1997

International student services opportunity database (ISSODB)

Sailaja Malireddy

Follow this and additional works at: http://scholarworks.lib.csusb.edu/etd-project

Part of the Computer Sciences Commons

Recommended Citation
http://scholarworks.lib.csusb.edu/etd-project/1408

This Thesis is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
INTERNATIONAL STUDENT SERVICES

OPPORTUNITY DATABASE

(ISSODB)

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Sailaja Malireddy

March 1997
INTERNATIONAL STUDENT SERVICES

OPPORTUNITY DATABASE

(ISSODB)

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Sailaja Malireddy

March 1997

Approved by:

Dr. Josephine Mendoza, Chair, Computer Science

Dr. George M. Georgiou

Dr. Tong L. Yu

3-17-97

Date
ABSTRACT

The International Student Services (ISS) is the primary contact for all international students who arrive at California State University, San Bernardino (CSUSB). These students, relatively new in the United States generally experience difficult conditions, which require them to quickly adapt to local academic and societal environments. The ISS plays a paramount role in assisting these students with information related to academic requirements and opportunities, immigration requirements, housing, health, and various other support services. The ISS also provides programs and activities that assist students with career and financial aid opportunities. In order to meet its objectives, ISS is required to collect, organize, and process enormous amount of practical information that is very valuable to the international students. Currently, ISS records most of this information using paper-based methods. The purpose of this project is to develop a computerized database management system (ISSODB) that will hold and offer detail information pertinent to fellowships, summer internships, scholarships, grants, loans, and study abroad programs. This database offers several benefits over the current file management system: allows multiple user access; reduces data redundancy; provides speedy recovery of data; maintains data integrity; allows security restrictions; and balances conflicting requirements. The database was designed using the entity-relationship (E-R) model concepts, and was implemented in ORACLE version 7. The database executes as an integrated menu application (designed using SQL*Menu). The database also includes user-friendly data entry forms (designed through SQL*Forms), and several SQL queries that generate organized reports of user requested information. The database was validated for its functionality and is ready for practical use by ISS staff and international students.
ACKNOWLEDGMENTS

I thank the faculty of Computer Science department for giving me an opportunity to pursue my M.S. in Computer Science at California State University, San Bernardino. I express my sincere appreciation to my graduate advisor, Dr. Josephine Mendoza who directed me through this entire effort. I also thank my other committee members, Dr. George M. Georgiou and Dr. Tong L. Yu for their valuable input. I would like to thank the International Student Services director Mrs. Elsa Fernandez who created the need for this project and Mr. Kenneth Nichols who partnered with me during the initial phase of the project.
# TABLE OF CONTENTS

ABSTRACT ........................................................................................................... iii

ACKNOWLEDGMENTS ...................................................................................... iv

1.0 INTRODUCTION ......................................................................................... 1

1.1 PURPOSE OF THIS PROJECT ........................................................................ 2

1.2 PROJECT PRODUCTS .................................................................................... 2

2.0 DATABASE REQUIREMENTS SPECIFICATION ............................................... 4

2.1 INTRODUCTION ............................................................................................ 4

2.1.1 Project Approach ...................................................................................... 4

2.1.2 Project Goals/Output ............................................................................... 5

2.2 PROJECT COMPONENTS .............................................................................. 5

2.3 FUNCTIONAL PARTITIONING ...................................................................... 8

2.4 FUNCTIONAL DESCRIPTION ........................................................................ 8

2.5 VALIDATION CRITERIA .................................................................................. 10

2.5.1 Performance Bounds ............................................................................... 10

2.5.2 Classes of Tests ....................................................................................... 10

2.5.3 Expected Software Response .................................................................. 11

2.5.4 Special Considerations ............................................................................ 11

3.0 PROJECT APPROACH .................................................................................. 12

3.1 CONCEPTUAL DESIGN PHASE .................................................................. 12

3.2 PHYSICAL DESIGN PHASE .......................................................................... 13

3.3 DATABASE VALIDATION ............................................................................. 13

4.0 CONCEPTUAL DESIGN ................................................................................ 15

4.1 DATABASE REQUIREMENTS COLLECTION AND ANALYSIS ..................... 15

4.2 DEVELOPMENT OF ENTITY-RELATIONSHIP (E-R) MODEL ...................... 17
4.2.1 Identify Entities and Associated Attributes ........................................... 18
4.2.2 Identify Relationships ........................................................................... 23
4.3 DERIVATION OF FUNCTIONAL DEPENDENCIES ................................ 27
4.4 DEVELOPMENT OF BASE RELATIONS .................................................... 29
4.5 NORMALIZATION OF BASE RELATIONS .................................................. 31
5.0 PHYSICAL DESIGN ..................................................................................... 35
5.1 CREATION OF TABLES ............................................................................. 35
5.2 DATA LOADING ......................................................................................... 42
5.3 DESIGN OF SQL*FORMS ........................................................................ 44
  5.3.1 Steps Used to Create Forms ................................................................. 46
  5.3.2 Steps Used to Create Blocks ................................................................. 48
  5.3.3 Steps Used to Test the Form ................................................................. 49
  5.3.4 Steps to Modify the Form .................................................................... 50
5.4 DESIGN OF REPORTS ............................................................................... 52
5.5 DESIGN OF SQL*MENUS ....................................................................... 53
6.0 DATABASE VALIDATION ........................................................................... 57
  6.1 UNIT TESTING ......................................................................................... 57
  6.2 INTEGRATION TESTING ......................................................................... 57
  6.3 SYSTEM TESTING .................................................................................. 59
7.0 USER INSTRUCTIONS .................................................................................. 60
APPENDIX A: Sample Data Sheet ................................................................. 64
APPENDIX B: ISSODB Data Entry Forms ....................................................... 66
APPENDIX C: SQLPLUS Code for Reports and Sample Output ................... 75
APPENDIX D: List of Acronyms ..................................................................... 84
REFERENCES ................................................................................................. 86
LIST OF TABLES

4.2.1 Attribute Details for the Opportunity Entity .............................................. 20
4.2.2 Attribute Details for the Requirements Entity ............................................ 20
4.2.3 Attribute Details for the Application Recipient Entity ............................... 21
4.2.4 Attribute Details for the Sponsor Entity .................................................. 22
4.2.5 Attribute Details for the Country Entity .................................................. 22
5.3.1 Function Key Definitions ........................................................................... 47
6.1.1 Unit Test Results ..................................................................................... 58
6.2.1 Integration Test Results ............................................................................ 59

LIST OF FIGURES

4.2.1 ISSODB Entity-Relationship (E-R) Diagram .................................................. 26
7.1 Main Menu for ISSODB Application ............................................................... 60
7.2 Available Forms Under the Data Entry Option .............................................. 61
7.3 List of Available Reports .............................................................................. 63
1.0 INTRODUCTION

The International Student Services (ISS) office at California State University, San Bernardino (CSUSB) is the primary contact for international students who are nationals of other countries[8]. The ISS office assists students upon arrival with locating temporary and permanent housing, provides an orientation to the community and the campus including information about academic requirements and opportunities, immigration requirements, housing, health, and support services available to the student. The ISS office also provides programs and activities to assist students with personal growth and development[8]. Assistance and referrals are available for academic and personal counseling. The ISS office works closely with the international clubs and the various nationality organizations in bringing student opportunity information as well as cultural events. As a result, the ISS office is required to collect, organize, and maintain an extensive amount of practical information that is very useful to international students. These requirements clearly drive the need for a computerized database system that allows a greater level of flexibility in the organization and storage of data, maintenance of data, and implementation of the database system. This need for a more versatile database system initiated a dialogue between Ms. Elsa Fernandez, Director, ISS and Dr. Josephine Mendoza, Professor, Department of Computer Science. This dialogue between Ms. Fernandez and Dr. Mendoza led to the shaping of this project which called for the development of International Student Services Opportunity Database (ISSODB).
1.1 PURPOSE OF THIS PROJECT

The purpose of this project is to design, build, and implement an information retrieval database system for ISS at CSUSB. The database will focus and offer detailed information to international students on practical and financial opportunities that are available and maintained by ISS staff. The database will offer detailed information pertinent to fellowships, summer internships, scholarships, grants, loans, and study abroad programs. The main purpose of selecting the database approach over the existing file management system is due to specific advantages that accrue from the notion of centralized control of data. A few of these advantages are as follows: redundancy can be reduced; data can be shared; multiple user access; standards can be enforced; security restrictions can be applied; data integrity can be maintained; and conflicting requirements can be balanced. The database will be implemented using a commercial relational database management system (ORACLE Version 7.0). This database will address the immediate needs of ISS and at the same time would also permit the ISS personnel to modify and extend the application in the future.

1.2 PROJECT PRODUCTS

This project would lead to the following products:

- Implementable Database Application: a working database with relevant application programs, which would achieve the specific needs of ISS with respect to storage and retrieval of the student opportunity data. ORACLE
SQL*Forms and SQL*Menu applications will be designed and developed to maximize the user-friendliness of the database system.

- **Users manual**: an implementation manual will be available for the user.
- **Systems Manual**: a project report (this report) will be available with design details and specifications.
2.0 DATABASE REQUIREMENTS SPECIFICATION

2.1 INTRODUCTION

The purpose of the project is to design, build, and implement an information retrieval database for International Student Services (ISS) at California State University, San Bernardino. The database will offer detailed information on Practical and Financial Opportunities maintained by the staff of ISS. This type of information is extremely useful for graduating as well as currently enrolled students pursuing a practical or a financial opportunity. The database will be implemented using ORACLE version 7.

2.1.1 Project Approach

The current database maintained by ISS follows the traditional file processing approach. In the past, an effort was made to convert the traditional file formatted information to an electronic format using dBASE IV. However, this approach remained incomplete and unusable. This situation created the need for a new database, initiating this project. This project is conducted using the concepts of Entity-Relationship model. The project under the direction of its Advisor will go through the following steps in completing the database design.

- collect database requirements
- review and understand the requirements and other details
- develop the conceptual E-R model
  - identify entities
- determine key attributes
- establish relationships, participation levels
- translate the above into E-R diagram
- develop the relational tables

- develop functional dependencies
- normalize relations
- devise formal objects, integrity rules, and formal operators
- implement all of the above using ORACLE 7

2.1.2 Project Goals/Output

Working database application: a database with relevant application programs that meet the immediate needs of the ISS with respect to storing and retrieving student opportunity data. Oracle SQL*Forms and SQL*Menu applications will be designed and developed to achieve and maximize user friendliness.

User manual: a user manual for ISS staff will be written.

System manual: a manual will be written that details the design and implementation of the ISSODB for personnel who may modify and extend the application in the future.

