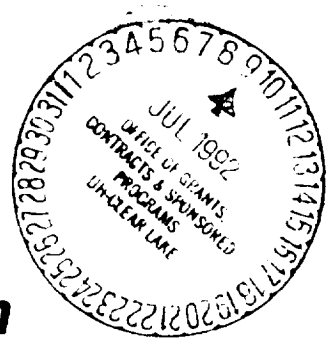

NASA Electronic Library System (NELS)

Database Schema

Version 1.2

Revision 1
July 1992

NASA
Lyndon B. Johnson Space Center
Information Systems Directorate
Software Technology Division

RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Clovis J. Melebeck of I-NET, Inc. Dr. E. T. Dickerson served as RICIS research coordinator.

Funding was provided by the Information Technology Division, Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA research coordinator for this activity was Ernest M. Fridge III, Deputy Chief of the Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

NASA Electronic Library System (NELS)

Database Schema

Prepared By: Clovis J. Melebeck
I-NET, Inc

Approved: Chris Culbert

Chris Culbert, Chief
Software Technology Branch

Version 1.2

Revision 1
July 1992

Information Technology Division

NASA
National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Table of Contents

1.0 NELS Version 1.2 Database Schema

1.1 Schema Introduction

The following section discusses the database tables used by NELS version 1.2. To provide the current functional capability offered by NELS, nineteen (19) tables have been created with ORACLE. Each table lists the ORACLE table name and provides a brief description of the tables intended use or function. The following sections cover four basic categories of tables; NELS object classes, NELS collections, NELS objects, and NELS supplemental tables. Also included in each section is a definition and/or relationship of each field to other fields or tables. The primary key(s) for each table is indicated with a single asterisk (*), while foreign keys are indicated with double asterisks (**). The primary key(s) indicate the key(s) which uniquely identifies a record for that table. The foreign key(s) is used to identify additional information in other table(s) for that record.

The two appendices are the command which are used to construct the ORACLE tables for NELS. Appendix A contains the commands which create the tables which are defined in the following sections. Appendix B contains the commands which build the indices for these tables.

1.2 NELS Object Classes

The tables listed in Table 1.2-1 NELS Object Classes, provide the necessary information to define object classes. These object classes are used by librarians to define the objects that are entered into the library. The following sections are an explanation of each table and their associated fields which are used to define these object classes.

Table Name	Description
ObjectClasses	Provides a list of all the known object classes that are defined in the library.
ClassFields	A subordinate table to ObjectClasses. It contains information about the attributes which describe the object.
Enum_Types	A subordinate table to ClassFields. It provides a list of all the known enumeration types.
Enum_Values	A subordinate table to Enum_Types. It contains the enumerated values for each enumeration type.

Table 1.2-1 NELS Object Classes

The *'ObjectClasses'* table is the highest level table in defining the object classes for objects. By using this table, one can construct a data structure which provides the details about each attribute used to define the metadata about an object.

1.2.1 ObjectClasses Table

The first table listed in Table 1.2-1, contains a list of every object class that will or has been created by a librarian. Normally, the Head Librarian will create all the object classes that will be used for the NELS library. The object class table consist of three fields. These fields are shown in Table 1.2-2 ObjectClasses Fields.

Field Name	Format	Length
ClassID	Character	22
ClassName	Character	80
NumFields	Number	38

Table 1.2-2 ObjectClasses Fields

The *'ClassID'* field is the key field used to access the attribute fields in the classfields table. This value is entered into the Class_ID field of objects to indicate what class object NELS is processing. Inheritance from base class objects is determined by this value. For example, the object class having a ClassID of '1002004' would inherit field attributes from the '1' Generic object class, the '1002' object class, and the '1002004' object class.

The *'ClassName'* field is a character string which represents the name of this object class.

The *'NumFields'* field contains the number of field attributes associated with this object class. This is a total of all fields, including those field attributes that are inherited from the base object class.

1.2.2 ClassFields Table

Table 1.2-3 shows the fields that make up an object class. These fields describe the information necessary for NELS to accept inputs or display output for the field attribute associated with this object class.

Field Name	Format	Length
ClassID	Character	22
Name	Character	50
Prompt	Character	50
Kind	Number	38
Visibility	Number	38
Editable	Number	38
Length	Number	38
NumColumns	Number	38
NumLines	Number	38
FieldUnique	Character	1
SeqNum	Number	38
Enum_ID	Number	

Table 1.2-3 ClassFields Fields

The *'ClassID'* field , along with the name, is the primary key used to access the attribute fields. All records containing the same classid are associated with the same object class. Multiple ClassID values, as described above are concatenated to produce the complete object class structure. For example, the object class having a ClassID of '1002004' would concatenate the field attributes from the classid '1', '1002', and '1002004'.

The *'Name'* field contains field name in the objects table that is used to store values for this attribute. For example, the 'title' field in the objects table would contain the title of the object.

