new programmers who may have to do a conversion in the future. Suggestions were also made that the first occurrence in the model of each term contained in the glossary be bold or italics to indicate to the reader the term is in the glossary. Also several programmers wanted to see more contact information in the conversion model. For example, contact information should be given for the appropriate person to contact in the database group to get tables approved before moving them to production. Before the final version of the conversion model is made available, I will investigate all of the important contacts needed and include it in the conversion model.

The third theme I looked for was whether or not the majority of the programmers would find the conversion model useful in future conversions. All of the programmers believed they would use the conversion model if they were ever involved in a VSAM to Oracle conversion. Also all of the programmers said they would update the conversion model if they ever came across any issues that would be helpful to others. Several programmers made the suggestion to put the conversion model on our department's intranet site in order to provide a central location for reading and updating. The same programmers suggested adding a footer to the conversion model indicating that central location.

While analyzing the feedback from the programmers, I came across another theme that I did not initially look for. The theme was that assumptions were made in the conversion model on my part as the writer. I realized after reading the feedback that there were some assumptions made about the level of knowledge of the users of the

conversion model. For example, many of the programmers did not know where the *tnsnames.ora* file was located on his or her computer and therefore did not know where to go to add database instances to this file. In other words, I had assumed that everyone knew where this file was located. I also made an assumption that everyone already had Oracle and PL/SQL Developer installed on his or her computer. Therefore, I did not include where to get the software or how to install it in the conversion model. I did find out, however, that there is already some documentation written on how to install the software and that I should make reference to that documentation in the conversion model I developed. I realized that I needed to make clearer how to create a DBLink, the complete or partial name of a database link to a remote database where the table is located. I also needed to explain how to grant the owner of the DBLink access to the user's tables. Reading the feedback given by the programmers allowed me to see some of the assumptions I made when writing the conversion model that need to be corrected before the conversion model is moved to our department's intranet site.

8.2 Testing the Graduate Online Application Conversion

As mentioned in the methodology section of this paper, the accuracy of the data in the new system (Oracle) and the performance of the new system are the main issues that need to be tested before the new system can be moved to production.

To check the integrity and accuracy of the data in the new system, I entered the same application into the current system (VSAM) and into the new system (Oracle). I then compared the data in the VSAM file against the data in the Oracle tables. After comparing the data, I found no problems with the data in the Oracle tables. All of the data appeared to be accurate and correct compared to the same data in the VSAM file.

Part of the reason there were no problems with the data in the new system was because much of the formatting and validating of the data was already in place in the COBOL program for the current system. Also all of the fields in the VSAM file were alphanumeric, therefore all of the columns in the Oracle tables ended up being the VARCHAR data type. The fact that I did not have to convert many data types when converting the VSAM file kept the number of errors at a minimum.

Another part of testing the new system was performance testing. In order to test the performance, I entered a total of six applications into the VSAM and Oracle systems during peak and non-peak hours. I entered the same three applications with only the required fields entered and another set of the same three applications with all possible fields entered and recorded the response times (see table 1 below).

Table 1 – Peak and Non-peak Response Hours

	Peak Response Hours	**Non-Peak Response Hours**
VSAM		
Required Fields Only		
APP # 1	4 seconds	2 seconds
APP # 2	5 seconds	3 seconds
APP # 3	4 seconds	3 seconds
All Fields		
APP # 4	5 seconds	4 seconds
APP # 5	4 seconds	4 seconds
APP # 6	5 seconds	3 seconds
Oracle		
Required Fields Only		
APP # 1	5 seconds	3 seconds
APP # 2	6 seconds	3 seconds
APP # 3	6 seconds	4 seconds
All Fields		
APP # 4	5 seconds	4 seconds
APP # 5	5 seconds	4 seconds
APP # 6	5 seconds	3 seconds

The response times during peak hours were between four to six seconds. For the VSAM response was four to five seconds and five to six seconds for the Oracle system. The response times during non-peak hours were between two to four seconds. For the VSAM response was two to four seconds and three to four seconds for the Oracle system. I am very pleased that the response time of the Oracle system did not vary much from the VSAM system. This was an acceptable outcome for the response time of the Oracle system and the Oracle system will be considered equivalent to the current VSAM system.

Another part of performance testing was stress testing. This was needed in order to verify that the new Oracle system could handle a heavy load during high traffic times. Those high traffic times occur a month before and up to the deadline for applying for admittance. In order to stress test the new system, five volunteers entered an application and submitted it at the same time. The volunteers recorded any errors they received and the response time for receiving the confirmation page (see table 2 below). I then verified the accuracy of the data for each application in the Oracle tables.

Table 2 – Stress Test Results

	Errors?	**Response Time**	**Data Accurate?**
Person # 1	No	5 seconds	Yes
Person # 2	No	6 seconds	Yes
Person # 3	No	4 seconds	Yes
Person # 4	No	5 seconds	Yes
Person # 5	No	6 seconds	Yes

All of the data in the Oracle tables for each of the five applications entered appeared correct and accurate. There also appeared to be no problems with the response time and no one reported receiving errors after submitting the application.

The overall testing of new Oracle system went very well. All of the results from the data accuracy and performance tests are acceptable and therefore this part of the conversion is considered “production ready”. However, there are many other processes that would need to get converted to read the Oracle tables before the new Oracle system could be moved to production.

9.0 Conclusion

The purpose of this paper was to develop a conversion model that would provide step-by-step instructions for converting a VSAM file to Oracle tables. The conversion model included the entire conversion process from setting up the test environment to moving the new system to production. Three methods were combined in order to develop the conversion model: getting stakeholders' input, evaluating literature for other conversion models, and implementing an actual conversion as a working example of the model. All three methods proved to be very useful and no one method stood out as being better than the other. However, some of the conversion models in the literature I reviewed could be improved. One of the conversion models assumed that the VSAM file being converted would already be normalized and no normalization would be needed during the conversion. I believe this is an unrealistic assumption because one of the major problems with VSAM files is the fact that it is not normalized. Therefore, I believe a step needs to be added that explains how to normalize the VSAM file. Even though the conversion models had this minor weakness, I still strongly believe the combination of all three methods contributed to the successful development of the conversion model.

One future goal will be to modify the conversion model according to the recommendations in the feedback given by the programmers. The programmers' feedback proved very useful and will help improve the conversion model as a whole. Once the modifications have been made to the conversion model, the conversion model

will then be moved to the department's intranet site for reading and updating by the programmers.

The conversion of the online graduate application VSAM file to Oracle tables is just the first phase of the complete conversion process. There are many batch processes that need to be modified to read the Oracle tables instead of the VSAM file (see Appendix A - Current Batch System Overview Flowchart). These processes must be modified before the new Oracle system replaces the VSAM system in production. Another future goal will be to develop a step-by-step conversion model to facilitate the conversion for the batch processes like the conversion model developed in this paper for the online process.

References

Chapple, Mike. (2002). *About Databases*. Retrieved April 10, 2002, from http://databases.about.com/

England, Rob. (1996). Reengineering VSAM, IMS, and DL/1 Applications into Relational Database. *Proceedings of the Sixth International Hong Kong Computer Society Database Workshop on Database Reengineering and Interoperability*, Kowloon, Hong Kong, (pp. 55-68). New York, NY: Plenum Press.

Finken, Shirley J. (1997). Planning the Data Transitions of a VLDB – A Case Study. *Proceedings of the SPIE - The International Society for Optical Engineering*, Boston, MA, (pp.33-45).

Huang, Hai, & Tsai, Wei-Tek. (1999). VSAM Conversion for COBOL Programs. *Journal of Software Maintenance: Research and Practice*, 11(5), 311-337.

IBM Corporation. *Application Development Guide*. Retrieved April 10, 2002 from, http://www3.ibm.com/software/data/db2/udb/ad/v7/adg/db2a0/frame3.htm#db2a0181

IBM Corporation, International Technical Support Organization, Boblingen Center. (1997). *How to Get to DB2 from VSE/VSAM Using DB2 Transparency for VSE/ESA*. (pp. 1-56). Boblingen, Germany.