2.2 PROJECT COMPONENTS

The project consists of a system-user interface, which includes menus, pop-up menus, pick lists, and user interaction levels. Each of these components are discussed below.
• **Menus** - the ISSODB will be accessed via a tree-structured menu created by SQL*Menu. This menu will feature a top bar of choices with pull-down sub-menus, a type of interface found in many personal-computer and graphics workstation programs. Each menu selection (with the exception of 'quit') will lead to another menu selection or a data-entry form. Each form will consist of one or more data fields, some of which may be hidden or read-only.

• **Pop-up menus** - some fields will represent data items which may hold one of a limited set of entries. For example, the field opportunity.type may contain only the values grant, scholarship, fellowship, internship, loan, or study abroad. When such a field is selected, a pop-up menu (PUM) containing all valid choices will appear. The PUM will allow the user to choose one of these options:
  
  • inform the user of valid choices,
  
  • eliminate the need to type the chosen item,
  
  • prevent spelling errors,
  
  • prevent invalid entries, and
  
  • enable internal mechanisms for conserving storage space and maintaining the data in Third Normal Form (3NF).

• **Pick lists** - the previous example using opportunity.type is quite simple. A more involved example is provided by examining country, which contains the
country (or set of countries) in which an opportunity is offered. The country is more complex because:

- a single opportunity may be offered in more than one country, so a PUM is not adequate. The user must have a pick list (PL), which is similar to a PUM except that it allows multiple elements from the list to be selected.

- the country named in the opportunity source text may not be on the PL, so the data-entry operator (DEO) will have to determine whether a country from the list can be substituted. The disparities may occur when a country changes its name (e.g., Ceylon to Sri Lanka), splits into two or more countries (e.g., the former Soviet Union), or merges with another country (e.g., the former East and West Germany). In the first case alternatives will be explored and the best solution will be implemented in this project. In the second and third cases, the DEO may have to contact the sponsor for clarification.

- **User levels** - the ISSODB will have three levels of user interaction:
  
  - student queries (a read-only activity),
  
  - routine data entry (entry, update, and delete) of opportunities and related information), and

  - maintenance data entry (the upkeep of data used by the ISSODB but not part of any opportunity).
2.3 FUNCTIONAL PARTITIONING

Unlike a conventional sequential language, such as C, Pascal, or FORTRAN, SQL*Forms organizes an application into segments of code referred to as 'triggers.' Each trigger is associated with a particular element of the form, such as a field, page, or block, and is called only when the user is traversing (entering or leaving) that element. SQL*Forms requires that application functionality be partitioned according to the design of the forms, at least at the code level; there is no explicit division of the code into procedures, modules, libraries, etc.

Within a form, functionality is delineated by dividing the form into blocks and pages. The goal is to design each form to provide a natural and logical user interface. Within the application, functionality is divided by the elements of the user menu. This menu may be thought of as a tree, each of whose leaves is a form (with exception of 'quit'). Again, the goal is to structure the menu tree in such a way that sub-menus contain logically-related operations and frequently-used operations are near the root of the tree.

2.4 FUNCTIONAL DESCRIPTION

- **Processing narrative** - the ISS will gather information from various sources about opportunities which may be of interest to students. These opportunities will be entered by the ISS staff, whereupon they will become available for students to access via queries framed on one or more variables, such as the opportunity type or the country where it is offered. The result of student queries will be a formatted report, which may be viewed on-screen or printed.
• **Restrictions/limitations** - the ISSODB will be limited in several ways.
  - **text-mode display** - as mentioned earlier, only text-mode display of menus and forms is available at this time.
  - **text data** - non-text data, such as bitmap displays, cannot be stored in the ISSODB.
  - **limited reports** - students will have a menu of reports to choose from, but will not have the ability to create custom report formats.
  - **limited SQL access** - students will execute queries by selecting a query form and entering the selection criteria, but they will not have access to free-form SQL.
  - **unanticipated data** - the ISS staff may receive opportunity data which does not 'fit' into the existing schema.

• **Performance requirements** - there are no explicit requirements for system processing or response time. Because the likely maximum size of the database is very small relative to ORACLE’s processing capacity, it is anticipated that response time will be good (i.e., less than 5 seconds for updates and less than 1 minute for a query). Unknown variables which could affect response are:
  - **processing load on Delphi** - the Delphi server computer may have other Oracle applications or other UNIX applications running, and
  - **network speed** - the ISS will communicate with the server via the campus computer network, which may at times be overburdened.
2.5 VALIDATION CRITERIA

Prior to putting any application into practical use, it has to be validated first. Validation is the process of verifying the accurate performance of the application. There are several validation criteria that can be applied to a database application, however, ISSODB relatively being a small application would warrant only the following performance checks.

2.5.1 Performance Bounds

As mentioned above, performance is not a critical issue with the ISSODB. Any performance problems will be the result of factors outside of the application and thus are beyond the scope of this project.

2.5.2 Classes of Tests

- **Interface design/usability** - to the extent possible, conform to standards defined in Pressman (473-74).

- **Correctness** - the results of correctly framed queries will return all matching records and only matching records.

- **Path** - each trigger will be tested for correct functioning, both for normal exit and exception exits.

- **Multi-user** - tests will verify that ORACLE’s multi-user features, such as record locking, concurrent read access, etc., work as expected.

- **Communication** - test network link, locally-defined function keys (if any), and modem access (if any).
2.5.3 Expected Software Response

As set out above, the ISSODB is expected to respond very well, the only caveats being those described in Performance requirements.

2.5.4 Special Considerations

- **Small data set** - because the data structures must be inferred from the existing data set, new data entries may contain information which was not anticipated in the original schema, requiring periodic modifications to the data tables and the application.

- **User feedback** - after the ISSODB is deployed in everyday use, it is likely that users will request modifications and enhancements.
3.0 PROJECT APPROACH

The current database maintained by ISS staff is implemented using the traditional file processing approach. In the past, an effort was made to convert the traditional file formatted information to an electronic format using dBASE IV. However, this approach remained incomplete and unusable. Consequently, this situation further strengthened the need for a computerized database management system. A review of different types of data used by ISS revealed that the data constituted distinguishable objects (entities) that can be linked to each other with certain relationships. Therefore, the concept of relational model was chosen as the design approach for the ISSODB. The relational model is a way of representing the data by means of tables and manipulating such a representation by means of operators such as JOIN, SELECT, PROJECT, etc. In general, the design process typically consists of two parallel activities. The first activity involves the conceptual design of the data content and structure of the database, whereas the second activity concerns the physical design of database processing and software applications. A brief summary of each of these design activities is provided below.

3.1 CONCEPTUAL DESIGN PHASE

The goal of this phase is to produce a conceptual schema for the database that is independent to ISSODB. The conceptual schema is a stable description of the ISSODB contents. This activity involved the following components:

- database requirements collection and analysis.
• development of an Entity-Relation (E-R) model. The E-R model is a high level conceptual data model, which displays the individual entity types and their attributes, relationship types, participation levels, and constraints in a diagrammatic format known as the E-R diagram.

• implementation of the E-R model into relations.

• derivation of functional dependencies based on user requirements.

• normalization of the relations.

3.2 PHYSICAL DESIGN PHASE

The physical design of the database is the process of implementing the relational model in a relational database management system (DBMS). For this project ORACLE Version 7 is used. This activity involved the following tasks:

• created the database and its tables.

• loaded the database with data.

• designed and implemented the queries and routine tasks employing user friendly interfaces such as SQL*Forms and SQL*Menu.

3.3 DATABASE VALIDATION

The ISSODB will be validated for several validation criteria. Various classes of tests will be conducted to verify the following database functionalities:

• correctness - the results of correctly framed queries should return all matching records and only matching records.
• multi-user functionality - tests should verify that ORACLE's multi-user feature (e.g. concurrent read access) should work as expected.

• communication - test network link, locally-defined function keys, and modem access.

The following chapters will describe the conceptual design process, physical implementation, and database validation in more detail.
4.0 CONCEPTUAL DESIGN

The objectives of any database design are manyfold: to satisfy user requirements; to organize information in a structured manner; and to support the processing requirements and performance objectives such as response time, processing time, and storage space. The design process begins with the conceptual design phase, which includes a clear definition of the database requirements, contents, structure, interrelationships, and constraints. The product from this conceptual phase will be a high-level data model (conceptual model) that has the following characteristics: expressiveness; simplicity; minimality; diagrammatic representation; and formality[2].

Prior to defining the design parameters, it is critical to identify the entities of interest to ISS and to identify the information to be recorded about those entities. Therefore, the first step is the collection and review of the database requirements.

4.1 DATABASE REQUIREMENTS COLLECTION AND ANALYSIS

In order to design an effective database, the expectations of the users and the intended uses of the database must be clearly understood. A series of meetings were held with ISS staff to discuss these issues. Meetings were held to identify major application areas and user groups that will use the database, review of available data and related documentation, and to analyze the operating environment and processing requirements. It was concluded from these meetings that the focus of the database would be on practical
and financial opportunities. The following is a detailed summary of database contents and user requirements:

**Application area:** the database should hold and offer information related to the following areas.

<table>
<thead>
<tr>
<th>Practical opportunities</th>
<th>Financial opportunities</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Fellowships</td>
<td>- Loans</td>
</tr>
<tr>
<td>- Internships</td>
<td>- Grants</td>
</tr>
<tr>
<td>- Study abroad programs</td>
<td>- Scholarships</td>
</tr>
</tbody>
</table>

**Application users:** the application should be accessible to two sets of users with different objectives.

<table>
<thead>
<tr>
<th>ISS staff</th>
<th>International students</th>
</tr>
</thead>
<tbody>
<tr>
<td>- data entry</td>
<td>- data retrieval</td>
</tr>
<tr>
<td>- data updates</td>
<td>(this set of users do not have write access to the database)</td>
</tr>
<tr>
<td>- maintenance</td>
<td></td>
</tr>
</tbody>
</table>

**Operating environment requirements:** the application is currently supported on UNIX platform (SunOS), however, it is envisioned that sometime in future the application would be ported to PC. The application executes using ORACLE Version 7, SQL*Forms 3.0, and SQL*Menu 5.0. The application can be accessed by multiple users simultaneously.

**Available Data and Data Formats:** data related to study abroad programs, fellowships, internships, loans, grants, and scholarships is currently available at ISS. Individual
attributes related to each data type such as eligibility requirements, deadlines, where and how to apply, etc. are also documented. This information currently is structured in tabular format and is organized in the traditional file management system. Sample data type and data formats are shown in APPENDIX A. In addition to this format, partial information is also available in electronic format stored in dBASE 4. An attempt was made to transfer data from dBASE IV format to ORACLE format, which posed two major challenges: first moving the data without having to re-key it, and second was the re-categorization and the arrangement of data so that only one item of data is stored in each field. Also, the data available in this format was incomplete, and therefore the option of using electronic format was abandoned.