The *'Prompt'* field is the prompt that will be displayed for this attribute. This prompt will be displayed any time the metadata about an object is displayed or entered.

The *'Kind'* field is an integer value which equates to the data type of this attribute. The following values are used for this field to indicate the data format type selected.

Value	Data Type
0	Character
1	Enumeration
2	List
3	One to One
4	One to Many
5	Integer
6	Float
7	Date
8	Long

The *'Visibility'* field is an integer value which represents whether or not this attribute's value is visible to the user. The following values indicate visibility.

Value	Visible
0	No
1	Yes

The *'Editable'* field is an integer value which determines whether or not this attribute can be edited or modified. However, when the value or values associated with this field are for display purposes only, this field is set to 'No' by the program.

Value	Editable
0	No
1	Yes

The *'Length'* field is an integer value which represents the maximum number of character for this attribute. Certain data types have preset values and this field can not be set by the librarian. The length for each of the data types is defined below. Only the character and long fields allow the librarian to enter a length.

Data Type	Length
Character	1 - 80
Enumeration	40
List	50
One to One	150
One to Many	150
Integer	10
Float	10
Date	10
Long	1 - 240

The *'NumColumns'* field is an integer value which represents the number of columns that are to be visually displayed. This value should be less than or equal to the 'length' field value.

The *'NumLines'* field is an integer value which represents the number of lines that are to be displayed for either character, long or list type fields.

The *'FieldUnique'* field is a character field which determines the input requirements for this attribute. The value indicates whether this attribute is Required (R), Not required (N), or Unique and required (U). If the field is required, then the user must enter a value for this attribute. If a 'U' is found in this field, the user must enter a value and that value must not exist in the library for this attribute.

The *'SeqNum'* field is an integer value which represent the order in which the field attributes are to be displayed.

The *'Enum_ID'* field is an integer value which identifies the enumeration type associated with this attribute. This field is only valid when the 'kind' field is set to a value of 1 for enumeration data types. This value is the key field into the 'enum_types' and 'Enum_Values' tables which provide information about the enumeration type attributes.

1.2.3 Enum_Types Table

The *'Enum_Types'* table contains a list of all the enumeration types currently defined. Any of these enumeration types can be used to define attributes associated with any object class. Table 1.2-4 Enum_types Fields lists the fields for this table.

Field Name	Format	Length
Enum_ID	Number	
Enum_Type*	Character	40

Table 1.2-4 Enum_Types Fields

The *'Enum_ID'* field is a unique integer value which identifies the enumeration type. This field is the key used to obtain the enumeration values for this enumeration type.

The *'Enum_Type'* field is a character field used to identify this enumeration type. For example, 'colors' would be the name entered for the enumeration type containing the colors (red, orange, yellow, green, blue, indigo, violet).

1.2.4 Enum_Values Table

The *'Enum_Values'* table contains the actual enumerated values for each of the enumeration types defined in table 'Enum_Types'. An entry is entered into this table for every value associated with the enumeration type. For example, the value 'red' would be one value entered for the enumeration type 'colors'.

Field Name	Format	Length
Enum_ID	Number	
Seq_Num	Number	
Value	Character	40

Table 1.2-5 Enum_Values Fields

The *'Enum_ID'* field is an integer value which identifies the enumeration type. Multiple occurrences of this value form the basis of the enumerated values for the enumeration type.

The *'Seq_Num'* field indicates the sequence in which the enumerated value was entered.

The *'Value'* field is a character value which represents one of the occurrences of this enumeration type.

1.3 NELS Collections

The section will discuss the tables which are used by NELS to describe the various collections created by a librarian. Table 1.3-1 NELS Collections provides a brief summary of each table used to define a collection in the library.

Table Name	Description
Librarians	Provides information about the persons authorized to make any make changes to the library.
Master	Provides information about the collection.

Table 1.3-1 NELS Collections

1.3.1 Librarians Table

The *'Librarians'* table is included in this section because librarians are the only persons authorized to add, update, move, or delete collections in the library. This table plays an important role in determining who can add, update, modify, copy, move, and delete library components. These components are the object classes, collections, objects, and addition of other librarians. The object classes are required entries in an NELS library. Without object classes, no metadata about an object could be entered. Collections are also required. These collections are place-holders for objects. It is also a means of organizing objects into a common collection. Collection hierarchies can be built to model a domain. Once the object classes and collections have been created, the metadata about the objects can be entered into the library. From this metadata, users can search for objects of interest similar to browsing the local library.

Field Name	Format	Length

User_ID	Character	35
Collection_ID	Character	22
Admin_Mask	Number	38
Custom_Mask	Number	38

Table 1.3-2 Librarians Fields

The *'User_ID'* field is a user identification string indicating that this user has certain administrative capabilities.