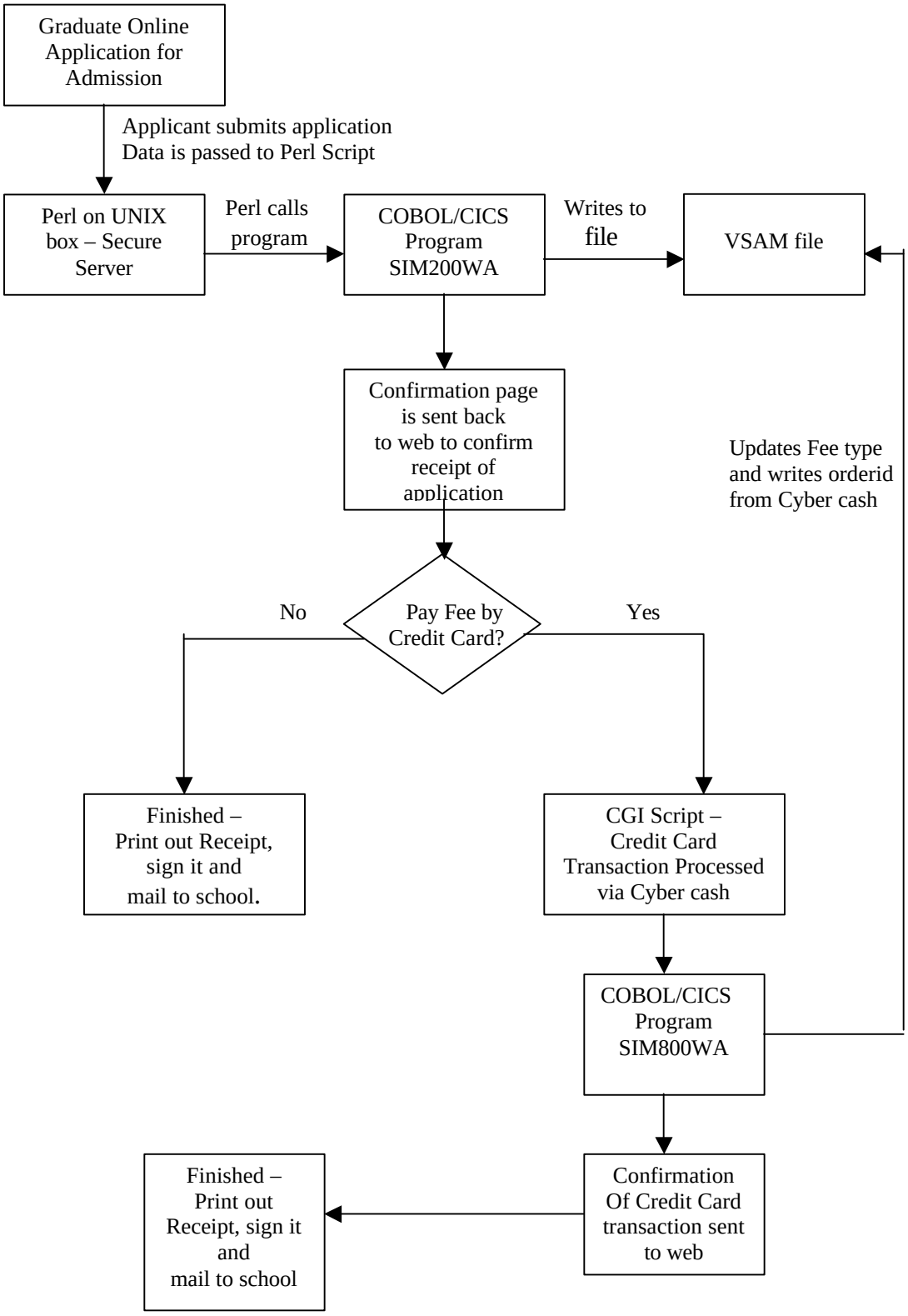
IBM Corporation, International Technical Support Organization. (2001). *VSAM Demystified.* (pp. 1-35). San Jose, California.

Appendices

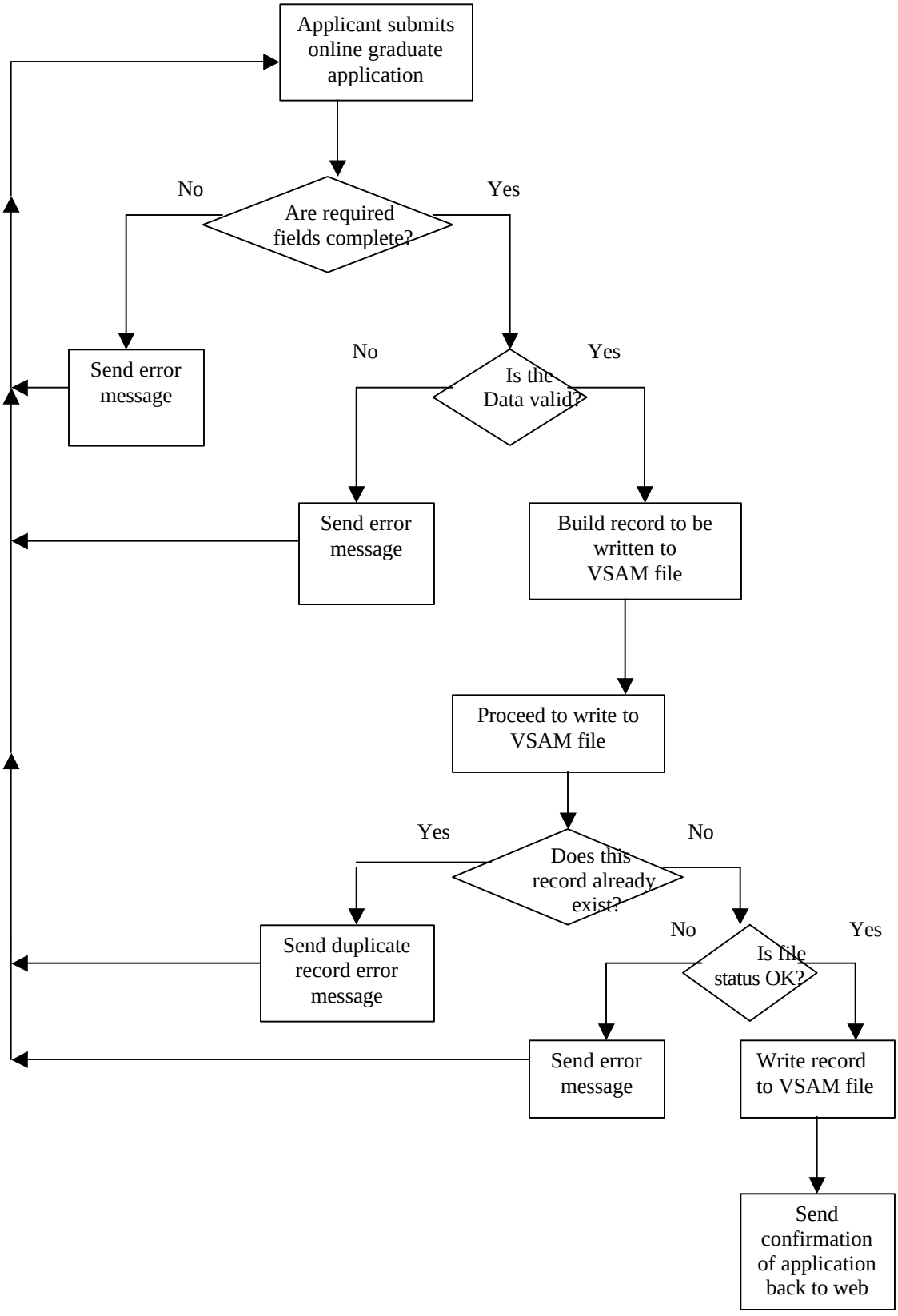
Appendix A – Current Online System Flow Charts
Appendix B – Entity Relationship Diagram
Appendix C – Database Schema
Appendix D – Conversion Model