4.2 DEVELOPMENT OF ENTITY-RELATIONSHIP (E-R) MODEL

Following the requirements collection and analysis, this step focuses on the development of the conceptual schema for the ISSODB. The conceptual schema is a concise description of the data requirements of the users and includes detailed description of the data types, relationships, and constraints[2]. The conceptual schema is expressed using concepts of a high-level data model such as an E-R model. The E-R model displays the individual entity types and their attributes, relationship types, participation levels, and physical constraints in a diagrammatic format known as the E-R diagram. An entity represented in the E-R model is a "thing" in the real world with an independent existence. For example, an entity can be an object with physical existence such as a person or an object with conceptual existence such as a company or a job[2]. Each entity has a
specific set of attributes (properties) that describe the entity. Each entity also has an
attribute whose value is distinct from itself. Such an attribute is called the key attribute
and its value can be used to identify each entity uniquely[1]. This constraint prohibits any
two instances of an entity from having the same value for the key value. Some entity
types have more than one key attribute.

4.2.1 Identify Entities and Associated Attributes

Based on the needs of ISS, it was clearly desirable to integrate both practical and
financial opportunity information into the same opportunity database (ISSODB).
Recognizing the above need, entities common to both information types (practical and
financial) were identified.

Entities are classified as either regular or weak entity types. A regular entity is
also referred to as a strong entity. If an entity’s existence is independent of another entity,
then it is known as a regular entity. If an entity’s existence is dependent on some other
type, then the depending entity is called the weak entity. In the ISSODB application, the
entities “sponsor” and “country” are regular entities, whereas all other entities are weak
entities. A regular entity is modeled as a single-lined rectangle, whereas a weak entity is
drawn as a double-lined rectangle.

Attributes are a set of properties that define an entity. For example, in the
ISSODB application, the entity “Sponsor” would possess properties such as sponsor
name, sponsor address, etc. Each of these properties draw their values from a

18
• **Atomic or Composite**: an atomic attribute is a property that is not divisible into sub properties. A composite attribute is made up of several atomic properties (e.g. address is made up of atomic properties like street name, city, state, etc.).

• **Single or Multi-valued**: a property with a single value is called a single-valued attribute (e.g. sponsor_name). Similarly, a property with multiple entry values is called a multi-valued attribute (e.g. fund_name)

• **Base or Derived**: a base attribute is an original attribute of an entity, whereas a derived attribute is a computed value using one or more of the base values or other derived attributes[2].

The following section describes the details of each entity and its attributes. The key attribute is generally underlined for easy recognition.

**Opportunity**: This entity defines the type of opportunity (whether it is a practical opportunity or a financial opportunity). This is a weak entity in relation to the entity, Sponsor, because the elimination of the offering sponsor would automatically eliminate the opportunity. The attribute details of this entity type are shown in Table 4.2.1.

**Requirements**: This entity specifies the required qualifications for an opportunity, and thus holds all information related to an opportunity. The details of the attributes related to this entity type are shown in Table 4.2.2.
### Table 4.2.1: Attribute Details for the Opportunity Entity

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
<th>C or A</th>
<th>S or M</th>
<th>F or P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opportunity_Num</td>
<td>a system generated unique sequence #</td>
<td>A</td>
<td>S</td>
<td>F &amp; P</td>
</tr>
<tr>
<td>Type</td>
<td>Type of opportunity, e.g. grant, loan</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Number_of_Opps</td>
<td>number of opportunities available</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Salary/Amount</td>
<td>financial award details</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Add'l_Info</td>
<td>provides additional details</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
</tbody>
</table>

### Table 4.2.2: Attribute Details for the Requirements Entity

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
<th>C or A</th>
<th>S or M</th>
<th>F or P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>determines the educational eligibility for an opportunity, e.g. senior</td>
<td>A</td>
<td>M</td>
<td>F</td>
</tr>
<tr>
<td>Visa</td>
<td>determines visa eligibility, e.g. F1, J1</td>
<td>A</td>
<td>M</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Degree</td>
<td>determines educational eligibility, e.g. BS</td>
<td>A</td>
<td>M</td>
<td>P</td>
</tr>
<tr>
<td>Min_Score</td>
<td>Score requirements, e.g. GRE, GMAT</td>
<td>C</td>
<td>M</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Gender</td>
<td>Gender requirements for an opportunity</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Ethnicity</td>
<td>defines the ethnicity preferences</td>
<td>A</td>
<td>M</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Field of Study</td>
<td>determines the field of study in which the opportunity is offered, e.g. Nursing</td>
<td>A</td>
<td>M</td>
<td>F</td>
</tr>
<tr>
<td>Add'l_Requirements</td>
<td>Preferred requirements, not mandatory, e.g. BS required, MS preferred</td>
<td>A</td>
<td>M</td>
<td>F&amp;P</td>
</tr>
</tbody>
</table>

1: C= Composite; A= Atomic

2: S= Single valued; M= multi-valued

3: F= Applies to financial opportunity; P= Applies to practical opportunity
Application Recipient: This entity holds the details of who and where to apply. All attributes listed under this entity type apply to both opportunities. The details of the attributes related to this entity type are shown in Table 4.2.3.

Table 4.2.3: Attribute Details for the Application Recipient Entity

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
<th>C or A¹</th>
<th>S or M²</th>
<th>F or P³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>a system generated unique sequence number given for each application recipient</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Name</td>
<td>name of the agency that is accepting the applications for an opportunity</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Attention</td>
<td>name of the person/dept. to whom the application is to be directed</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Phone</td>
<td>application recipient’s phone number</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Fax</td>
<td>application recipient’s fax number</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>E-mail</td>
<td>application recipient e-mail address</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Web site</td>
<td>application recipient web site address, e.g. <a href="http://www.database.com">http://www.database.com</a></td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Address</td>
<td>application recipient mailing address details</td>
<td>C</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Deadline</td>
<td>this is an attribute stored under the relationship (is_applied_for_at). This is necessary to keep track of the individual opportunity versus the application recipient.</td>
<td>A</td>
<td>M</td>
<td>F&amp;P</td>
</tr>
</tbody>
</table>

¹: C= Composite; A= Atomic

²: S= Single valued; M= Multi-valued

³: F= Applies to financial opportunity; P= Applies to practical opportunity
**Sponsor:** This entity's attributes will hold the information about the sponsor offering a practical or a financial opportunity. Reflecting its importance in the financial opportunity itself, the sponsor entity 'supports' several weak entities in the sense that, if a sponsor is deleted from the database, its associated opportunities (and their requirements) must be deleted as well. The attribute details of this entity type are shown in Table 4.2.4.

**Table 4.2.4: Attribute Details for the Sponsor Entity**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
<th>C or A¹</th>
<th>S or M²</th>
<th>F or P³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sponsor_Name</td>
<td>name of the sponsor offering an opportunity</td>
<td>A</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Fund_Name</td>
<td>fund name, e.g. Rockefeller foundation</td>
<td>A</td>
<td>M</td>
<td>F&amp;P</td>
</tr>
<tr>
<td>Address</td>
<td>sponsor's address details, e.g. street name, suite #, City name, etc.</td>
<td>C</td>
<td>S</td>
<td>F&amp;P</td>
</tr>
</tbody>
</table>

**Country:** Inclusion of country as an entity will allow the user to search information by country. Information including the sponsor's headquarters is available under this entity type. This entity has only one attribute shown in Table 4.2.5.

**Table 4.2.5: Attribute Details for the Country Entity**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
<th>C or A¹</th>
<th>S or M²</th>
<th>F or P³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country_Name</td>
<td>name of the country where the opportunity is offered, e.g. a summer internship with Morgan Stanley, Inc. in England, etc.</td>
<td>A</td>
<td>M</td>
<td>F&amp;P</td>
</tr>
</tbody>
</table>

1: C= Composite; A= Atomic
2: S= Single valued; M= multi-valued
3: F= Applies to financial opportunity; P= Applies to practical opportunity
4.2.2 Identify Relationships

As described earlier, ISSODB has five entities. Each of these entities are associated among themselves by means of relationships. To set up a relationship between any two entities, one must first determine the nature of the relationship. There are three types of relationships, which are briefly described below.

- One-to-many relationship (1-M): a one-to-many relationship is the most common type of relationship in a relational database. In this type, a record in Table A (Entity A) can have more than one matching record in Table B (Entity B), but a record in Table B has at most one matching record in Table A[9].

- Many-to-many relationship (M-M): in this type of relationship, a record in Table A can have more than one matching record in Table B, and similarly a record in Table B can also have more than one matching record in Table A.

- One-to-one relationship (1-1): this is not a common relationship, in which a record in Table A can have no more than one matching record in Table B, and a record in Table B can have no more than one matching record in Table A[9].

The entities involved in a specific relationship are called the participants of the relationship, and the number of participants in that relationship defines the degree of the relationship. The participation level of an entity in a relationship can either be total or partial. It is said to be total participation when every instance of the a participating entity participates in at least one instance of the relationship, otherwise, the participation level is
termed as partial. The following section describes the different relationships that were derived in the ISSODB application.

**Requires:**

![Requires Diagram]

Requires is a 1-1 relationship between opportunity and requirements: This is a 1-1 relationship because every opportunity has one set of requirements and every participating set of requirements is required for one opportunity. Opportunity participation is full in this relationship, whereas requirement participation is partial (since all of the attributes do not participate in this relationship).

**Is Appliend For At:**

![Is Applied For At Diagram]

This is a relationship between opportunity and application_recipient. This is a M-1 relationship because each opportunity has only one application recipient, whereas an application recipient can receive many applications for one or more opportunities available with the recipient's sponsor. Both entities have full participation in the relationship. Application_Recipient is not weak with respect to opportunity because it is common to administer several opportunities in one office.

**Accepts Applications At:**

![Accepts Applications At Diagram]

This is a M-M relationship between sponsor and application_recipient. This is a M-M relationship as one office (such as a scholarship clearing house or an practical
agency) serves multiple sponsors and a single sponsor accepts applications at multiple offices (which might, for example, serve distinct regions of the country).

*Is_Funded_By:*

![Diagram for Is_Funded_By relationship]

*Is_Funded_By* is a M-1 relationship between opportunity and sponsor. This is a M-1 relationship because each opportunity is offered by only one sponsor, whereas a sponsor can offer several opportunities. Both entities have full participation in the relationship (assuming that every sponsor has at least one opportunity available). This is weak because the opportunity's existence depends on the existence of the sponsor.