The *'Collection_ID'* field is the collection identifier which indicates the top-most collection in which these privileges are active. A user who has privileges at the specified collection, also has these same privileges at all sub-collections in this hierarchy. However, if this user has been given other privileges at a sub-collection, then the lower level privileges are enforced when the user traverses the hierarchy beyond this point. This means that a librarian can give a user all privileges at a high level, but take these privileges away at an sub-collection in the hierarchy.

The *'Admin_Mask'* field contains the administrative mask indicating the privileges the user has at or below the collection specified in subject_id.

The *'Custom_Mask'* field contains the administrative mask indicating the privileges the user has at or below this collection specified in subject_id. Initially, this mask is the same as the admin_mask. If the person defined by 'user_id' has been granted "update librarian" privileges, the user can enter the Update Librarian menu and change or customize his privileges. The user can only add or delete privileges given at creation. Privileges not granted can not be altered.

1.3.2 Master Table

The *'Master'* table contains the information about an NELS collection. A collection is similar to a container or a folder. This folder can hold different classes of objects and other collections (normally called sub-collections). Following is Table 1.3-3 Master Fields, which contains the fields used to describe a collection.

Field Name	Format	Length
Collection_ID	Character	22
Collection_Name	Character	35
Description	Character	240
Related_Count	Number	
Node_Type	Character	40
Password	Character	8
Total_Objects	Number	
Num_Production	Number	
Num_Developmental	Number	
Num_Secure	Number	
Num_Archive_Production	Number	
Num_Archive_Developmental	Number	
Num_Archive_Secure	Number	

Table 1.3-3 Master Fields

The *'Collection_ID'* field is the collection identifier of a collection. The highest level collection identifier is 'S'. All other collections are subordinate to this collection. Each subordinate collection is indicated by one character. For example, 'S2' would indicate the second subordinate collection under collection 'S'. Subordinate collections are labeled from 1 - 9, then from A through Z. A maximum of 35 subordinate collections can exist for each collection.

The *'Collection_Name'* field is the collection name given to the collection. This is the name that will appear on any collection lists.

The *'Description'* field is a brief description of the collection.

The *'Related_Count'* field is the number of related collections which contain similar or related information to this collection. The Collection_ID value(s) entered for this collection are entered into the Links table and the number of collection links is entered into this field.

The *'Node_Type'* field is an enumerated type field. Legal values are "production", "developmental", "secure", "archive production", "archive developmental", and "archive secure". Normal users will only see objects that are of type production. Librarians will be able to see all objects.

The *'Password'* field is reserved for future use with secure type collections.

The *'Total_Objects'* field contains the total number of "production" objects at this level and all levels (sub-collections) below this level.

The *'Num_Production'* field contains a count of all the objects in this collection which are of node_type "production".

The *'Num_Developmental'* field contains a count of all the objects in this collection which are of node_type "developmental".

The *'Num_Secure'* field contains a count of all the objects in this collection which are of node_type "secure".

The *'Num_Archive_Production'* field contains a count of all the objects in this collection which are of node_type "archive production".

The *'Num_Archive_Developmental'* field contains a count of all the objects in this collection which are of node_type "archive developmental".

The *'Num_Archive_Secure'* field contains a count of all the objects in this collection which are of node_type "archive secure".

1.4 NELS Objects

Table Name	Description
Objects	This table contains most of the metadata about each object and each tool in the library.
Abstracts	A subordinate table to Objects. It contains the value(s) which make up the abstract/description of the object.
Authors	A subordinate table to Objects. It contains the value(s) for the list of authors of an object.
Keywords	A subordinate table to Objects. It contains the subject terms/keywords associated with the object at each collection the object exists.
Links	A subordinate table to Objects. It contains the value(s) for a user-defined 'one-to-one' and 'one-to-many' link data type.
Long_Enum	A subordinate table to Objects. It contains the value for user-defined 'long' data type.
Preabstracts	A subordinate table to Objects. It contains the pre-processed abstract of the object.

Table 1.4-1 NELS Objects

1.4.1 Objects Table

A total of seven ORACLE tables are used to fully describe the metadata for an object entered into the NELS library. The primary table which contains a majority of the metadata is the 'objects' table. Table 1.4-2 Objects Fields defines the items which comprise this table. This table contains the most of the metadata about an object or a tool. Each object is represented by a single row (record) in the objects table.

Field Name	Format	Length
Collection_ID	Character	22
Object_Id	Number	
Class_ID	Character	22
Host_Name	Character	80
Format	Character	40
Object_Name	Character	80
PathName	Character	240
Version	Character	40
Lib_Entry_Date	Date	
Last_Modified	Date	
Flags	Number	
Node_Type	Character	40
Title	Character	80
field9	Character	80
field10	Character	80
...	... ...	
field50	Character	80

Table 1.4-2 Objects Fields

The *'Collection_ID'* field is the collection in which this objects is defined. This field and the object_id field uniquely identifies the object.

The *'Object_ID'* field is an internally generated value which is used by NELS to uniquely identify the object.

The *'Class_ID'* field indicates the object class of the object.