Appendix A - Current Online System Flow Charts

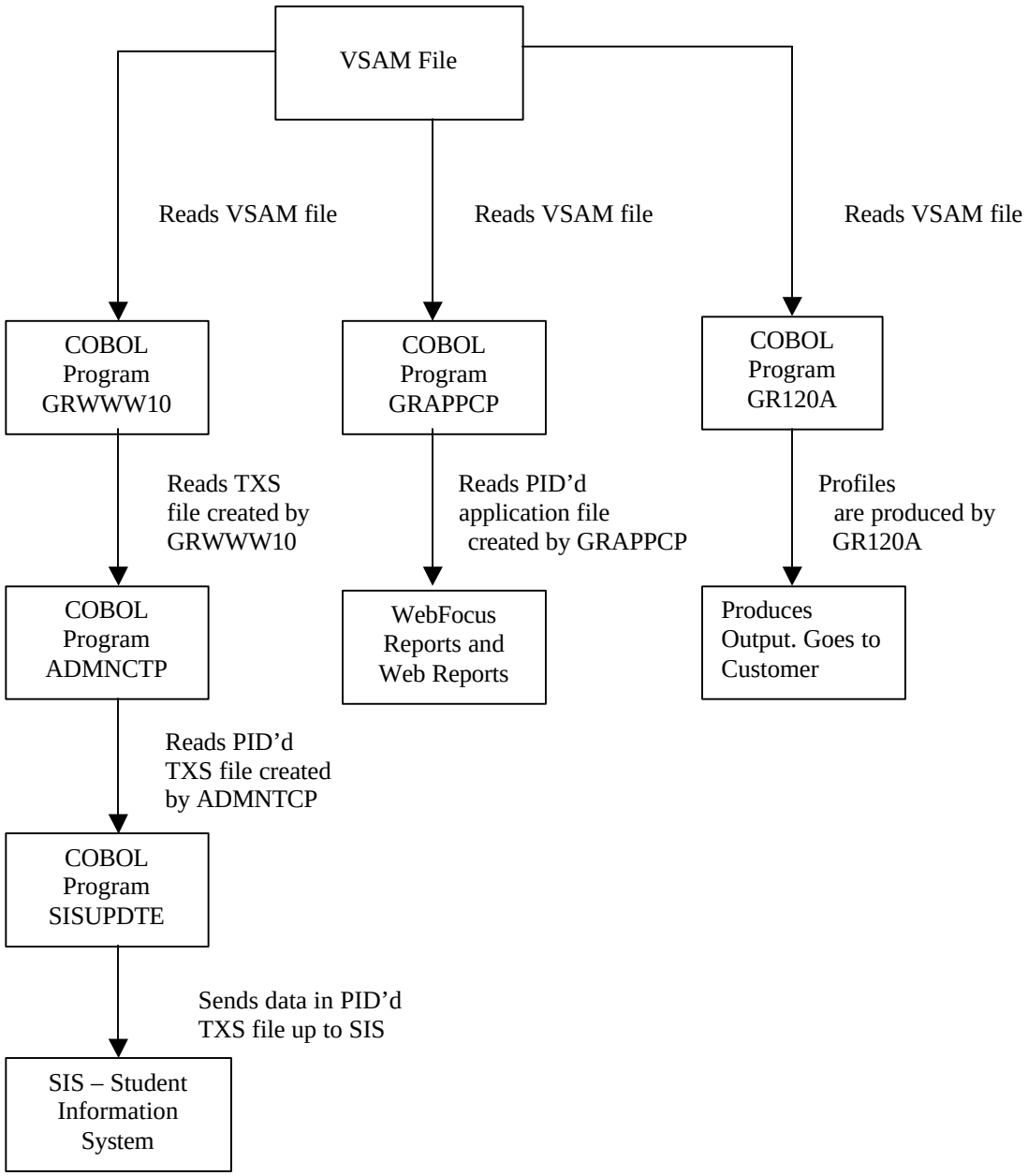
Appendix A - Current Online System Overview Flow Chart



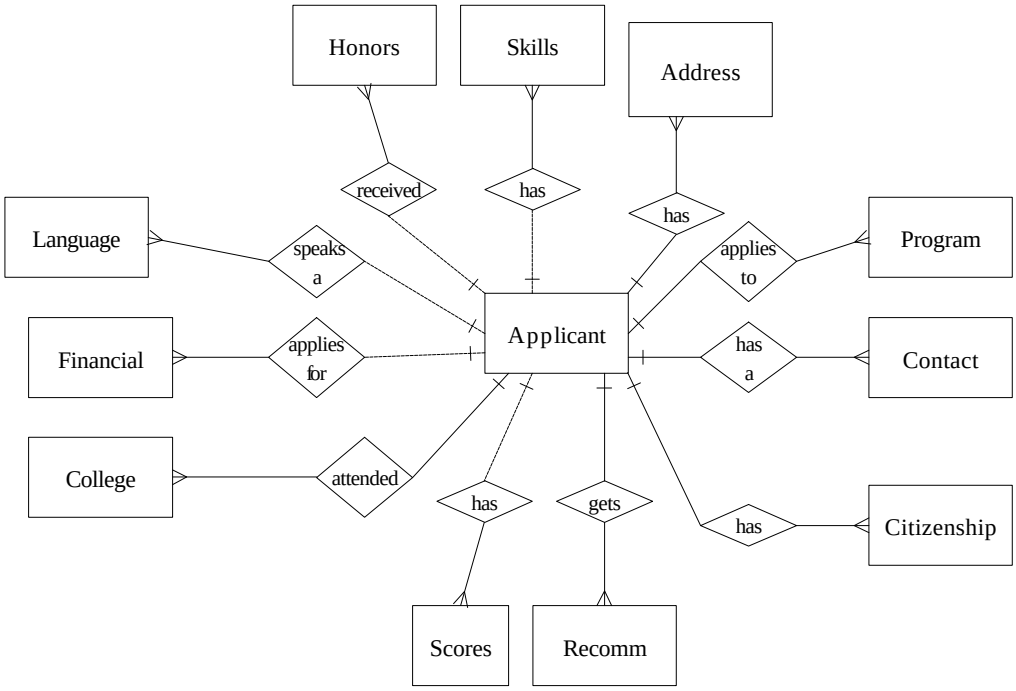
Appendix A - Current Online System Detailed Flow Chart



Appendix A – Current Batch System Overview Flowchart



Appendix B – Entity Relationship Diagram



Appendix C – Database Schema

SIS_GRAD_APPLICANT (APP_SSN, First_Name, Middle_Name, Last_Name, Salutation, Former_Last_Name, Birth_Date, Sex, Ethnic_Origin, Email_Address, Discp_Action, Arrested, Resstat)

SIS_GRAD_ADDRESS (APP_SSN, Address_Type, Addr_Line1, Addr_Line2, Addr_Line3, City, State, Zip, Country, Home_Phone, Work_Phone, Until_Date)

SIS_GRAD_PROGRAM (APP_SSN, APP_Term, APP_College, APP_Major, APP_Level, APP_Degree, APP_Career, APP_Class, Admit_Type, Fee_Type, APP_Resstat, APP_Interest, APP_Date, APP_Time, Maint_Flag, Profile_Flag)

SIS_GRAD_CONTACT (APP_SSN, Contact_Name, Relationship, Addr_Line1, Addr_Line2, Addr_Line3, City, State, Zip, Country, Phone)

SIS_GRAD_CITIZENSHIP (APP_SSN, Status, Origin_County, Origin_State, Origin_Country)

SIS_GRAD_RECOMM (APP_SSN, Recomm_Name, Recomm_Date, Recomm_Addr)

SIS_GRAD_HONORS (APP_SSN, Honor_Name, Honor_Date)

SIS_GRAD_LANGUAGE (APP_SSN, Language_Name, Fluency)

SIS_GRAD_SCORES (APP_SSN, Score_Type, Score_Date, Score, Percentile, Subject)

SIS_GRAD_FINANCIAL (APP_SSN, Fin_Choice, Fin_Date)

SIS_GRAD_COLLEGE (APP_SSN, College_Name, Major, Degree_Type, From_Date, To_Date, Degree_Date, Degree_Specific, GPA, NC_Ccode)

SIS_GRAD_SKILLS (APP_SSN, Skill_Name, Skill_Date)

Appendix D – Conversion Model

Documentation for Converting VSAM to Oracle

Written by Melissa B. Florio – Student Information Services

(NOTE: the examples in this documentation are simplified for illustrative purposes. Also there are references in the documentation to Oracle links that require a login and password.)

Table of Contents

Purpose…… 2
Setting up the test environment…… 2
ORD1……... 2
OD07…… 2
OD09…… 3
DBlink …… 3
Converting VSAM file to Oracle tables……. 4
Creating the relational model……. 4
Naming the fields in Oracle…… 5
Determining the relationship…….. 5
Determining primary keys……. 6
Determining foreign keys…….. 6
Converting data types……. 7
Creating tables in PL/SQL Developer…… 8
COBOL/CICS code changes…… 11
Overview of the structure of SQL statements in COBOL……. 11
Code changes for your COBOL program…….. 12
READ/SELECT…….. 16
WRITE/INSERT……. 16
DELETE/DELETE……. 17
REWRITE/UPDATE……. 17
Testing your system……. 19
Data Integrity……. 19
Performance……... 20
Functionality…….. 20
Moving system to production……. 20
Obtain approval from database group…… 20
Create tables in OP07…… 20
DBlink…… 20
Backup current file and processes……. 21
Migrating VSAM data to Oracle tables……. 21
Glossary of Terms…… 22
Other Resources……. 24

Purpose

The purpose of this documentation is to guide programmers through the process of converting a VSAM file to Oracle. This documentation was written to assist programmers who are converting web (online) applications to access Oracle tables instead of VSAM. Therefore, the information in this documentation pertains only to COBOL/CICS programs and CICS VSAM files.

This documentation is meant to be a helpful tool for programmers. Therefore, if you come across any issues that you feel would be helpful to include in the documentation please do not hesitate to add it. If you don't feel comfortable modifying the documentation, I will be glad to do that for you.

Setting up the test environment

- **ORD1**
  For CICS, the default development Oracle instance is ORD1. The ORD1 instance is on the mainframe.