*Having_Opportunity_In:*

![Diagram for Having_Opportunity_In relationship]

*Having_Opportunity_In* is a M-M relationship between sponsor and country because a sponsor can have one or more opportunities located in different countries, whereas a country can have one or more sponsors offering opportunities. Sponsor has full participation as every sponsor has at least one opportunity. Country has partial participation since every country may not have an opportunity.

All of the above information was compiled to form an E-R diagram as shown in Figure 4.2.1. The cardinality ratio and participation constraints were determined from the requirements collected.
Figure 4.3.1
International Student Services Opportunity Database (ISSODB)
Entity-Relationship (E-R) Diagram
4.3 DERIVATION OF FUNCTIONAL DEPENDENCIES

Once the E-R diagram is mapped with the database requirements and constraints, the next logical step is to derive functional dependencies (FDs). Functional dependencies are used to group attributes into relation schemas that are in normal form. A functional dependency is a constraint between two sets of attributes from a database[1].

For example, in a relation R; X and Y are arbitrary subsets of the set of attributes of R, then we can say that Y is functionally dependent on X. In symbolic form, this relationship is denoted as X → Y. The symbolic notation is read “X functionally determines Y” - if each X value has an association with precisely one Y value in relation R. In other words, whenever two tuples (records) of R agree on their X value, they also agree on their Y value[1].

Based on the review of the user requirements, and the E-R Diagram the following FDs have been established. These functional dependencies are grouped by each determinant attribute (e.g. Sponsor_Name, Opportunity_Num, etc.)

- Sponsor_Name ----> Street
- Sponsor_Name ----> City
- Sponsor_Name ----> State
- Sponsor_Name ----> Zip
- Sponsor_Name ----> Country
- Sponsor_Name ----> Country_Code
- Opportunity # ----> Type
- Opportunity # ----> Salary
- Opportunity # ----> Additional_Info
Opportunity # — Start Date
Opportunity # — Duration
Opportunity # — Gender
Opportunity # — Term
Opportunity # — Num_Of_Opp
Opportunity # — Country_Code
Opportunity # — Sponsor_Name
Opportunity # — Fund_Name

AR_Num, Sponsor_Name, Fund_Name — AR_Name
AR_Num, Sponsor_Name, Fund_Name — AR_Attention
AR_Num, Sponsor_Name, Fund_Name — AR_Title
AR_Num, Sponsor_Name, Fund_Name — AR_Phone
AR_Num, Sponsor_Name, Fund_Name — AR_Fax #
AR_Num, Sponsor_Name, Fund_Name — AR_Street
AR_Num, Sponsor_Name, Fund_Name — AR_City
AR_Num, Sponsor_Name, Fund_Name — AR_State
AR_Num, Sponsor_Name, Fund_Name — AR_Zip
AR_Num, Sponsor_Name, Fund_Name — AR_Country
AR_Num, Sponsor_Name, Fund_Name — AR_Gopher
AR_Num, Sponsor_Name, Fund_Name — AR_Web

Division_Name — Low
Division_Name — High

Sub_Category_Name — Low
Sub_Category_Name — High
This section defines the initial set of base relations that can be derived from the E-R diagram. A relation is just a mathematical term for a table of a specific kind. A relation’s primary components are tuples and fields. The tuple corresponds to a row, whereas the field corresponds to an attribute[1]. The number of tuples in a relation is called the cardinality and the number of attributes is called the degree of the relation. For each relation, there is also a primary key denoted with an underline, and sometimes a foreign key denoted within a dashed rectangle. The primary key is a unique identifier for
the table. It can be a single attribute or a combination of attributes with the property that, at any given time no two rows of the table can contain the same value for that attribute or the combination of the attributes. For ISSODB, the following base relations are derived:

**Sponsor**

<table>
<thead>
<tr>
<th>Sponsor_Name</th>
<th>Address</th>
<th>Country_Code</th>
</tr>
</thead>
</table>

**Fund_Sponsor**

<table>
<thead>
<tr>
<th>Fund_Name</th>
<th>Sponsor_Name</th>
</tr>
</thead>
</table>

**Opportunity**

<table>
<thead>
<tr>
<th>Opportunity#</th>
<th>Type</th>
<th>Salary</th>
<th>Additional_Info</th>
<th>Num_of_Opp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country_Code</th>
<th>AR_Num</th>
<th>Sponsor_Name</th>
<th>Fund_Name</th>
</tr>
</thead>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Degree</th>
<th>Class</th>
<th>Visa</th>
<th>Min_Score</th>
<th>Gender</th>
<th>Ethnicity</th>
<th>Field_of_Study</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Opportunity_Num</th>
<th>Additional_Requirements</th>
</tr>
</thead>
</table>

**Application_Recipient**

<table>
<thead>
<tr>
<th>AR_Num</th>
<th>AR_Name</th>
<th>AR_Attn</th>
<th>AR_Title</th>
<th>AR_Phone_Num</th>
<th>AR_Fax</th>
<th>AR_Address</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>AR_Email</th>
<th>AR_Gopher</th>
<th>AR_Web</th>
<th>Sponsor_Name</th>
<th>Fund_Name</th>
</tr>
</thead>
</table>
4.5 NORMALIZATION OF BASE RELATIONS

The next step is to normalize the base relations obtained in the previous section. Normalization of data is a process during which unsatisfactory relation schemas are decomposed by breaking up their attributes into smaller relations that possess desirable properties[2]. Normalization of a database is achieved through a series of steps which are grouped as First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF). Further there are Fourth Normal Form (4NF) and Fifth Normal Form (5NF) also. But, in general a database is desired to be at least in 3NF.

Initially some of the relations in ISSODB were not even in 1NF. For a relation to be in 1NF, it should not have any attributes that are composite or multi-valued. For example, the “Address” attribute under sponsor entity is composite and the “Field of Study” attribute of the requirements entity was multi-valued. In order to bring the relations to 1NF, all the composite attributes were decomposed to atomic attributes and the multi-valued attributes were broken down into separate relations. In the case of the ISSODB, once all the relations were brought to 1NF, they automatically fell into the 3NF. The Normalized relations are described in the following section.

<table>
<thead>
<tr>
<th>Country Code</th>
<th>Country Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethnicity Code</td>
<td>Ethnicity Code</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Sponsor</td>
<td>Sponsor Name</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Fund Sponsor</td>
<td>Fund Name</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Opportunity</td>
<td>Opportunity #</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Term</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethnicity</td>
<td>Ethnicity Code</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Recipient</td>
<td>AR Num</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AR City</td>
</tr>
<tr>
<td></td>
<td>AR Country</td>
</tr>
<tr>
<td></td>
<td>AR Gopher</td>
</tr>
<tr>
<td></td>
<td>AR Web</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Division_Range

<table>
<thead>
<tr>
<th>Division_Name</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
</table>

### Category_Range

<table>
<thead>
<tr>
<th>Sub_Category_Name</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
</table>

### Major_Codes

<table>
<thead>
<tr>
<th>Major_Code</th>
<th>Major_Name</th>
</tr>
</thead>
</table>

### Majors

<table>
<thead>
<tr>
<th>Major_Code</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>

### Visa

<table>
<thead>
<tr>
<th>Visa_Type</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>

### Class

<table>
<thead>
<tr>
<th>Standing</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>

### Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>Min_Score</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>

### Additional_Requirements

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>
### Deadline

<table>
<thead>
<tr>
<th>Deadline Date</th>
<th>Term Begins</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>

### Locality

<table>
<thead>
<tr>
<th>Locality Name</th>
<th>Country Code</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>

### Opportunity Degree

<table>
<thead>
<tr>
<th>Degree</th>
<th>Remark</th>
<th>Opportunity #</th>
</tr>
</thead>
</table>
5.0 PHYSICAL DESIGN

The physical design of the database is the physical implementation of the conceptual design output using an RDBMS like ORACLE 7. This phase defines the data storage structures, associated mappings, and access paths related to the database application. This process generally involves the design of tables, design of data entry forms, loading the data, generating reports, and testing for performance. The following sections describe the ISSODB physical design process in more detail.

5.1 CREATION OF TABLES

The tables for the database were created using ORACLE 7.0. ORACLE 7 was chosen for this application primarily due to its existence in CSUSB as well as other powerful capabilities such as: flexibility to expand the application into the future; enormous data processing capabilities; industry standard RDBMS; etc.

The normalized relations that were derived in the conceptual design phase were converted to a table format. The ISSODB application has 18 tables. Each table varied in size ranging from two fields to sixteen fields. A table is created in ORACLE using the command Create Table. The syntax for this command is as follows:

```
create table table_name (  
column1   datatype column_constraint | table_constraint  
column2   datatype  
.............  .............  
);  
```
For example, the sponsor table is created as follows:

```sql
create table sponsor (  
sponsor_name varchar(50) primary key,  
street varchar(20),  
city varchar(10),  
state varchar(10),  
zip varchar(20),  
country_code varchar(20)  
references country_data(country_code)  
);
```

The syntax used to create a table is not case sensitive. The `create table` command enforces different kinds of constraints on the table including primary keys, foreign keys, and check conditions. Enforcement of these constraints allow ORACLE to maintain the database integrity. Based on the type of the data, the data type is specified as `CHAR`, `NUMBER`, `VARCHAR`, etc. A majority of the fields in the ISSODB tables are defined as either `VARCHAR` or `NUMBER`. The advantage of using `VARCHAR` over `CHAR` is that `VARCHAR` uses only the required spaces, whereas `CHAR` utilizes all defined uses by padding the unfilled spaces (after using the required spaces) with blanks.