The *'Host_Name'* field identifies the host computer on which the object resides. This field will be implemented in a later version.

The *'Format'* field determines the objects format. For example, having ASCII as a format value would mean that the object defined by this metadata is ASCII format and can be viewed by an ASCII viewer.

The *'Object_Name'* field can be used by the user to supply additional object information.

The *'PathName'* field contains the complete path name or location of the object. This field must start with a '/' if the object is on-line, else the object is considered to be off-line. This location is used to view, copy, or print the contents of the object.

The *'Version'* field contains the version of the object that is being placed into the library.

The *'Lib_Entry_Date'* field contains the date in which the object was entered into the library. This field can not be altered by the user. It is determined by the system date.

The *'Last_Modified'* field contains the date in which the object was last modified. This field can not be altered by the user.

The *'Flags'* field indicates which metadata fields in other tables have values. The flags field will be implemented in a later version.

The *'Node_Type'* field is an enumerated type field which indicates the node type of the object. Legal values are "production", "developmental", "secure", "archive production", "archive developmental", and "archive secure".

The *'Title'* field contains a title or brief information about the object.

The *'field9'* through *'field50'* fields are generic character fields which are used to contain other user-defined metadata fields (attributes) about an object. The data type of each field is determined by its classid. If user-defined fields for an object class were defined, the information contained in the ObjectClasses, ClassFields, Enum_Types, and Enum_Values tables would describe the contents contained in one of these fields.

1.4.2 Abstracts Table

This table contains the abstract segments which describe the object or contents of an object. Multiple records with the same object_id can be entered to provide abstracts as large as 2400 characters. Table 1.4-3 Abstracts Fields defines the items which form this table.

Field Name	Format	Length
Collection_ID*	Character	22
Object_ID	Number	
Abstract	Long	

Table 1.4-3 Abstracts Fields

The *'Collection_ID'* field identifies the collection in which the object exists.

The *'Object_ID'* field identifies the object. This is the key that associates the abstracts with an object.

The *'Abstract'* field contains the actual abstract information that was entered by the user for the object. The abstract is broken into 240 character segments and entered into this table. The sequence number (Seq_Num) is incremented by 1 every time a new 240 character segment is required to contain the abstract. A maximum of 10 segments may be entered into this table at this time.

1.4.3 Authors Table

This table contains the list of authors associated with an object. Multiple records with the same Object_ID can be entered to provide an unlimited number of authors. Table 1.4-4 Authors Fields defines the items which form this table.

Field Name	Format	Length
Collection_ID*	Character	22
Object_ID	Number	
Author	Character	240

Table 1.4-4 Authors Fields

The *'Collection_ID'* field identifies the collection in which the object exists.

The *'Object_ID'* field identifies the object. This is the key that associates the authors with an object.

The *'Author'* field identifies the author of the object. A list of authors may be entered for any object. No limiti exits, other than storage and memory.

1.4.4 Keywords Table

This table contains the list of keywords associated with an object in a specific collection. Multiple records with the same Object_ID and different Collection_IDs can be entered to provide a separate set of keywords for the same Object_ID in different collections. Table 1.4-5 Keywords Fields defines the items which form this table.

Field Name	Format	Length
Collection_ID	Character	22
Object_ID	Number	
Keyword*	Character	35

Table 1.4-5 Keywords Fields

The *'Collection_ID'* field identifies the collection in which this object exists.

The *'Object_ID'* field identifies the object. This is the key that associates the keyword with an object and a collection.

The *'Keyword'* field contains the keyword entered for this object in this collection.

1.4.5 Links Table

This table contains the list of links (one-to-one and one-to-many) to other objects associated with object. It also contains the related collection for collections. Multiple records with the same Object_ID can be entered to provide an unlimited number of links. Table 1.4-6 Links Fields defines the items which form this table.

Field Name	Format	Length
Collection_ID	Character	22
Object_ID	Number	
Name*	Character	50
Link_Collection*	Character	22
Link_Object*	Character	15

Table 1.4-6 Links Fields

The *Collection_ID'* field identifies the collection associated with this link.

The *'Object_ID'* field identifies the object. For object links, this is the Object_ID of the object that contains a link to another object. For related collection links, this value is zero (0).

The *'Name'* field contains the database field name defined as a one-to-one or one-to-many link field attribute.

The *'Link_Collection'* field contains a Collection_ID value of the collection being linked.

The *'Link_Object'* field contains an Object_ID value of the object being linked.

1.4.6 Long_Enum Table

This table contains values of attributes which have been defined as a long data type. A field name is also required to uniquely identify this attribute. Table 1.4-7 Long_Enum Fields defines the items which form this table.

Field Name	Format	Length
Collection_ID*	Character	22
Object_ID	Number	
Name*	Character	50
Abstract	Character	240

Table 1.4-7 Long_enum Fields

The *'Collection_ID'* field identifies the collection in which the object exists.