  Below is the text to add ORD1 to your *tnsnames.ora* file. This file is needed in order for you to connect to ORD1 through PL/SQL Developer or SQL Plus.

```
ORD1 =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = mvs.ais.unc.edu)(PORT =3052))
  )
  (CONNECT_DATA =
   (SID = ORD1)
  )
 )
```

- We have two development Oracle instances on Unix (riogrande).
  - OD07 – this is an updateable instance.
  - OD09 – this is a read-only instance.

- **OD07**
  This is the development Oracle instance where you put the tables you create that need to be updated frequently. Everyone should have a login and password for this instance. If you don't have a login, please see Kay Bobbitt (our FACS coordinator) to request a login. Once you have a login, you can create your tables under your login. This will be your personal schema.

Below is the text to add to your *tnsnames.ora* file for OD07:

```
OD07 =
   (DESCRIPTION =
     (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = riogrande.ais.unc.edu)(PORT =1521))
   )
    (CONNECT_DATA =
    (SID = OD07)
   )
 )
```

We currently have a default login to OD07 called SIRTPD/SIRTPD0822. Some tables will be created under this schema instead of your own schema. This will depend on the project you are working on. Please check with the project leader to determine where you should be creating your tables.

- **OD09**

  This is the development Oracle instance used for read-only. This instance will be used for our WebFOCUS reporting. This instance is mainly used for our nightly batch updates of data.

  Below is the text to add to your *tnsnames.ora* file for OD09:

```
OD09 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = riogrande.ais.unc.edu)(PORT =1521))
    )
    (CONNECT_DATA =
      (SID = OD09)
    )
  )
```

  Again, if you don't have a login for this instance contact our FACS coordinator Kay Bobbitt to get a login. If you are creating tables for our WebFOCUS reporting you should use the group default login SIRTPD/SIRTPD0822.

- **DBLink**

  The ORD1 instance is the default instance for CICS. However, the tables you create will most likely be in OD07. Therefore, you will need to use a DBlink to link over to OD07 from ORD1. A DBlink is the complete or partial name of a database link to a remote database where the table or view is located. If you omit the DBLink, Oracle assumes that the table or view is located on the local database. Please check with your project leader to see if there is an existing DBLink you can use or if one needs to be created. You need to grant the owner of the DBlink access to your tables.

Converting VSAM file to Oracle tables

This section is not meant to teach you everything you need to know about relational theory (especially since there are entire books on the subject). There will be some assumptions made about the level of knowledge regarding relational theory. An example will be used throughout to help explain how to create a relational model from a VSAM file.

- **Creating the relational model**
  In VSAM, you have one record in the file for each unique key. For example, the graduate online application has one record in the file for each application that has been submitted. This one record contains all of the applicant's bio/demo data, address data, application data, test score data, college data, etc. In relational theory, this one record would be broken up into many smaller tables instead of one long record. The tables would be created based on category or subject. For example, one table would be created for the applicant's bio/demo data. Another table would be created for the college data and another table would be created for the test score data, and so on. This grouping based on category would continue until all of the fields in the VSAM file had a corresponding table. Sometimes you may have a field in VSAM that doesn't really belong in a category. When this happens, you just have to use your best judgment on deciding what table you should add this field too. One of the goals of relational theory is to minimize the amount of nulls (blanks) you have in your table. Breaking up each category into a table helps minimize the problem of nulls.

  Let's take a look at an example:

  Here is a section of a VSAM file:

```
01  REC-EXAMPLE.
    03  APP-REC-KEY.
        05  APP-SSN          PIC X(09).
    03  APP-NAME.
        05 APP-LNAME         PIC X(20).
        05 APP-FNAME         PIC X(15).
        05 APP-MNAME         PIC X(08).
    03  APP-SEX              PIC X(01).
    03  APP-DOB              PIC X(08).
    03  APP-ETHNIC           PIC X(01).
    03  APP-TERM             PIC X(05)
    03  APP-COLLEGE          PIC X(02).
    03  APP-MAJOR            PIC X(04).
    03  APP-LEVEL            PIC X(01).
    03  APP-DEGREE           PIC X(05).
```

  This example section of the VSAM file contains bio/demo data and program data for an applicant. Notice there are two categories in this record: bio/demo data and program (application) data.

Here is how you would convert this section to a relational table. Since there are two categories in this record, then we would have two tables.

Table 1 (bio/demo)	Table 2 (program)
APP-SSN	APP-SSN
APP-LNAME	APP-TERM
APP-FNAME	APP-COLLEGE
APP-MNAME	APP-MAJOR
APP-SEX	APP-LEVEL
APP-DOB	APP-DEGREE
APP-ETHNIC	

Notice that for the APP-NAME, I took the field names at the 05 level to include in my table. In this step, just worry about what tables you need and what fields should be in each of those tables. Make sure you account for all of the fields in your VSAM file. See the resources page for other sources regarding Relational Theory.

- **Naming the fields in Oracle**

  Now that you have decided on what tables you need and what fields go in each table, you should come up with some meaningful names for the tables and fields for Oracle. You can probably use the same names that you used in COBOL replacing hyphens (-) with underscores (_). So for example, if you had a field called COLLEGE-NAME in COBOL you need to change it to COLLEGE_NAME for Oracle. For the tables, one standard for our group is to start every table name with SIS.

  Back to our example, here's how to name the tables and fields:

SIS_APPLICANT_BIODEMO	SIS_APPLICANT_PROGRAM
APP_SSN	APP_SSN
APP_LNAME	APP_TERM
APP_FNAME	APP_COLLEGE
APP_MNAME	APP_MAJOR
APP_SEX	APP_LEVEL
APP_DOB	APP_DEGREE
APP_ETHNIC	

  Notice how the COBOL names were used with underscores instead of hyphens.

- **Determining the Relationship**

  A relationship is needed for the following reasons:
  - "It establishes a connection between a pair of tables that are logically related to each other in some form or manner" (Hernandez, 1996).
  - "It helps to further refine table structures and minimize redundant data" (Hernandez, 1996).
  - "It is the mechanism that allows data from multiple tables to be drawn together simultaneously" (Hernandez, 1996)

"In order to take advantage of the many benefits provided by a relational database, you must make certain that you establish each relationship carefully and properly" (Hernandez, 1996).

There are three types of relationships in a relational model: one-to-one (1:1), one-to-many (1:M) or many-to-many (N:M). Once you have decided on what tables you need, you also need to decide on the relationship between those tables. Determining these relationships will determine how you create your foreign keys, etc. For our example, let's assume that we have a one-to-many relationship. In other words, each applicant can only apply to one program per term, but one program may have many applicants per term.

- **Determining Primary Keys**
  Every record in an Oracle table must have a unique identifier. This identifier is called the primary key. The primary key can be made up of one field (column) or can be a group of fields (columns). The primary key is used to join two different tables. Only one primary key can be defined per table and no field (column) in the primary key can contain a Null (blank) value.

  For our example, the APP_SSN will be the primary key for the SIS_APPLICANT_BIODEMO table. APP_SSN and APP_TERM will be the primary key for the SIS_APPLICANT_PROGRAM table. (Assumption: that an applicant can only have one application per term.) The primary key is denoted with an underline.

SIS_APPLICANT_BIODEMO	SIS_APPLICANT_PROGRAM
APP_SSN - PK	APP_SSN - PK
APP_LNAME	APP_TERM - PK
APP_FNAME	APP_COLLEGE
APP_MNAME	APP_MAJOR
APP_SEX	APP_LEVEL
APP_DOB	APP_DEGREE
APP_ETHNIC	

- **Determining Foreign Keys**
  A foreign key can be made up one field (column) or a group of fields (columns). A foreign key provides a reference to rows in another table. The foreign key usually matches the primary key in the table being referenced. When you have a one-to-many relationship (as in our example) the primary key from the one side (SIS_APPLICANT_PROGRAM) gets added to the many side (SIS_APPLICANT_BIODEMO) as a foreign key. For our example, we are assuming that an applicant can only apply to one program per term. Therefore, we can create a foreign key as follows:

SIS_APPLICANT_BIODEMO	SIS_APPLICANT_PROGRAM
APP_SSN - PK	APP_SSN – PK, FK
APP_LNAME	APP_TERM - PK
APP_FNAME	APP_COLLEGE
APP_MNAME	APP_MAJOR
APP_SEX	APP_LEVEL
APP_DOB	APP_DEGREE
APP_ETHNIC	

In this example, because APP_SSN is part of the primary key in both tables this automatically acts as the foreign key. When you have a many-to-many relationship a new table is created to represent that relationship. This table would consist of the primary keys from both tables.

- **Converting Data Types**

  At this point you have decided on the names for your tables and fields. Now you need to decide on the appropriate Oracle data type for each of your fields.