The order in which the tables are dropped and created is critical, especially because of the referential integrity constraints. Referential integrity constraint (a foreign key constraint) specifies that the values of the foreign key correspond to actual values of the primary key in the other table[3]. For example, in the fund_sponsor table, the `sponsor_name` field refers to the values of the `sponsor_name` field in the sponsor table. In
this case drop fund_sponsor table before dropping the sponsor table. The sequence of the
ISSODB tables and their field definitions are described below:

drop TABLE LOCALITY;
drop TABLE ETHNICITY;
drop TABLE DIVISION_RANGE;
drop TABLE CATEGORY_RANGE;
drop TABLE MAJORS;
drop TABLE MAJOR_DATA;
drop TABLE VISA;
drop TABLE CLASS;
drop TABLE TESTS;
drop TABLE ADDITIONAL_REQUIREMENTS;
drop TABLE DEADLINE;
drop TABLE OPPORTUNITY_DEGREE;
drop TABLE OPPORTUNITY;
drop TABLE APPLICATION_RECIPIENT;
drop TABLE ETHNICITY_DATA;
drop TABLE FUND_SPONSOR;
drop TABLE SPONSOR;
drop TABLE COUNTRY_DATA;

CREATE TABLE COUNTRY_DATA
(
  COUNTRY_CODE VARCHAR(20) PRIMARY KEY,
  COUNTRY_NAME VARCHAR(20)
);  

CREATE TABLE ETHNICITY_DATA
(
  ETHNICITY_CODE VARCHAR(20) PRIMARY KEY,
  ETHNICITY_NAME VARCHAR(20)
);
CREATE TABLE SPONSOR
(
 SPONSOR_NAME VARCHAR(50) PRIMARY KEY,
 STREET VARCHAR(20),
 CITY VARCHAR(10),
 STATE VARCHAR(10),
 ZIP VARCHAR(20),
 COUNTRY VARCHAR(20),
 COUNTRY_CODE VARCHAR(20)
)
REFERENCES COUNTRY_DATA(COUNTRY_CODE);

CREATE TABLE FUND_SPONSOR
(
 SPONSOR_NAME VARCHAR(50)
 REFERENCES SPONSOR(SPONSOR_NAME),
 FUND_NAME VARCHAR(50),
 PRIMARY KEY(SPONSOR_NAME, FUND_NAME)
);

CREATE TABLE APPLICATION.Recipient
(
 AR_NUM NUMBER(10),
 AR_NAME VARCHAR(70),
 AR_ATTN VARCHAR(30),
 AR_TITLE VARCHAR(30),
 AR_PHONE_NUM VARCHAR(20),
 AR_FAX VARCHAR(20),
 AR_STREET VARCHAR(40),
 AR_CITY VARCHAR(40),
 AR_STATE VARCHAR(20),
 AR_ZIP VARCHAR(30),
 AR_COUNTRY VARCHAR(20),
 AR_E_MAIL VARCHAR(40),
 AR_GOPHER VARCHAR(40),
 AR_WEB VARCHAR(40),
 SPONSOR_NAME VARCHAR(50),
)

CREATE TABLE OPPORTUNITY
(
  OPPORTUNITY_NUM NUMBER(10) PRIMARY KEY,
  TYPE VARCHAR(15),
  POSITION VARCHAR(30),
  SALARY VARCHAR(20),
  ADDITIONAL_INFO VARCHAR(50),
  START_DATE VARCHAR(20),
  DURATION VARCHAR(22),
  GENDER VARCHAR(5),
  TERM VARCHAR(10),
  NUM_OF_OPP VARCHAR(20),
  COUNTRY_CODE VARCHAR(20),
  REFERENCES COUNTRY_DATA(COUNTRY_CODE),
  SPONSOR_NAME VARCHAR(50),
  FUND_NAME VARCHAR(50),
  AR_PHONE_NUM VARCHAR(20),
  FOREIGN KEY(AR_PHONE_NUM, SPONSOR_NAME, FUND_NAME)
  REFERENCES APPLICATION_RECIPIENT(AR_PHONE_NUM, SPONSOR_NAME, FUND_NAME)
);

CREATE TABLE ETHNICITY
(
  ETHNICITY_CODE VARCHAR(15),
  REFERENCES ETHNICITY_DATA(ETHNICITY_CODE),
  OPPORTUNITY_NUM NUMBER(10),
  REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
  PRIMARY KEY(ETHNICITY_CODE, OPPORTUNITY_NUM)
);
CREATE TABLE DIVISION_RANGE
(
DIVISION_NAME VARCHAR(50) PRIMARY KEY,
DIVISION_LOW NUMBER(5),
DIVISION_HIGH NUMBER(5)
);

CREATE TABLE CATEGORY_RANGE
(
SUBCATEGORY_NAME VARCHAR(50) PRIMARY KEY,
SUBCATEGORY_LOW NUMBER(5),
SUBCATEGORY_HIGH NUMBER(5)
);

CREATE TABLE MAJOR_DATA
(
MAJOR_NAME VARCHAR(50),
MAJOR_CODE NUMBER(5) PRIMARY KEY
);

CREATE TABLE MAJORS
(
MAJOR_CODE NUMBER(5)
REFERENCES MAJOR_DATA(MAJOR_CODE),
OPPORTUNITY_NUM NUMBER(10)
REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
PRIMARY KEY(MAJOR_CODE, OPPORTUNITY_NUM)
);

CREATE TABLE VISA
(
VISA_TYPE VARCHAR(15),
OPPORTUNITY_NUM NUMBER(10)
REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
PRIMARY KEY(VISA_TYPE, OPPORTUNITY_NUM)
);

CREATE TABLE CLASS
(
    STANDING VARCHAR(60),
    OPPORTUNITY_NUM NUMBER(10)
    REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
    PRIMARY KEY(STANDING, OPPORTUNITY_NUM)
);

CREATE TABLE TESTS
(
    TEST VARCHAR(10),
    MIN_SCORE VARCHAR(10),
    OPPORTUNITY_NUM NUMBER(10)
    REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
    PRIMARY KEY(TEST, OPPORTUNITY_NUM)
);

CREATE TABLE ADDITIONAL_REQUIREMENTS
(
    REQUIREMENTS VARCHAR(50),
    OPPORTUNITY_NUM NUMBER(10)
    REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
    PRIMARY KEY(REQUIREMENTS, OPPORTUNITY_NUM)
);

CREATE TABLE DEADLINE
(
    DEADLINE_DATE CHAR(20),
    TERM_BEGINS CHAR(20),

OPPORTUNITY_NUM NUMBER(10)
REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
PRIMARY KEY(DEADLINE_DATE, TERM_BEGINS, OPPORTUNITY_NUM)
);

CREATE TABLE LOCALITY
(
    LOCALITY_NAME VARCHAR(20),
    COUNTRY_CODE VARCHAR(20)
    REFERENCES COUNTRY_DATA(COUNTRY_CODE),
    SPONSOR_NAME VARCHAR(50)
    REFERENCES SPONSOR(SPONSOR_NAME),
    PRIMARY KEY(LOCALITY_NAME, COUNTRY_CODE, SPONSOR_NAME)
);

CREATE TABLE OPPORTUNITY_DEGREE
(
    DEGREE VARCHAR(20),
    REMARK VARCHAR(250),
    OPPORTUNITY_NUM NUMBER(10)
    REFERENCES OPPORTUNITY(OPPORTUNITY_NUM),
    PRIMARY KEY(DEGREE, OPPORTUNITY_NUM)
);

5.2 DATA LOADING

Following the layout of tables, the next step was to populate these tables with relevant data. The ISSODB data was available in two formats: (i) manually documented tabular format stored in a traditional drawer-file management system and (ii) partial data stored in dBASE IV format. To reduce the data loading time, an attempt was made to import dBASE IV data into ORACLE format. This task presented two challenges, the first one was simply the technical problem of moving data without having to re-key, and
the second one was more of a trouble-some issue dealing with the re-categorization and
the re-arrangement of the data. One possible solution was to use dBASE's ASCII export
facility and a series of SQL INSERT commands. A dBASE program was written to
execute the export and INSERT functions, one for each record containing the data for
each field. In spite of the automated process, manual editing and re-keying was
necessary for each record for accurate positioning of data. Some specific problems
encountered during this data conversion process are highlighted below:

- In dBASE files, the attributes amount and duration of a financial award are
  stored in a single field. In ORACLE, these attributes are stored under separate
  fields. The only way to separate the imported data was to manually re-
  arrange, which would be a very time consuming effort.

- The application_recipient information in ORACLE was designed to be in a
  separate table, whereas the dBASE format was stored in a memo field, which
  is an unstructured text document. Because of this problem, all of this data
  would have to be entered manually.

- When manually editing the addresses of application_recipient, it was often
  impossible to determine the breakdown of the address information such as
  street, city, etc.

- Fund_name and funded_by attributes in dBASE were stored in switched order,
  whereas in ORACLE, they are designed to be in the correct order.
• Information on class standing in dBASE was duplicated in two fields. When this information was imported, adjustments were necessary to eliminate the redundancy.

Considering all of the above problems, and especially with the fact that the ISSODB dBASE data was only partial, the data conversion process from dBASE to ORACLE was found to be not efficient. At this point, data entry through SQL*Forms seemed to be an efficient and user-friendly approach.

5.3 DESIGN OF SQL*FORMS

Realizing that using SQL*Forms for data entry is a better approach, this task focused on the design of such forms. For definition, SQL*Forms aid in the quick development of form-based applications for entering, querying, updating, and deleting data. The application needs can be specified using simple menus, spread tables, and a powerful screen painter[5]. The non-procedural approach of SQL*Forms allow prototyping of applications very effectively. Prototyping allows refinement of an application as it is built. Prior to the detailed discussion that follows, it is critical that the following terms be well understood[5].

**Block**: a section of a form that usually corresponds to one table in the database.

Blocks contain a group of related fields that can spread across one or more pages.

**Base Table**: the database table on which a block is based.

**Record**: data from one row in a table, as represented in a form.

**Multi-Record Block**: a block that can display more than one-record at a time.
**Pop-up Window:** an area of a form that temporarily overlays another area by “popping up” in response to some event or user action. A pop-up window can be a pop-up page, a pop-up list of values, or the pop-up field editor.

**Field:** a highlighted or underlined area on a page that can display data and where operators input data. The data usually corresponds to data from a column in a database table.

For the ISSODB application, eight separate forms have been developed. These forms can be grouped under two different categories, namely the first and the second categories.

There are two first category forms in the ISSODB application, namely the CNTRY_ETH_CODE_ENTRY form and the MAJORS_DATA form. The first category forms relate to the tables that store general information such as the list of countries and their associated codes, list of fields of study and their associated codes, and ethnicity and their associated codes. For this project, a standardized set of codes for these variables were developed using information listed in the CSUSB’ application forms and GRE information bulletin. The ISSODB application also has six second category forms, namely the ADD’TNL_REQ_DEADLINE form, APPLN_RECEP_ENTRY form, LOCALITY_DEGREE form, OPPORTUNITY form, SPONSOR form, and VISA_CLASS_TSTS form. These second category forms correspond to the tables which are directly related to the information regarding an opportunity. Some of the fields (e.g.: country name, field of study, etc.) in the second category forms can be entered through a pop-up list of available choices. All eight forms are illustrated in APPENDIX B. For the
purpose of clarity, function key definitions are provided in Table 5.3.1. The following sections provide a more detailed discussion on the development of the ISSODB SQL*Forms.

5.3.1 Steps Used to Create Forms

The following steps were used to create the ISSODB forms:

- Start SQL*Forms by typing the following command:
  sqlform30 -C vt100u;utd <username>/<password>

- Use the left or right arrow keys to reach the Action option on the SQL*Forms main menu.