The *'Object_ID'* field identifies the object. This key plus the 'name' value provide the key necessary to retrieve the long values for this object and field attribute.

The *'Name'* field contains the database field name defined as a long field attribute. This key plus the 'Object_ID' value provide the key necessary to retrieve the long values for this object and field attribute.

The *'Abstract'* field contains the value of the long field.

1.4.7 Preabstracts Table

This table contains the pre-processed abstract derived from the title, keywords, and abstract fields of the object. Table 1.4-8 Preabstracts Fields defines the items which form this table.

Field Name	Format	Length
Collection_ID	Character	22
Object_ID	Number	
Keywords*	Character	80

Table 1.4-8 Preabstracts Fields

The *'Collection_ID'* field identifies the collection in which this object exists.

The *'Object_ID'* field identifies the object. This key is necessary to retrieve the pre-processed abstract for the object.

The *'Keywords'* field contains the pre-processed keyword that will be used by the natural language search feature of NELS. This is a processed version of the title, keywords, and abstract fields of the object.

1.5 NELS Supplemental Tables

Table Name	Description
Admin_Defs	Contains pre-defined librarian classes and the privileges associated with this class.
Applications	Contains information regarding applications that can be run from the "Applications" menu, used as object viewers, and for output requests.
App_Formats	Contains legal format information pertaining to the applications defined in the Applications table.
History	Contains audit information about NELS usage.
Notify	Contains information about messages sent to users of NELS.
Output_Queue	Contains queued output requests which are processed just prior to the user exiting NELS.

Table 1.5-1 Supplemental Tables

1.5.1 Admin_Defs Table

This table is a supplemental table used by NELS. This table contains a list of pre-defined librarian classes. This list allows the head librarian to pre-define privileges for different types of librarians and assign these special privileges to a pre-defined librarian class. Table 1.5-2 Admin_defs describes the items which form the Admin_defs table.

Field Name	Format	Length
Name	Character	35
Mask	Number	38

Table 1.5-2 Admin_defs Fields

The *'Name'* field contains the name assigned to the privileges for a specified librarian class. For example, "Head Librarian" would be a name of a librarian who has all administrative privileges assigned. This means that a librarian with these privileges could perform any of the administrative functions currently available in NELS.

The *'Mask'* field contains the preset privileges allowed for this type of librarian. A librarian with these privileges can only perform those functions that are determined by the value in the mask.

1.5.2 Applications Table

This table provides the necessary information required by NELS to build a command that will execute the application. The application can be activated from the "Applications" menu if defined as an application. If the application is defined as an object viewer, the application will be activated when an object of the format specified for the application is selected for viewing in the "Object Browser" window. If the application is an "output request" type application, the application is executed in the same manner as the object viewer. Table 1.5-3 Applications Fields describes the items which form this table.

Field Name	Format	Length
Name*	Character	40
Description	Character	80
AppType	Character	20
Process	Character	20
Path	Character	80
User Mode	Character	5
Parameters	Character	240

Table 1.5-3 Applications Fields

The *'Name'* field contains the application name that will be used to uniquely identify the application to execute.

The *'Description'* field contains a short description of the application.

The *'AppType'* field identifies the type of application or intended use of the application. Legal selections are "application", "viewer", "archive", and "output request"

The *'Process'* field identifies the processing for the application. The application is either processed "immediately" or is "queued" for later processing.

The *'Path'* field identifies the complete path name of the executable program.

The *'User_Mode'* field identifies the mode of operation of the application. The User_Mode selections are "X" if the application only runs under X windows, "ASCII" if the application only runs in character mode, or "Both" if the application can execute in either mode.

The *'Parameters'* field identifies any parameter(s) that is to be sent the the application defined.

1.5.3 App_Formats Table

This table provides the formats that the applications defined in the Applications table can handle. The format entered for the application should be the word "all" or one of the format values listed in the add/update objects window. Table 1.5-4 Appl_Formats Fields describes the items which form this table.

Field Name	Format	Length
Name*	Character	40
AppType	Character	20
User_Mode	Character	5
Format	Character	40

Table 1.5-4 App_Formats Fields

The *'Name'* field contains the application name associated with the format entered.

The *'AppType'* field identifies the type of application or intended use of the application. Legal selections are "application", "viewer", "archive", and "output request".

The *'User_Mode'* field identifies the mode of operation of the application. The User_Mode selections are "X" if the application only runs under X windows, "ASCII" if the application only runs in character mode, or "Both" if the application can execute in either mode.

The *'Format'* field contains the format that the application can process. A value of "all" indicates that all object formats or files can be processed.

1.5.4 History Table

This table provides an audit trail of many of the actions perform against an object. Actions such as add, update, delete, viewing, copying, etc are recorded in this history file. Table 1.5-5 History Fields describes the items which form this table.

Field Name	Format	Length
User_ID	Character	35
Action	Number	3
Acct_Date	Date	

Collection_ID	Character	22
Object_ID	Number	

Table 1.5-5 History Fields

The *'User_ID'* field identifies the user who performed the action.