  Let's take a look at our example:

SIS_APPLICANT_BIODEMO	COBOL DATATYPE	ORACLE DATAYPE
APP_SSN - PK	PIC X(09)	VARCHAR2(9)
APP_LNAME	PIC X(20)	VARCHAR2(20)
APP_FNAME	PIC X(15)	VARCHAR2(15)
APP_MNAME	PIC X(08)	VARCHAR2(8)
APP_SEX	PIC X(01)	VARCHAR2(1)
APP_DOB	PIC X(08)	VARCHAR2(8)
APP_ETHNIC	PIC X(01)	VARCHAR2(1)
PROGAM_APP_SSN	PIC X(09)	VARCHAR2(9)
APP_TERM - FK	PIC X(05)	VARCHAR2(5)
SIS_APPLICANT_PROGRAM		
APP_SSN – PK	PIC X(09)	VARCHAR2(9)
APP_TERM - PK	PIC X(05)	VARCHAR2(5)
APP_COLLEGE	PIC X(02)	VARCHAR2(2)
APP_MAJOR	PIC X(04)	VARCHAR2(4)
APP_LEVEL	PIC X(01)	VARCHAR2(1)
APP_DEGREE	PIC X(05)	VARCHAR2(5)

This example only contains a COBOL data type of alphanumeric. However, there are other data types available in COBOL. COBOL data types include: alphanumeric, alphabetic, and numeric. We use alphanumeric (PIC X) and numeric (PIC 9) the most. Below are more examples to help you in converting your COBOL data types to Oracle data types.

COBOL DATATYPE	**ORACLE DATATYPE**
PIC X(30)	VARCHAR2(30)
PIC 9(09)	NUMBER(9)
PIC 9(02)V99	NUMBER(4,2)
PIC 9(04)V99	NUMBER(6,2)

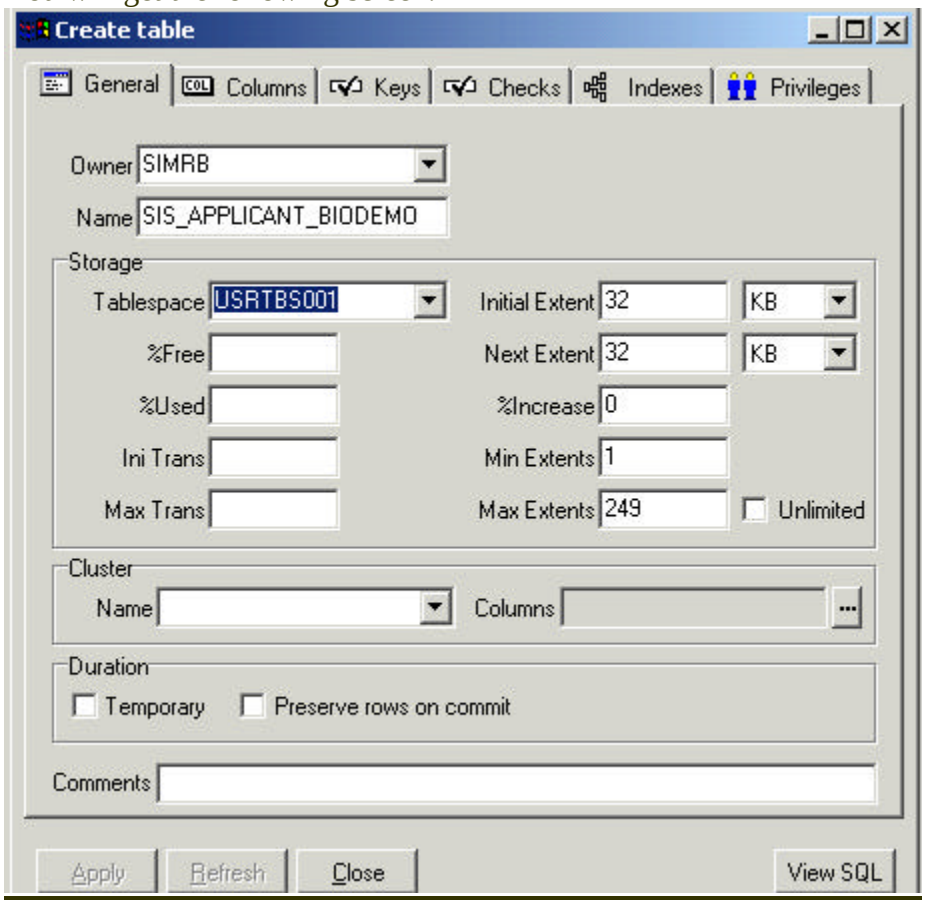
```
VARCHAR2(size) – variable-length character data.
NUMBER(p,s) – variable-length numeric data. P = precision,
S = Scale
```

Oracle has a DATE data type that you may find useful for storing dates. Most of our dates are stored as alphanumeric in COBOL.

- **Creating tables in PL/SQL Developer**
  Now, you can begin to create your tables in PL/SQL developer. PL/SQL developer is a GUI product unlike SQLPLUS.

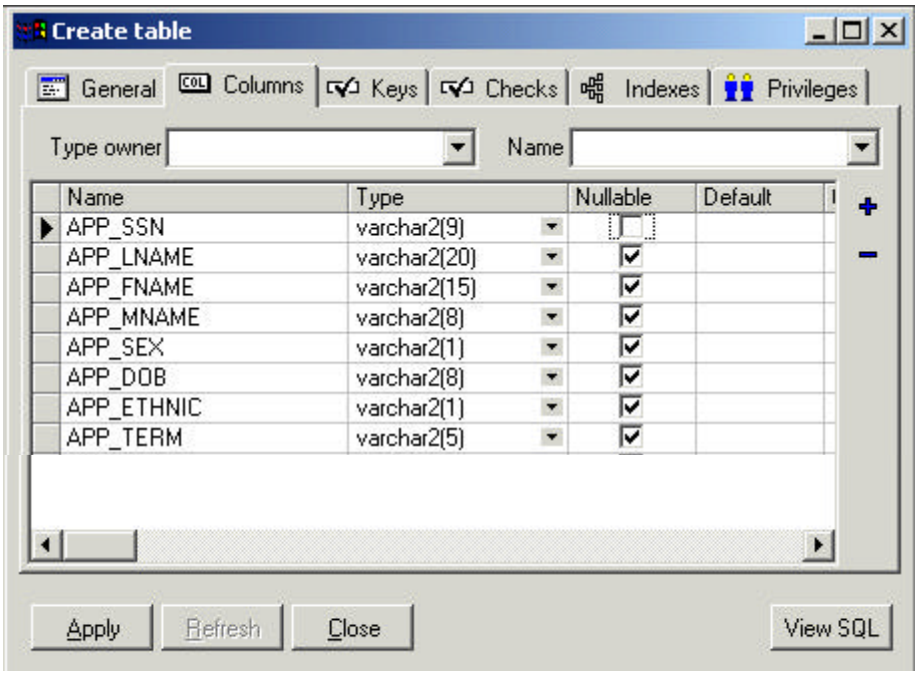
  To create a new table:

  1) Go to File → New → Table
  You will get the following screen:



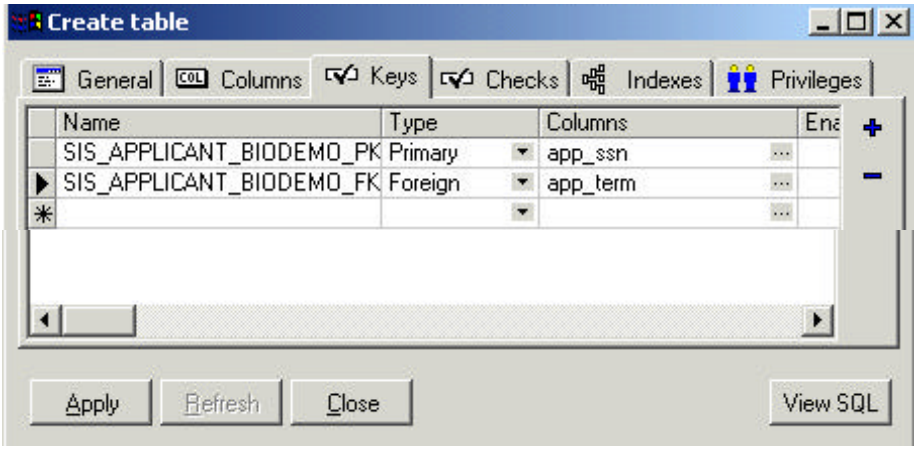
On this screen you specify the owner and name of the table. You also need to select the appropriate tablespace your table should be created in. The drop down box will contain all of the tablespaces you have access to. Once you select the tablespace, the fields on the left will be automatically filled in. Don't worry about any of the other fields on this screen.