- Press [select], then the Action menu appears, with the New option highlighted.

- Press [select] to choose the New option. The New Form dialog box appears. At this instance, type OPPORTUNITY in Name field and press [Accept]. Following this action, SQL*Forms returns to the main menu.

- Press R for the Form menu and then press M to reach the Form Definition form. The name of the form OPPORTUNITY_ENTRY already appears in the title field.

- Press [Next Field] to reach the Validation unit field. Press [list] to display the list of values for this field. Use [Up] and [Down] keys to scroll up or down.

- Press [Select] to choose the value of the “field” into the Validation Unit field.
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Left]</td>
<td>^B</td>
</tr>
<tr>
<td>[Right]</td>
<td>^F</td>
</tr>
<tr>
<td>[Up]</td>
<td>^P</td>
</tr>
<tr>
<td>[Down]</td>
<td>^N</td>
</tr>
<tr>
<td>[Next Field]</td>
<td>TAB</td>
</tr>
<tr>
<td>[Next Block]</td>
<td>^XB</td>
</tr>
<tr>
<td>[Previous Block]</td>
<td>^Z^XB</td>
</tr>
<tr>
<td>[Next Record]</td>
<td>^XR</td>
</tr>
<tr>
<td>[Previous Record]</td>
<td>^Z^XR</td>
</tr>
<tr>
<td>[Insert/Replace]</td>
<td>ESC /</td>
</tr>
<tr>
<td>[Edit]</td>
<td>ESC E</td>
</tr>
<tr>
<td>[Enter Query]</td>
<td>^X3</td>
</tr>
<tr>
<td>[Execute Query]</td>
<td>^X4</td>
</tr>
<tr>
<td>[Commit/Accept]</td>
<td>^XA</td>
</tr>
<tr>
<td>[Insert Record]</td>
<td>ESC I</td>
</tr>
<tr>
<td>[Delete Record]</td>
<td>ESC D</td>
</tr>
<tr>
<td>[List]</td>
<td>^XL</td>
</tr>
<tr>
<td>[Help]</td>
<td>^XH</td>
</tr>
<tr>
<td>[Print]</td>
<td>ESC ESC P</td>
</tr>
<tr>
<td>[Accept]</td>
<td>^XA</td>
</tr>
<tr>
<td>[Change Display Type]</td>
<td>ESC T</td>
</tr>
</tbody>
</table>
• Press [Next Field] until the cursor is in the Comment region of the window. This region can be used to describe the purpose of the form or to provide any other relevant information.

• Press [Accept] to return to the main menu. This action will store changes temporarily in a working memory allocation. Forms must be saved to save the changes permanently.

5.3.2 Steps Used to Create Blocks

Blocks are the building units of each form. Each block corresponds to at most one database table called the base table[4]. The following steps were used to create the ISSODB blocks:

• Type B to select the Block option in the main menu. The Block menu appears.

• Select the Default option. The Default Block form appears.

• Type OPPORTUNITY in the Block Name field.

• Place the cursor in the Base Table field by using [Next Field].

• Press [List] to display the names of tables available.

• Use [Up] and [Down] to reach the OPPORTUNITY table and press [select]. At this instant, "OPPORTUNITY" appears in the Base Table field.

• Press [Accept]. SQL*Forms returns to the main menu.

Multiple blocks can also be created in the same form by following similar procedures as outlined above.
5.3.3 Steps Used to Test the Form

The form that was created in the previous section must be saved to store the form definitions permanently in the database. The form can be saved by selecting the Save option under the Action menu. Once the form is saved, it must be generated before executing or testing the form. Generating a form converts the form definitions into a file that can be run by SQL*Forms. The steps followed to generate and execute a form is listed below.

- Select the Generate option from the Action menu. The Generate Form dialog box will appear with the name of the current form in both the Name and the File fields.
- Press [Accept]. At this instinct the Generate Form dialog box disappears, and the form is generated.
- Select the Execute option from the Action menu. A dialog box appears with the name of the form. Press [Accept] to run the OPPORTUNITY form. The form that was just created appears.
- Press [Insert record] to enter data. Enter data in different fields and press [Commit/Accept].

In this form, OPPORTUNITY_NUM is a mandatory field since it is the primary key for the OPPORTUNITY table. Therefore, the cursor will not move until a value is entered in this field. The following section will discuss in detail the creation of an automatic sequence generator.
5.3.4 Steps to Modify the Form

This section provides steps to modify the OPPORTUNITY_ENTRY form. This task is composed of two actions. The first action is to generate automatic sequence numbers, which retrieves the data for certain fields automatically. The following steps are used to generate these automatic sequence numbers.

- Press F to select the field menu.
- Press the return key to display the Field Definition spread table.
- Press [Change Display Type] to move to the Field Definition form for the OPPORTUNITY_NUM field. Type the following text in the Default Value field:

  :sequence.opportunity_num.nextval

This statement instructs SQL*Forms to retrieve the next available sequence number generated by the database into the opportunity_num field.

To improve accuracy (e.g. consistency in spelling) and efficiency of the data entry task, pop-up lists can also be used. A pop-up list provides all available choices for a specific field. The following section will describe the steps involved in developing a pop-up window (e.g. country_data pop-up window).

- Press F to select the Field menu.
- Press the return key to display the Field Definition spread table.
- Move to the country-code field and press [Change Display Type] to move to the Field Definition form for the country_code field.
• Type the following SQL statement in the List of values SQL text section:

```
select country_code, country_name
into :country_code from country_data
```

• Return to the main menu, save, generate, and execute the form.

After implementing the above two tasks, the opportunity_num field is automatically inserted by the next sequence number. At this time, press [List] in the country_code field for a list of available choices. [Up] and [Down] keys are used to scroll among the choices, and the selection of an appropriate code will place the relevant value in the country_code field of the opportunity block. Similarly, pop-up lists for sponsor_name, fund_name, and ar_num are also created by using SQL statements in the list of values SQL text region for the sponsor_name field:

```
select sponsor_name, fund_name, ar_num
into :sponsor_name, :fund_name, :ar_num
from Application_Recipient.
```

This will ensure that the correct information is stored even in case where a sponsor has multiple application recipients.

The forms that are created through steps described above not only aid data entry but also aid data querying. Once the OPPORTUNITY form is generated and executed, the following steps are used to perform data queries.

• Press [Enter Query] to get into the enter query mode.
• Move to the field on which the query is to be performed. For example, to search grant opportunities, move to the field “type” and enter grant.
• Press [Execute Query]. This request will retrieve all records related to the grant opportunities.

Though queries can be performed using data entry forms, an even better approach of querying is through reports. Reports are well formatted and can be created using SQLPLUS. The following section focuses on this topic.

5.4 DESIGN OF REPORTS

A report is a summary of information that is well organized and formatted to suit the user’s specifications. Although forms and query datasheets can be printed, reports provide more flexibility in presenting data that is easy to understand. Some real world examples of reports are mailing labels, invoices, sales summaries, etc.

For ISSODB, several reports can be generated based on query criteria. These reports can be generated based on single variable or multiple variable requests. Some examples of such requests are:

• Search for opportunities based on field of study (single variable request)
• Search for opportunities based on opportunity type (single variable request)
• Search for opportunities based on country (single variable request)
• Search for opportunities based on country and opportunity type (multi-variable request).
ISSODB reports are comprised of several SQLPLUS commands. The most commonly used SQLPLUS commands used to generate reports are briefly described below [3].

- **Remark**: instructs SQLPLUS that the words to follow this command are to be treated as comments, not instructions.
- **Set headsep**: the heading separator identifies the single character that tells SQLPLUS to split a title on to two or more lines.
- **Title**: sets the top title for each page of the report.
- **Btitle**: sets the bottom title for each page of the report.
- **Column**: gives SQLPLUS a variety of instructions on the heading format and treatment of a column.
- **Start**: tells SQLPLUS to execute the instructions that are saved in a file.
- **Set pagesize**: sets the maximum number of lines per page.

The details of the SQLPLUS code written to generate various ISSODB reports are shown in APPENDIX C.

### 5.5 DESIGN OF SQL*MENUS

A SQL*Menu is a productivity tool that provides a single menu interface for executing multiple data-processing tools. A menu is a list of choices that a user can select from to specify their next action [7]. Menu items can call other menus, execute
commands or run programs. There are many advantages to using a menu interface, some of which are listed below[7].

- provides a very user-friendly environment, thereby reducing the amount of technical knowledge needed by the user.
- presents a wide range of available choices
- improves data entry accuracy by minimizing typing errors (spelling errors or incorrect commands)
- makes the application more structured, thereby making the application easier to learn, use, and maintain.
- improves the application security.

The ISSODB main menu will have Data Entry, Reports, and Exit as the three available choices. When the Data Entry option is selected, all available forms are listed. Selection of a form through this listing eliminates the need for steps such as opening, generating, and executing a form. Similarly, selection of the Report option provides a list of different types of reports that can be generated from the ISSODB. The Exit option will allow the user to quit the application. The following section discusses the steps that were followed to create the ISSODB menu application.

- **SQL*Menu** is invoked by typing the following command:

  sqlmenu50 -C vt100u:utd <username>/<password>
Creating a Menu Application

• From the Action menu select New item. The New Menu Application dialog box appears.
• Enter ISSODB in the Name field.
• Press [Accept] to create the new application

Defining a Menu

• Select Menu item from the Menu submenu.
• Press [Change display type] to display the form. Menu definition form appears. Fill the menu definition information.

Defining a Menu Item

• Select the Menu Item from the Menu submenu.
• Press [Change display type] to display the Item Definition form.
• Enter 1 in the command type field to invoke a submenu (the submenu must already be defined using the steps described above)
• Enter descriptive names in the Item Text and Short Item Name fields.
• Enter the submenu name (reports) in the command line field.
• Press [Accept]
• Save, generate, and execute the ISSODB menu.

The other command types used in the ISSODB application are command type 2, which is used to execute an operating system command, and command type 7 is used to execute a PL/SQL command. Examples of these command types are presented below:
Operating system command that invokes the OPPORTUNITY_ENTRY form is:
runform30 -C vt100u:utd OPPORTUNITY_ENTRY <username>/</password>

Operating system command that invokes the FIELD OF STUDY report is:
sqplus -S <username>/</password> @fld_sdy

PL/SQL command that invokes the exit function is:
exit_menu;
6.0 DATABASE VALIDATION

Once the database is successfully designed and implemented, the next critical step is to validate the database application for its functionality. Positive results in these validation tests would imply that the database would perform satisfactorily, and its initial objectives have been met. Tests performed on the ISSODB application can be grouped under three primary categories: unit testing, integration testing, and system testing. Test results under each of these categories is presented in the following sections.