The *'Action'* field describes the action which took place. Such as Copied or Browsed.

The *'Acct_Date'* field contains the date and time the action took place.

The *'Collection_ID'* field identifies the collection upon which the action occurred.

The *'Object_ID'* field identifies the object upon which the action occurred.

1.5.5 Notify Table

The notify table is a table used for passing messages from one user to another. This table is also used to notify users of changes to objects. Notification of changes to objects only occurs if the object was copied by the user. Table 1.5-6 Notify Fields describes the items which form the notify table.

Field Name	Format	Length
User_ID	Character	35
Object_ID	Number	
Notifier	Character	1
New_Vers_ID	Character	40
Message	Character	240
RetainDate	Date	
From_User	Character	35
Time	Date	

Table 1.5-6 Notify Fields

The *'User_ID'* field identifies the user to whom the message is to be sent.

The *'Object_ID'* field identifies the object, if any, associated with the message. This is used whenever a user wants to be notified of changes to an object.

The *'Notifier'* field identifies the type of notify message that to be sent.

The *'New_Vers_ID'* field contains the new version id of the object specified in the object_id field above, if any.

The *'Message'* field contains the message to be sent to the user. This message only applies to messages sent by another user. Messages related to changes associated with an object are formulated by the NELS program based on the value of notifier.

The *'RetainDate'* field contains the last date that the message will be displayed to all users. This field only applies to messages sent to all users.

The *'From_User'* field contains the user id of the person sending the message. This field only applies to messages sent from one user to another or from one user to all users.

The *'Time'* field contains the date and time the message is sent.

1.5.6 Output_Queue Table

The Output_Queue table is a table used to store the output request which was specified in a browsing session. The output request is placed into this table until the user exits NELS. Upon exiting, the use is given the option of cancelling the requests or processing the requests at this time. The requests are then removed from the table. Table 1.5-7 Output_Queue Fields describes the items which form the output queue table.

Field Name	Format	Length
User_ID*	Character	35
Directory	Character	80
Command	Character	240

Table 1.5-7 Output_Queue Fields

The *'User_ID'* field identifies the user requesting the delayed output request.

The *'Directory'* field contains the directory where the files for output can be found.

The *'Command'* field contains the actual command that is to be issued to complete the output request.

1.6 NELS Index Tables

To provide more efficient retrieval of data, NELS uses index tables. Table 1.6-1 Index Tables lists all the tables for which index tables have been created. This table shows the table name, index table name, and the field names which are used to define a key for that table.

Table Name	Index Table Name	Key Fields
Abstracts	iabstracts	Collection_ID, Object_ID
Admin_Defs	iadmin_defs	Name
Applications	iapplications	AppType, Name
App_Formats	iapp_formats	Name, Format
Authors	iauthors	Collection_ID, Object_ID
ClassFields	iclassfields	ClassID, SeqNum
Enum_Types	ienum_types	Enum_ID
Enum_Values	ienum_values	Enum_ID, Seq_Num
History	ihistory	Acct_Date, Action
Keywords	ikeywords	Collection_ID, Object_ID, Keyword
Librarians	ilibrarians	User_ID, Collection_ID
Links	ilinks	Collection_ID, Object_ID, Name
Long_Enum	ilong_enum	Collection_ID, Object_ID, Name
Master	imaster	Collection_ID
Master	imaster2	Node_Type, Collection_ID
Notify	inotify	User_ID
ObjectClasses	iobjectclasses	ClassID
Objects	iobjects	Collection_ID, Object_ID, Class_ID
Objects	iobjects2	Class_ID, Collection_ID, Object_ID
Objects	iobjects3	Node_Type, Class_ID, Collection_ID, Object_ID
Preabstracts	ipreabstracts	Collection_ID, Object_ID

Table 1.6-1 Index Tables

1.7 Schema Hierarchy

The NELS 1.2 schema hierarchy is illustrated by Figure 1-1 NELS Table Hierarchy.