2) Next go to the Columns tab:



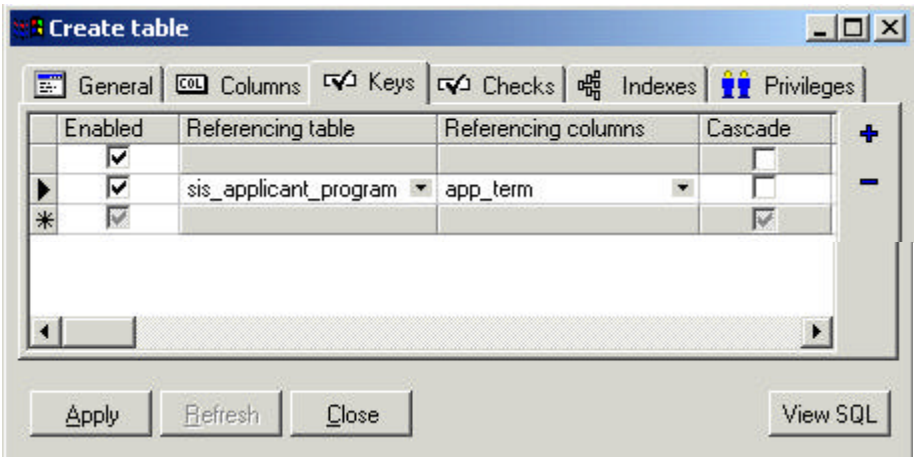
This screen is where you define your fields (columns) that will be in your table. You specify the name of the field, the data type, and whether it is nullable. Notice that APP_SSN is not checked as nullable because the primary key cannot be NULL.

3) Next go to the Keys tab:



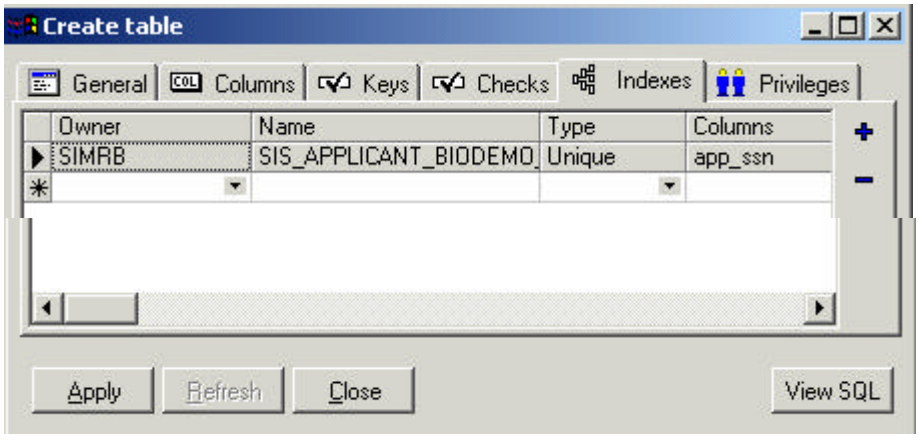
On this screen you will define your primary and foreign keys. You will give the key a name. In this example, the name is the table name with PK added on the end. You then must specify the column for the primary key. For the foreign key you do the same thing.

On this same screen you need to scroll over to specify the referencing table and column for the foreign key. In this example, the referencing table is SIS_APPLICANT_PROGRAM and the referencing column is APP_TERM.

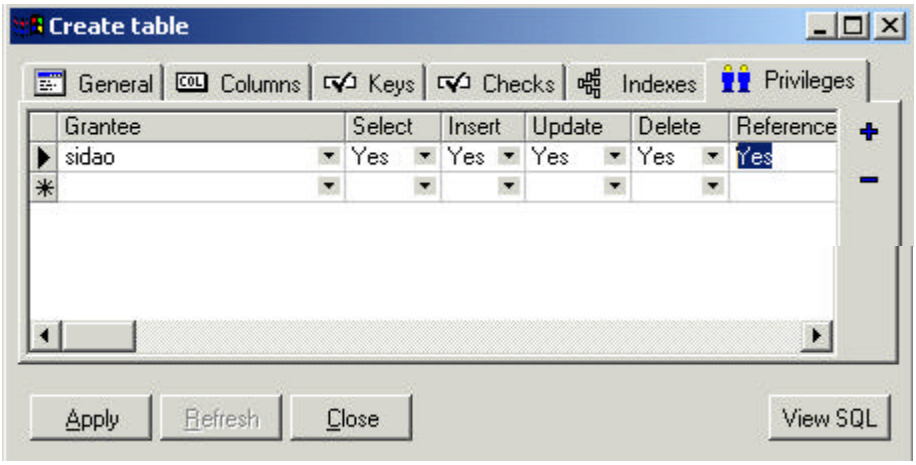


4) Next go to the Indexes tab:

This is where you create any indexes on the fields in your tables. Indexes are used for querying the table and helps speed up queries. Notice here that an index was automatically created for the primary key.



5) Next go to the Privileges tab:



This is where you grant access to the table you are creating. For example, I'm granting sidao access to the SIS_APPLICANT_BIODEMO table. I'm giving select, insert, update, delete, and reference authority.

6) Once you have finished, you click the Apply button to create your table. Now your table is created. Your table will now appear on the left hand side under the Tables folder. In order to make changes, you must edit the table by right clicking on the table name and choosing edit.

COBOL/CICS code changes

1) Overview of the structure of SQL statements in COBOL.

SQL statements consist of the following three elements:

Keyword pair
EXEC SQL

Statement string
Any valid SQL statement

Statement terminator
END-EXEC.

The following rules apply to SQL statements embedded in COBOL:
(Note: These rules were taken from the IBM Application Development Guide at
http://www-3.ibm.com/software/data/db2/udb/ad/v7/adg/db2a0/frame3.htm#db2a0181)

- Executable SQL statements must be placed in the PROCEDURE DIVISION. The SQL statements can be preceded by a paragraph name just as a COBOL statement.

- Start each SQL statement with EXEC SQL and end it with END-EXEC. The SQL precompiler includes each SQL statement as a comment in the modified source file.
- You must use the SQL statement terminator. If you do not use it, the precompiler will continue to the next terminator in the application. This may cause indeterminate errors.
- SQL statements follow the same line continuation rules as the COBOL language. However, do not split the EXEC SQL keyword pair between lines.
- Do not use the COBOL COPY statement to include files containing SQL statements. SQL statements are precompiled before the module is compiled. The precompiler will ignore the COBOL COPY statement. Instead, use the SQL INCLUDE statement to include these files.
- SQL arithmetic operators must be delimited by blanks.
- Full-line COBOL comments can occur anywhere in the program, including within SQL statements.
- Use host variables exactly as declared when referencing host variables within an SQL statement.

2) Code changes for your COBOL program

1) Change the version of your program

The version of your program will change from 23 to 25 for CICS/Oracle. Here are the version numbers for any batch programs that may need to read Oracle:

- 35 – Oracle (batch)
- 36 – Oracle/IDMS (batch)

2) Add the following required SQL statements to your program in Working Storage for declaring host variables.

```
EXEC SQL
   BEGIN DECLARE SECTION
END-EXEC.

EXEC SQL
   END DECLARE SECTION
END-EXEC.
```

These lines of code allow you to define variables called host variables to be used by the SQL statements in your program. Host variables allow an application to pass input data to the database manager and receive output data from the database manager. Any host variables that need to be used in your SQL statements need to be declared in between these lines of code. For example, the applicant's name needs to be available for use by the SQL statements.

Here's how you would use the applicant's name as a host variable:

```
EXEC SQL
   BEGIN DECLARE SECTION
END-EXEC.

SQL-APPLICANT-NAME      PIC X(32).

EXEC SQL
   END DECLARE SECTION
END-EXEC.
```

You can name the host variables whatever you want as long as the name complies with the naming conventions of COBOL. However, I found it helpful to add SQL to the name to make it clearer that the field is a host variable for SQL.

3) The following SQL also needs to be added to your Working Storage section.

```
EXEC SQL
   INCLUDE SQLCA
END-EXEC.
```

The SQLCA file defines the SQL Communication Area structure. The SQLCA contains variables that are used by the database manager to provide an application with error information about the execution of SQL statements.

4) Add the following code to connect to the Oracle database.
You need to add a host variable for the Oracle login to the SQL declare section that you added in step 2. An example is below.

```
EXEC SQL
   BEGIN DECLARE SECTION
END-EXEC.

01  USER-ID             PIC X(11)
                        VALUE 'XXGFB/XXGFB'.

EXEC SQL
   END DECLARE SECTION
END-EXEC.
```

The value of USER-ID is the generic username/password for the ORD1 Iinstance. (Note: you do not have to specify what database to connect to within the USER-ID fields because ORD1 is the default for CICS. However, you may see this done in batch programs).