6.1 UNIT TESTING

Unit testing focuses on the verification effort of the smallest unit of the database application[10]. For ISSODB application, each form was tested for its functionality. Some examples of these functionalities are: correct generation of sequence numbers, proper storage and retrieval of data, and returning valid error messages, etc. Similarly, each report is also tested for data organization, format, and accurate retrieval of the user requested query. The results of the tests are summarized in Table 6.1.1

6.2 INTEGRATION TESTING

Following the unit testing phase, the next phase of testing would focus on the performance of the integrated components of the database application. The individual components like forms and reports are integrated using a SQL*Menu, an ORACLE based tool. All ISSODB forms are grouped under the Data Entry category, whereas all reports are categorized under the Reports category. Various tests were performed to verify the
### Table 6.1.1: Unit Test Results

<table>
<thead>
<tr>
<th>UNIT TESTED</th>
<th>TESTS PERFORMED</th>
<th>RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FORMS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDTNL_REQ_DLINE</td>
<td>• tested for proper storage and retrieval of data</td>
<td>Pass</td>
</tr>
<tr>
<td>APPLN_RECEP</td>
<td>• sequence # generation for the AR_Num field</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• data entered using pop-up list of Sponsor_Name and Fund_Name is tested for consistency</td>
<td></td>
</tr>
<tr>
<td>CNTRY_ETH_CODE</td>
<td>• Country_Code and Ethnicity_Code fields were verified for valid data ranges</td>
<td>Pass</td>
</tr>
<tr>
<td>LOCALITY_DEGREE</td>
<td>• tested for proper storage and retrieval of data</td>
<td>Pass</td>
</tr>
<tr>
<td>MAJORS_DATA</td>
<td>• Major_Code field was verified for valid data ranges</td>
<td>Pass</td>
</tr>
<tr>
<td>OPPORTUNITY</td>
<td>• sequence number generation for the Opportunity_Num field</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• data entered through pop-up lists of Country_Code, Sponsor_Name, Fund_Name, and AR_Num are verified for consistency</td>
<td></td>
</tr>
<tr>
<td>SPONSOR</td>
<td>• data entered through pop-up list of Country_Code is verified for consistency</td>
<td>Pass</td>
</tr>
<tr>
<td>VISA_CLASS_TSTS</td>
<td>• tested for proper storage and retrieval of data</td>
<td>Pass</td>
</tr>
<tr>
<td><strong>REPORTS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLD_STDY</td>
<td>• user is prompted to enter a field of study, after which the generated report must list all the opportunities available for that field of study</td>
<td>Pass</td>
</tr>
<tr>
<td>OPPY_TYPE</td>
<td>• user is prompted to enter an opportunity type, after which the generated report must list all the opportunities for the requested opportunity type</td>
<td>Pass</td>
</tr>
<tr>
<td>OPPY_CNTRY_TYPE</td>
<td>• user is prompted to enter both country name and an opportunity type, after which the generated report must list all opportunities for the requested type in the requested country</td>
<td>Pass</td>
</tr>
</tbody>
</table>
linkage between all forms and the ISSODB menu. Similarly, tests were also performed to verify the linkage between all reports and the ISSODB menu. Tests results are summarized in Table 6.2.1.

**Table 6.2.1: Integration Test Results**

<table>
<thead>
<tr>
<th>COMPONENTS INTEGRATED</th>
<th>TESTS PERFORMED</th>
<th>RESULTS</th>
</tr>
</thead>
</table>
| ISSODB Menu and Forms | • tested to verify if data entry option lists all the available forms  
• tested for proper linkage between all forms and the ISSODB menu | Pass |
| ISSODB Menu and Reports | • tested to verify if reports option lists all the available reports  
• tested for proper linkage between all reports and the ISSODB menu | Pass |

**6.3 SYSTEM TESTING**

In this phase of testing, the ISSODB application is tested as a complete system.

Two primary tests were performed to ensure the completeness of the system, which include tests on database user-friendliness and communication.

- **User-friendliness**: tests were performed to verify whether the application prompts the user for information in an accurate and easily interpretable format. Also, the application was tested to verify if useful help text was provided when necessary.

- **Communication**: tests were performed to check if the application would execute effectively when accessed through modem (remote access).
7.0 USER INSTRUCTIONS

Steps to invoke ISSODB application

- Start SQL*Menu by executing the command `sqlmenu50 -C vt100u:utd <username>/password`.

- ISSODB application can be invoked by opening, generating and executing the ISSODB menu. The main menu would appear as shown in Figure 7.1

Figure 7.1: Main Menu for ISSODB Application
Steps for Data Entry

- Choose the Data Entry option from the ISSODB main menu. All available forms under this option would appear as shown in Figure 7.2.
- Use [Up] and [Down] keys to select the appropriate form.
- Enter the data and press [Insert Record].
- Press [Commit/Accept] to commit the changes.
- Press [Exit] to return to the ISSODB main menu.

Figure 7.2: Available Forms Under the Data Entry Option
Data retrieval using Forms

The forms that were created for data entry can also be used for querying. The following steps are used to perform data queries using forms.

- Press [Enter Query] to get into the enter query mode.
- Move to the field on which the query is to be performed. For example, to search grant opportunities, move to the field “type” and enter grant.
- Press [Execute Query]. This request will retrieve all records related to the grant opportunities.

Data retrieval using Reports

- Choose the Reports option from the ISSODB main menu. A listing of available reports under this option would appear as shown in Figure 7.3.
- Use [Up] and [Down] keys to reach the appropriate report.
- Select the report to be executed.
- When prompted for input, type the necessary information.
- A report will be generated for the query performed.
- Type exit to return to the ISSODB main menu.

Exit out of the ISSODB application

- Choose the Exit option from the ISSODB main menu.
- Press return to exit out of the ISSODB application.
Figure 7.3: List of Available Reports
APPENDIX A

(SAMPLE DATA SHEET)
<table>
<thead>
<tr>
<th>PROGRAM'S NAME</th>
<th>COLLEGE SEMESTER ABROAD IN BOTSWANA</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFERED BY</td>
<td>BOTSWANA ORIENTATION CENTRE</td>
</tr>
<tr>
<td>FIELD(S) OF STUDY</td>
<td>ECONOMICS, ENVIRONMENTAL STUDIES, HISTORY, LANGUAGES, POLITICAL STUDIES, RURAL DEVELOPMENT</td>
</tr>
<tr>
<td>NATIONALITY</td>
<td>BOTSWANA</td>
</tr>
<tr>
<td>UNDERGRADUATE/POST GRADUATE</td>
<td>SOPHOMORES, JUNIORS, SENIORS, GRADUATE STUDENTS</td>
</tr>
<tr>
<td>GENDER</td>
<td>ALL</td>
</tr>
<tr>
<td>COST OF PROGRAM</td>
<td>One semester: $9900 for tuition, housing, all meals, insurance, excursions, international airfare, lab equipment, orientation</td>
</tr>
<tr>
<td>AMOUNT AND DURATION OF AWARD/Program</td>
<td>Sep 2 - Dec 13 (Fall)</td>
</tr>
<tr>
<td></td>
<td>Feb 2 - May 17 (spring)</td>
</tr>
<tr>
<td>ANNUAL NUMBER OF TERMS</td>
<td>2</td>
</tr>
<tr>
<td>REQUIREMENTS</td>
<td>2.5 GPA</td>
</tr>
<tr>
<td>WHERE TO APPLY</td>
<td>Admissions Representative, School for International Training, World Learning, P.O. Box DSOPG, Brattleboro, Vermont 05302</td>
</tr>
<tr>
<td>HOW TO APPLY</td>
<td>Tel: 802-336-1116, Fax: 802-258-3520</td>
</tr>
<tr>
<td>APPLICATION DEADLINE</td>
<td></td>
</tr>
<tr>
<td>ADDITIONAL INFORMATION</td>
<td>$400 refundable deposit required, Financial aid and available for all students. Scholarships/grants, 16 participants.</td>
</tr>
</tbody>
</table>
APPENDIX B

(ISSO DB DATA ENTRY FORMS)
Data Entry Form For
Additional Requirements, Deadline, and Majors Tables

<table>
<thead>
<tr>
<th>REQUIREMENTS</th>
<th>OPPORTUNITY_NUM</th>
<th>DEADLINE_DATE</th>
<th>TERM_BEGINS</th>
<th>MAJOR_CODE</th>
<th>OPPORTUNITY_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Count: *0*  <Replace>
## Data Entry Form For Application Recipient Table

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICANT_RECIPIENT</td>
<td></td>
</tr>
<tr>
<td>ATTENTION</td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td></td>
</tr>
<tr>
<td>PHONE NUMBER</td>
<td></td>
</tr>
<tr>
<td>STREET</td>
<td></td>
</tr>
<tr>
<td>CITY</td>
<td></td>
</tr>
<tr>
<td>STATE</td>
<td></td>
</tr>
<tr>
<td>ZIP</td>
<td></td>
</tr>
<tr>
<td>COUNTRY</td>
<td></td>
</tr>
</tbody>
</table>

**Count:** 0

(Replace)
Data Entry Form For
Country Data and Ethnicity Data Tables

COUNTRY_DATA
COUNTRY_CODE
COUNTRY_NAME

ETHNICITY_DATA
ETHNICITY_CODE
ETHNICITY_NAME

Count: 0
(Replace)
Data Entry Form For
Locality, Opportunity Degree, and Ethnicity Tables

<table>
<thead>
<tr>
<th>LOCALITY_NAME</th>
<th>COUNTRY_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPONSOR_NAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OPPORTUNITY_DEGREE</td>
</tr>
<tr>
<td>DEGREE</td>
<td></td>
</tr>
<tr>
<td>REMARK</td>
<td></td>
</tr>
<tr>
<td>OPPORTUNITY_NUM</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ETHNICITY_CODE</th>
<th>OPPORTUNITY_NUM</th>
</tr>
</thead>
</table>

Count: 90  <Replace>
### Data Entry Form For
**Division Range, Category Range, and Major Data Tables**

<table>
<thead>
<tr>
<th>DIVISION_NAME</th>
<th>DIVISION_LOW</th>
<th>DIVISION_HIGH</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SUBCATEGORY_NAME</th>
<th>SUBCATEGORY_LOW</th>
<th>SUBCATEGORY_HIGH</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MAJOR_NAME</th>
<th>MAJOR_CODE</th>
</tr>
</thead>
</table>