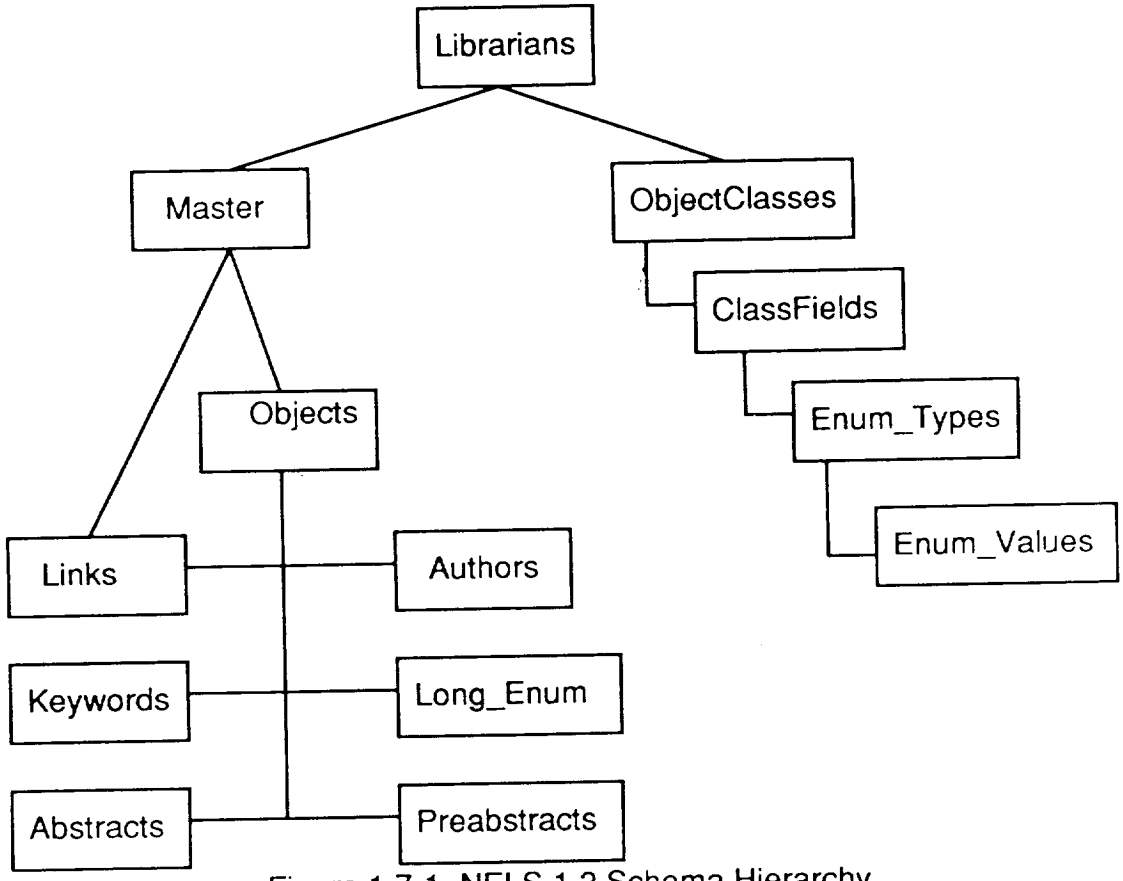

Figure 1.7-1 NELS 1.2 Schema Hierarchy

1.8 NELS Table Relationships

This section discusses the relationships of the tables to other tables in regards to their use by NELS. Each table in the hierarchy of Figure 1.7-1 above is discussed in greater detail in the following paragraphs. The use of each specific field for each table will not be discussed at this time.

1.8.1 Access Privileges

The librarians table contains the necessary information required by NELS to validate whether the current user has privileges to perform various NELS administrative functions. Every NELS library must have a minimum of one "Head Librarian". This person has full authorization to add, update, modify, and delete object classes, collections, tools, and objects in the library. The "Head Librarian" is assigned all privileges from collection "S" and all subordinate level collections. The 'user_id' and 'subject_id' fields are used to determine the current user privileges. Without this table, no other table in the library can be modified.

1.8.2 Collections

A collection consists of a defined area in the library (masters table), the tools associated with the collection (tools table), and the objects associated with the collection (objects table and subordinate tables). The masters tables contains the information about all collections in the library.

When a librarian performs an administrative function on a collection, the tables below this table may be affected. Depending on the function being performed, different tables may be modified. If a new collection is being added or modified in the library, only the masters table is changed by the function. However, if a move or delete function is performed, the 'subject_id' value would be used to move/remove all entries from the tools, members, and associated objects tables having that collection value. All subordinate collections would also be moved/removed. Additionally, any librarians that were defined for any of these collections would be modified/deleted from the librarians table.

Once collections have been created, tools and objects may be entered into the library. When tools are added to a collection, the information about the tool is entered into the 'tools' table and the 'objects' table. When objects are entered into a library collection, the 'members' table and the 'objects' tables are changed. Note that objects that are entered into the library also depend on information contained in the 'objectclasses' table and subordinate tables. The objectclasses table hierarchy determines what information is to be entered for the object based upon the object class that has been selected for that object. This is the 'type_id' field in the objects table.

The 'objects' table is the primary table used in defining the metadata about an object entered into the library. The 'objects' table is composed of the "objects" table and 6 subordinate tables. The six other tables which complete the definition of a object's metadata are the 'authors', 'abstracts', 'preabstracts', 'keywords', 'long_enum', and 'links' table. All of these tables use the 'object_id' field to uniquely identify the values associated with a particular object. However, the keywords table, uses the subject_id and object_id fields to uniquely identify the keywords for a particular object. This use of subject_id (collection id), provides the addition of collection specific keywords, since an object can be used in more than one collection.