You will also need to add a paragraph to your procedure division like the one below:

```
200-CONNECT-TO-ORACLE.
    EXEC SQL
       CONNECT :USER-ID
    END-EXEC.

    IF SQLCODE = 0
       NEXT SENTENCE
    ELSE
      PERFORM 910-SQL-ERROR.

200-EXIT.
    EXIT.
```

Notice there is a colon (:) in front of the host variable USER-ID. This is how you reference the host variables that are declared in the SQL declare section. There is also some error checking in this example that will be discussed later.

5) Next you should identify all of the I/O statements for your VSAM file. Go through your program and make note of all of the I/O statements that make reference to your VSAM file. You can easily find all of the I/O statements by doing a search on the CICS cluster name for your file. VSAM I/O statements include READ, WRITE, DELETE, and REWRITE. For now, just comment out the code that accesses the VSAM file instead of deleting it and add in the appropriate SQL. Delete the VSAM code once you are sure your program works properly with the SQL.

The table below gives an overview of how VSAM I/O statements map to SQL statements.

VSAM I/O operation	SQL statement
READ <FD name>	SELECT * INTO <host variable list> FROM <table name> WHERE <key> = < :record key variable>
WRITE <FD name>	INSERT INTO <table name> <column list> VALUES (host variable list)
DELETE <FD name>	DELETE FROM <table name> WHERE <key> = < :record key variable>
REWRITE <FD name>	UPDATE <table name> SET {column = <scalar expression>} WHERE <key> = < :record key variable>

This table gives the purpose of each of the primary SQL operations.

SQL Statement	Purpose
SELECT	To retrieve data from one or more tables, object tables, views, object views, or materialized views.
INSERT	To add rows to a table, a view's base table, a partition of a partitioned table or a subpartition of a composite-partitioned table, or an object table or an object view's base table.
DELETE	To remove rows from a table, a partitioned table, a view's base table, or a view's partitioned base table.
UPDATE	To change existing values in a table or in a view's base table.

Think back to the tables we created earlier when looking at steps 7 –10. The fields (columns) for each table need to be defined as host variables in the SQL declare section. For example:

```
EXEC SQL
   BEGIN DECLARE SECTION
END-EXEC.

01  SQL-APP-REC.
    03  SQL-APP-SSN     PIC X(09).
    03  SQL-APP-LNAME   PIC X(20).
    03  SQL-APP-FNAME   PIC X(15).
    03  SQL-APP-MNAME   PIC X(08).
    03  SQL-APP-SEX     PIC X(01).
    03  SQL-APP-DOB     PIC X(08).
    03  SQL-APP-ETHNIC  PIC X(01).
    03  SQL-APP-TERM    PIC X(05).

EXEC SQL
   END DECLARE SECTION
END-EXEC.
```

Numbers 7-10 show you how to convert VSAM I/O statements to SQL. Some things to know about these examples: 1) they are simplified for example purposes, only part of the code is actually shown; and 2) the word DBLink is used to represent where you would put the actual name of the DBLink you are using (DBLink is not the real name); and 3) error handling is left out and will be discussed later in the documentation; and 4) Also notice how I prefix the table name with SIMRB (my login for the Oracle instance). This is because the tables I am referencing are owned by me and created within my schema.

6) Convert any READ statements to SQL.

The example below takes a piece of code that reads a VSAM file and shows how you would then read the Oracle tables using SQL. To read an Oracle table you must create what is called a Cursor. You use the SELECT statement to retrieve all of the data from the table. In the SELECT statement, you must specify what fields (columns) to retrieve. In this example, we are retrieving all of the fields by using the *. You can also specify a WHERE clause to restrict the retrieval of records to the ones that satisfy the condition. This example omits the WHERE statement because we want to retrieve all records.

VSAM I/O operation	SQL statements

200-READ-APPFILE.
    MOVE CS-SSN TO APP-SSN.
 	
    MOVE 'READ' TO CICS-CMD.
    MOVE +2525 TO REC-LENGTH.
 	
   EXEC CICS READ
       DATASET ('SIGRAPPL')
       INTO   (APP-REC)
       LENGTH (REC-LENGTH)
       RIDFLD (APP-REC-KEY)
       EQUAL
    END-EXEC.
 	
200-EXIT.
    EXIT.
	200-READ-APPFILE.
	    EXEC SQL
	       DECLARE APPCURSOR CURSOR FOR
	       SELECT *
	          FROM SIMRB.SIS_APPLICANT_BIODEMO@DBLink
	    END-EXEC.
	
	    EXEC SQL
	       OPEN APPCURSOR
	    END-EXEC.
	
	    MOVE 'N' TO APP-IN-EOF
	    PERFORM 201-FETCH-APPCURSOR
	        UNTIL APP-IN-EOF = 'Y'.
	
	    EXEC SQL
	       CLOSE APPCURSOR
	    END-EXEC.
	
	200-EXIT.
	    EXIT.
	
	201-FETCH-TRANSCURSOR.
	
	    EXEC SQL
	       FETCH APPCURSOR
	       INTO :SQL-APP-REC
	
	        EXIT WHEN APPCURSOR%NOTFOUND OR
	                  APPCURSOR IS NULL
	    END-EXEC.
	
	201-EXIT.
	    EXIT.

7) Convert any WRITE statements to SQL.

The example below takes a piece of code that writes to a VSAM file and converts it to write to Oracle tables. In this example, we are inserting data into the SIS_APPLICANT_BIODEMO table we created earlier. You first have to move the incoming data from the web application to the host variables. Then you can use the host variables in the Insert statement. For this example I did not include the column list in the Insert statement. This is optional, if you omit it you must specify a value for all of the columns in the table in the order in which they appear in the table.

VSAM	ORACLE
250-WRITE-APPFILE. MOVE CS-SSN TO APP-SSN. MOVE CS-LNAME TO APP-LNAME. MOVE CS-FNAME TO APP-FNAME. MOVE CS-MNAME TO APP-MNAME. MOVE CS-SEX TO APP-SEX. MOVE CS-DOB TO APP-DOB. MOVE CS-ETHNIC TO APP-ETHNIC. MOVE 'WRITE' TO CICS-CMD. EXEC CICS WRITE FILE ('SIGRAPPL') FROM (APP-REC) RIDFLD (APP-REC-KEY) END-EXEC. 250-EXIT. EXIT.	250-WRITE-APPFILE. MOVE CS-SSN TO SQL-APP-SSN. MOVE CS-LNAME TO SQL-APP-LNAME. MOVE CS-FNAME TO SQL-APP-FNAME. MOVE CS-MNAME TO SQL-APP-MNAME. MOVE CS-SEX TO SQL-APP-SEX. MOVE CS-DOB TO SQL-APP-DOB. MOVE CS-ETHNIC TO SQL-APP-ETHNIC. EXEC SQL INSERT INTO SIMRB.SIS_APPLICANT_BIODEMO@DBLink VALUES (:SQL-APP-SSN, :SQL-APP-LNAME, :SQL-APP-FNAME, :SQL-APP-MNAME, :SQL-APP-SEX, :SQL-APP-DOB, :SQL-APP-ETHNIC) END-EXEC.

8) Convert any DELETE statements to SQL.

The example below takes some code that deletes a record from a VSAM file and converts it to delete a record from an Oracle table. You must move the value of the key to a host variable (in this case the SSN is the key). The host variable is used in the WHERE part of the DELETE statement. This specifies which record (row) you would like to delete from the table. WARNING: If you do not specify the WHERE clause every record (row) in the table will be deleted!!!

VSAM	ORACLE
300-DELETE-APPFILE. MOVE SPACES TO APP-REC-KEY. MOVE CS-SSN TO APP-SSN. EXEC CICS DELETE DATASET ('SIGRAPPL') RIDFLD (APP-REC-KEY) END-EXEC. 300-EXIT. EXIT.	300-DELETE-APPFILE. MOVE CS-SSN TO SQL-APP-SSN. EXEC SQL DELETE FROM SIMRB.SIS_APPLICANT_BIODEMO@DBLink WHERE APP_SSN = :SQL-APP-SSN END-EXEC. 300-EXIT. EXIT.

9) Convert any REWRITE statements to SQL.

The example below takes some code that rewrites a record to a VSAM file and converts it to “rewrite” a record in an Oracle table. The SET statement specifies what field (column) in the table you want to update and what value to update it to. You must specify what record needs to be updated in the table by referring to its key. Therefore, you must move the key to the host variable to be used in the WHERE statement. The WHERE statement specifies what record will be updated in the table.

WARNING: If you do not specify the WHERE clause every record (row) in the table will be updated!!!