---

Count: 0  <Replace>
Data Entry Form For Opportunity Table

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPPORTUNITY_NUM</td>
<td>Opportunity number</td>
</tr>
<tr>
<td>SALARY</td>
<td>Salary</td>
</tr>
<tr>
<td>ADDITIONAL_INFO</td>
<td>Additional information</td>
</tr>
<tr>
<td>START_DATE</td>
<td>Start date</td>
</tr>
<tr>
<td>DURATION</td>
<td>Duration</td>
</tr>
<tr>
<td>GENDER</td>
<td>Gender</td>
</tr>
<tr>
<td>TERM</td>
<td>Term</td>
</tr>
<tr>
<td>NUM_OF_OPP</td>
<td>Number of opportunities</td>
</tr>
<tr>
<td>COUNTRY_CODE</td>
<td>Country code</td>
</tr>
<tr>
<td>SPONSOR_NAME</td>
<td>Sponsor name</td>
</tr>
<tr>
<td>FUND_NAME</td>
<td>Fund name</td>
</tr>
<tr>
<td>AR_NUM</td>
<td>AR number</td>
</tr>
</tbody>
</table>

Count: *0*

<Replace>
Data Entry Form For
Sponsor and Fund Sponsor Tables

<table>
<thead>
<tr>
<th>SPONSOR_NAME</th>
<th>STREET</th>
<th>COUNTRY</th>
<th>COUNTRY_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUNDSponsor_NAME</th>
<th>FUND_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Count: 0

<Replace>
Data Entry Form For
Visa, Class, and Tests Tables

<table>
<thead>
<tr>
<th>VISA_TYPE</th>
<th>OPPORTUNITY_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDING</td>
<td>OPPORTUNITY_NUM</td>
</tr>
<tr>
<td>TEST</td>
<td>MIN_SCORE</td>
</tr>
</tbody>
</table>

Count: 0
<Replace>
APPENDIX C

(SQLPLUS CODE FOR REPORTS AND SAMPLE OUTPUT)
List Opportunities By Country

set echo off
REM Author: Sailaja Malireddy
REM Filename: oppy-by-cntry.sql
REM Masters Project: ISSODB
REM DESCRIPTION: contains formatting & select commands to list opportunities by country

set linesize 90
set pagesize 66
set headsep |
title 'CSUSB - International Student Services|Opportunity Database'

accept mCntry char prompt 'Enter country name (or enter for "all"): ' 
set echo off
set verify off

column opportunity_num heading 'Number' format 9999
column type heading 'Category' format a12 word_wrapped
column country_name heading 'Country_Name' format a12 word_wrapped
column sponsor_name heading 'Sponsor' format a18 word_wrapped
column fund_name heading 'Fund' format a18 word_wrapped
column num_of_opp heading 'Number of|Openings' format a9 word_wrapped

select opportunity_num,  
type,  
country_name,  
sponsor_name,  
fund_name,  
um_of_opp  
from opportunity, country_data  
where opportunity.country_code = country_data.country_code  
and upper(country_name) like upper('%&mCntry%');
List Opportunities By Major

set echo off

REM Author: Sailaja Malireddy
REM Filename: oppy-by-field.sql
REM Masters Project: ISSODB
REM DESCRIPTION: contains formatting & select commands
to list opportunities by major

set linesize 90
set pagesize 66
set headsep

title 'CSUSB - International Student Services|Opportunity Database'

accept mMajor char prompt 'Enter major name (or enter for "all"): '

set echo off
set verify off

column opportunity_num heading 'Number' format 9999
column major_name heading 'Major' format a18 word_wrapped
column type heading 'Category' format a12 word_wrapped
column sponsor_name heading 'Sponsor' format a18 word_wrapped
column fund_name heading 'Fund' format a18 word_wrapped
column num_of_opp heading 'Number of Openings' format a9 word_wrapped

select opportunity.opportunity_num,
    major_name,
    type,
    sponsor_name,
    fund_name,
    num_of_opp
from opportunity, major_data, majors
where opportunity.opportunity_num = majors.opportunity_num
and major_data.major_code = majors.major_code
and upper(major_name) like upper('%&mMajor%');
List Opportunities By Type of Opportunity

set echo off

REM Author: Sailaja Malireddy
REM Filename: oppy-by-type.sql
REM Masters Project: ISSODB
REM DESCRIPTION: contains formatting & select commands to list opportunities by type

set linesize 80
set pagesize 66
set headsep |
title 'CSUSB - International Student Services|Opportunity Database'

accept mType char prompt 'Enter opportunity type (or enter for "all"): '

set echo off
set verify off

column opportunity_num heading 'Number' format 9999
column type heading 'Category' format a12 word_wrapped
column sponsor_name heading 'Sponsor' format a18 word_wrapped
column fund_name heading 'Fund' format a18 word_wrapped
column num_of_opp heading 'Number of Openings' format a9 word_wrapped

select opportunity_num,
    type,
    sponsor_name,
    fund_name,
    num_of_opp
from opportunity where type like upper('%&mType%');
List Opportunities By Both Opportunity Type and Country

set echo off
REM Author: Sailaja Malireddy
REM Filename: oppy-by-cntry-type.sql
REM Masters Project: ISSO.DB
REM DESCRIPTION: contains formatting & select commands to list opportunities by country and type
REM
set linesize 80
set pagesize 66
set headsep

title 'CSUSB - International Student Services|Opportunity Database'

accept mType char prompt 'Enter opportunity type (or enter for "all")': '
accept mCntry char prompt 'Enter country name (or enter for "all")': ' 

set echo off
set verify off

column opportunity_num heading 'Number' format 9999
column type heading 'Category' format a12 word_wrapped
column sponsor_name heading 'Sponsor' format a16 word_wrapped
column fund_name heading 'Fund' format a18 word_wrapped
column country_name heading 'Country' format a10 word_wrapped
column num_of_opp heading 'Number of Openings' format a9 word_wrapped

select opportunity.opportunity_num,
     type,
     sponsor_name,
     fund_name,
     country_name,
     num_of_opp
from opportunity, country_data
where type like upper('%&mType%')
and opportunity.country_code = country_data.country_code
and upper(country_name) like upper('%&mCntry%');
List Opportunities By Both Country And Major

set echo off

REM Author: Sailaja Malireddy
REM Filename: oppy-by-fld-entry.sql
REM Masters Project: ISSODB
REM DESCRIPTION: contains formatting & select commands
to list opportunities by country and major

set linesize 100
set pagesize 66
set headsep |
title 'CSUSB - International Student Services|Opportunity Database'

accept mMajor char prompt 'Enter major name (or enter for "all"): '
accept mCntry char prompt 'Enter country name (or enter for "all"): '

set echo off
set verify off

column opportunity_num heading 'Number' format 9999
column major_name heading 'Major' format a18 word_wrapped
column type heading 'Category' format a12 word_wrapped
column sponsor_name heading 'Sponsor' format a16 word_wrapped
column fund_name heading 'Fund' format a18 word_wrapped
column country_name heading 'Country' format a10 word_wrapped
column num_of_opp heading 'Number of Openings' format a9 word_wrapped

select opportunity.opportunity_num,
       major_name,
       type,
       sponsor_name,
       fund_name,
       country_name,
       num_of_opp
from opportunity, major_data, majors, country_data
where opportunity.opportunity_num = majors.opportunity_num
and major_data.major_code = majors.major_code
and upper(major_name) like upper('%&mMajor%')
and opportunity.country_code = country_data.country_code
and upper(country_name) like upper('%&mCntry%');
## SAMPLE REPORT 1
(Query by Country and Type)

<table>
<thead>
<tr>
<th>Number</th>
<th>Category</th>
<th>Sponsor</th>
<th>Fund</th>
<th>Country</th>
<th>Number of Openings</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>GRANT</td>
<td>DAAD</td>
<td>DAAD (GERMAN ACADEMIC EXCHANGE SERVICE)</td>
<td>GERMANY</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>GRANT</td>
<td>DAAD</td>
<td>DAAD (GERMAN ACADEMIC EXCHANGE SERVICE)</td>
<td>GERMANY</td>
<td>N/A</td>
</tr>
</tbody>
</table>

2 rows selected.
### SAMPLE REPORT 2
*(Query by Opportunity Type)*

Wed Mar 12

CSUSB - International Student Services
Opportunity Database

<table>
<thead>
<tr>
<th>Number</th>
<th>Category</th>
<th>Sponsor</th>
<th>Fund</th>
<th>Country</th>
<th>Number of Openings</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>GRANT</td>
<td>DAAD</td>
<td>DAAD (GERMAN ACADEMIC EXCHANGE SERVICE)</td>
<td>GERMANY</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>GRANT</td>
<td>DAAD</td>
<td>DAAD (GERMAN ACADEMIC EXCHANGE SERVICE)</td>
<td>GERMANY</td>
<td>N/A</td>
</tr>
<tr>
<td>6</td>
<td>GRANT</td>
<td>TRAVEL GRANT AWARD</td>
<td>AUSTRALIAN ACADEMY OF THE HUMANITIES</td>
<td>AUSTRALIA</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>GRANT</td>
<td>EAST-WEST CENTER DEGREE STUDENT AWARD</td>
<td>FOUNDATION FOR SCHOLARLY EXCHANGE</td>
<td>TAIWAN</td>
<td>5</td>
</tr>
</tbody>
</table>

4 rows selected.
**SAMPLE REPORT 3**  
*(Query by Country)*

Wed Mar 12

CSUSB - International Student Services  
Opportunity Database

<table>
<thead>
<tr>
<th>Number</th>
<th>Category</th>
<th>Sponsor</th>
<th>Fund</th>
<th>Country</th>
<th>Number of Openings</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>FELLOWSHIP</td>
<td>A.W. HOWARD MEMORIAL FELLOWSHIP</td>
<td>A.W. HOWARD MEMORIAL TRUST</td>
<td>AUSTRALIA</td>
<td>5-8</td>
</tr>
<tr>
<td>6</td>
<td>GRANT</td>
<td>TRAVEL GRANT AWARD</td>
<td>AUSTRALIAN ACADEMY OF THE HUMANITIES</td>
<td>AUSTRALIA</td>
<td>2</td>
</tr>
</tbody>
</table>

2 rows selected.
APPENDIX D

(LIST OF ACRONYMS)
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSUSB</td>
<td>California State University, San Bernardino</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>DEO</td>
<td>Data Entry Operator</td>
</tr>
<tr>
<td>ER</td>
<td>Entity Relationship</td>
</tr>
<tr>
<td>ISS</td>
<td>International Student Services</td>
</tr>
<tr>
<td>ISSODB</td>
<td>International Student Services Opportunity Database</td>
</tr>
<tr>
<td>PL</td>
<td>Pick List</td>
</tr>
<tr>
<td>PUM</td>
<td>Pop-Up Menu</td>
</tr>
</tbody>
</table>
REFERENCES