VSAM	ORACLE
350-REWRITE-APPFILE. MOVE CS-SSN TO SQL-APP-SSN. MOVE 'Florio' TO SQL-APP-LNAME. EXEC CICS REWRITE DATASET ('SIGRAPPL') FROM (APP-REC) LENGTH (REC-LENGTH) END-EXEC. 350-EXIT. EXIT.	350-REWRITE-APPFILE. MOVE CS-SSN TO SQL-APP-SSN. MOVE 'Florio' TO SQL-APP-LNAME. EXEC SQL UPDATE SIMRB.SIS_APPLICANT_BIODEMO@DBLink SET APP_LNAME = :SQL-APP-LNAME WHERE APP_SSN = :SQL-APP-SSN END-EXEC. 350-EXIT. EXIT.

10) Add this required code to COMMIT your changes to the database.
Add a paragraph like the one below to your procedure division. You only need to perform this paragraph once after you have finished making all of the changes to your tables (this includes inserts, deletes, updates, etc. to your tables). A commit will make permanent all changes performed in the transaction. A transaction is a sequence of SQL statements that Oracle treats as a single unit.

```
600-COMMIT-WORK.
    EXEC SQL
      COMMIT WORK RELEASE
    END-EXEC.

600-EXIT.
    EXIT.
```

11) Add the appropriate error handling to your program.
Add a paragraph like the one below to your procedure division.

```
910-SQL-ERROR
    EXEC SQL
       ROLLBACK WORK RELEASE
    END-EXEC.

    PERFORM 999-ENTRY-ERROR.

910-EXIT.
    EXIT.

999-ENTRY-ERROR.
    MOVE HTML-EMSGS TO CS-RETURN.
    GO TO IDMS-STATUS.

999-EXIT.
    EXIT.
```

The purpose of a rollback is to undo any changes that have occurred within the transaction. Whenever you encounter an error, you want to call this paragraph so your changes can be rolled back. You've seen a paragraph like 999-Entry-Error before. This is how we send error messages back to the web for the user. If an error occurs and a roll back is needed, you still need to send an error message to the user by calling paragraph 999-ENTRY-ERROR from 910-SQL-ERROR. The actual message that you send to the user will be moved to HTML-EMSGS when the error is captured. Look at how error handling is added to the delete example given earlier. The SQLCODE is a field within the SQLCA member you are including in Working Storage.

```
300-DELETE-APPFILE.
    MOVE CS-SSN TO SQL-APP-SSN.

     EXEC SQL
       DELETE FROM
         SIMRB.SIS_APPLICANT_BIODEMO@DBLink
       WHERE APP_SSN = :SQL-APP-SSN
    END-EXEC.

    IF SQLCODE = 0
       NEXT SENTENCE
    ELSE
      MOVE 'ERROR ON DELETE OF RECORD' TO HTML-EMSGS
      PERFORM 910-SQL-ERROR.

 300-EXIT.
     EXIT.
```

Testing your system

Now that you have your COBOL code changes completed, you can begin to test your application in the test environment. The new converted system needs to be tested for data integrity, performance, and functionality.

1. Data Integrity

Your new system needs to be tested to make sure it stores the data properly. To verify data integrity, you need to compare the data in the current VSAM file against the data in the new Oracle tables. Check the following VSAM field types in Oracle:

- Alpha – PIC X
- Numeric – PIC 9
- Date formats – CCYYMMDD, MMDDCCYY, CCYY, ETC.
- Decimal – PIC 9V99
- Required fields

You also need to make sure your new system can handle data when it receives bad data. Handling bad data can be accomplished by validation routines that

check the data before it is written to the Oracle tables. You should also check that your new system can handle errors and that the appropriate error messages are sent back to the user.

2. **Performance**
The goal is to make sure the new system performs at the same level as the old system. Depending on the functionality of your system, you need to run your system to test the performance. Run your old system and record the response times and then run your new system with the same criteria and also record the response times. You then want to compare the response times from both systems. You may also want to stress test your new system to make sure the new system performs properly during high traffic times.

3. **Functionality**
You also want to make sure that your system did not lose any functionality during the conversion. For example, if your old system allowed the user to delete a record, then the new system should also allow the deletion of a record. For functionality you should perform unit and integration tests. Unit tests should be performed for each program that needs to be tested. Integration tests should test the converted system as a whole.

Moving system to production

- **Obtain approval from database group**
Before you move your tables to production, you must get approval from the database administrator. Once your tables are approved, you can then create your tables in production.

- **Create tables in OP07**
This is the production Oracle instance that corresponds to the OD07 test instance. You will create your tables in this instance for production. Below is the text to add to your *tnsnames.ora* file for OP07:

```
OP07 =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = riogrande.ais.unc.edu)(PORT =1521))
  )
  (CONNECT_DATA =
   (SID = OP07)
  )
 )
```

If you don't have a login for this instance contact our FACS coordinator Kay Bobbitt to get a login.

- **DBLink**
The ORP1 instance is the default production instance for CICS. You will need to use a DBlink to link over to OP07 from ORP1. This will be a different DBlink

than the one you used for test (OD07). Make sure you grant the owner of the DBlink the appropriate access to your tables. Below is the text to add to your *tnsnames.ora* file for ORP1:

```
ORP1 =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = mvs.ais.unc.edu)(PORT =3052))
  )
  (CONNECT_DATA =
   (SID = ORP1)
  )
 )
```

- **Backup current file and processes**
  One thing you should do before you move any of your modified programs to production is to make backups of the programs. You should backup all programs and copy members that you modified. You should also archive any VSAM files that were converted to Oracle tables.

- **Migrate VSAM data to Oracle tables**
  You can migrate the old VSAM data to the new Oracle tables using a utility called SQL*Loader. SQL*Loader is a bulk loader utility used for moving data from external files into the Oracle database tables.

  Below is an example using SQL*Loader:

```
LOAD DATA
INTO TABLE SIS_APPLICANT_BIODEMO
(APP_SSN          POSITION(1:9),
 APP_LNAME        POSITION(10:30),
 APP_FNAME        POSITION(31:46),
 APP_MNAME        POSITION(47:55),
 APP_SEX          POSITION(56:57),
 APP_DOB          POSITION(58:66),
 APP_ETHNIC       POSITION(67:68))
```

  The field name on the left corresponds to the column name in the Oracle table. The position on the right is the position of field in the VSAM file.

- **Conduct final test in production**
  Since there are slight differences between our test and production environments, one final test in production needs to be conducted. This is to ensure the move to production went smoothly.

Glossary of terms

- **Attribute** - A single data item related to a database object. The database schema associates one or more attributes with each database entity.
- **_____** - Database tables are composed of individual columns corresponding to the attributes of the object.
- **DBlink** - the complete or partial name of a database link to a remote database where the table or view is located.
- **Entity** - An entity is a single object about which data can be stored. It is the "subject" of a table. Entities and their interrelationships are modeled through the use of entity-relationship diagrams.
- **ER Diagram** - An entity-relationship diagram is a specialized graphic that illustrates the interrelationships between entities in a database.
- **Foreign Key** - A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.
- **Join** - The SQL JOIN statement is used to combine the data contained in two relational database tables based upon a common attribute.
- **Key** - A database key is a attribute utilized to sort and/or identify data in some manner.
- **Normalization** - Normalization is the process of structuring relational database schema such that most ambiguity is removed.
- **Null** - The NULL SQL keyword is used to represent either a missing value or a value that is not applicable in a relational table.
- **Primary Key** - key of a relational table that uniquely identifies each record in the table.
- **Query** - Queries are the primary mechanism for retrieving information from a database and consist of questions presented to the database in a predefined format.
- **Record** - In a relational database, records correspond to rows in each table.
- **Relation** - A database relation is a predefined row/column format for storing information in a relational database. Relations are equivalent to tables.

Relational Database a collection of data items organized as a set of formally described tables from which data can be accessed or reassembled in many

Row - In a relational databas
corresponding to one instance of the entity that a table schema describes.

- **Select** The SELECT statement in SQL is the primary mechanism for retrieving information from a relational database.

- **SQL** – nteractive and programming language for getting information from and updating a database.

- **SQL*LOADER** a bulk loader utility used for moving data from external files

- **Table** - A table in a relational database is a predefined format of rows and columns that define an entity.

Other Resources

IBM, Application Development Guide
http://www-3.ibm.com/software/data/db2/udb/ad/v7/adg/db2a0/frame3.htm#db2a0181

Oracle DataTypes
http://technet.oracle.com/doc/server.815/a68003/01_04blt.htm

Oracle - SQL Reference
http://technet.oracle.com/doc/server.815/a67779/toc.htm

Oracle - PL/SQL Reference
http://otn.oracle.com/docs/products/oracle8i/doc_library/817_doc/appdev.817/a77069/toc.htm

Database Design Links
http://databases.about.com/cs/specificproducts/index.htm?terms=relation

If AIS decides to buy the Safari online book service, I highly recommend everyone taking a look at Database Design for Mere Mortals by Michael J. Hernandez.
http://safari.oreilly.com/main.asp?bookname=0201694719