1.8.3 Object Classes

Object classes must be defined by librarians before an object of this class can be entered into the library. To define an object class, the hierarchy relationship depicted by Figure 1.7-1 is used. The objectclasses table contains basic information about the object class being defined. Each object class is given a unique identifier, 'class_id' field, which can be used to obtain information about all the field attributes associated with an object class. This value is entered into the 'type_id' field of the objects table when a new object is added to the library.

The classfields table contains all the field attributes which make up this object class. When field attributes are inherited from another base class, the field attributes of this class plus the field attributes of the base class are used to complete the object class definition. For example, "TG1" would require all the field attributes of object class "T" (the generic class) plus all the field attributes of "TG1". Note, all object classes other than the generic class "T", contain two characters.

If an attribute field in the classfields table is an enumeration type field, then information about the enumeration type can be found by using the 'enum_id' foreign key field. The 'enum_types' and 'Enum_Values' tables provide information about the definition of the enumeration type and the enumeration values.

Appendix A SQL for NELS 1.2 Tables

```
/* Create tables for NELS */

CREATE TABLE Abstracts      (
 Collection_ID              CHARACTER(22)
 Object_ID                  NUMBER,
 Abstract                   LONG                    );

CREATE TABLE Output_Queue   (
 UserID                     CHAR(35)                NOT NULL,
 Directory                  CHAR(80)                NOT NULL,
 Command                    CHAR(240)               NOT NULL);

CREATE TABLE Authors        (
 Collection_ID              CHARACTER(22)
 Object_ID                  NUMBER                  NOT NULL,
 Author                     CHAR(240)               NOT NULL);

CREATE TABLE Enum_Values    (
 Enum_ID                    NUMBER                  NOT NULL,
 Seq_Num                    NUMBER                  NOT NULL,
 Value                      CHAR(40)                );

CREATE TABLE Enum_Types     (
 Enum_ID                    NUMBER                  NOT NULL,
 Enum_Type                  CHAR(40)                NOT NULL,
 Name                       CHAR(50)                NOT NULL);


CREATE TABLE History        (
 User_ID                    CHAR(35)                NOT NULL,
 Action                     NUMBER(3)               NOT NULL,
 Acct_Date                  DATE,
 Collection_ID              CHAR(22),
 Object_ID                  NUMBER                  );

CREATE TABLE Keywords       (
 Collection_ID              CHAR(22),
 Object_ID                  NUMBER,
 Keyword                    CHAR(35)                );

CREATE TABLE Links          (
 Collection_ID              CHAR(22)                NOT NULL,
 Object_ID                  NUMBER                  NOT NULL,
 Name                       CHAR(50),
 Link_Collection            CHAR(22),
 Link_Object                CHAR(15)                );

CREATE TABLE Long_Enum      (
 Collection_ID              CHARACTER(22)
```

```
 Object_ID                  NUMBER          NOT NULL,
 Name                       CHAR(50),
 Abstract                   CHAR(240)       NOT NULL);

CREATE TABLE Master         (
 Collection_ID              CHAR(22)        NOT NULL,
 Collection_Name            CHAR(35)        NOT NULL,
 Description                CHAR(240),
 Related_Count              NUMBER,
 Node_Type                  CHAR(40)        NOT NULL,
 Password                   CHAR(8),
 Total_Objects              NUMBER,
 Num_Production             NUMBER,
 Num_Developmental          NUMBER,
 Num_Secure                 NUMBER,
 Num_Archive_Production     NUMBER,
 Num_Archive_Developmental  NUMBER,
 Num_Archive_Secure         NUMBER          );

CREATE TABLE Notify         (
 User_ID                    CHAR(35),
 Object_ID                  NUMBER,
 Notifier                   CHAR(1),
 New_Vers_ID                CHAR(40),
 Message                    CHAR(240),
 RetainDate                 DATE,
 From_User                  CHAR(35),
 Time                       DATE            );

CREATE TABLE Objects        (
 Collection_ID              CHAR(22)        NOT NULL,
 Object_ID                  NUMBER          NOT NULL,
 Class_ID                   CHAR(22)        NOT NULL,
 Host_Name                  CHAR(80),
 Format                     CHAR(40)        NOT NULL,
 Object_Name                CHAR(80),
 PathName                   CHAR(240),
 Version                    CHAR(40),
 Lib_Entry_Date             DATE,
 Last_Modified              DATE,
 Flags                      NUMBER,
 Node_Type                  CHAR(40),
 Title                      CHAR(80),
 FIELD9                     CHAR(80),
 FIELD10                    CHAR(80),
 FIELD11                    CHAR(80),
 FIELD12                    CHAR(80),
 FIELD13                    CHAR(80),
 FIELD14                    CHAR(80),
 FIELD15                    CHAR(80),
 FIELD16                    CHAR(80),
 FIELD17                    CHAR(80),
 FIELD18                    CHAR(80),
 FIELD19                    CHAR(80),
```

```
FIELD20                 CHAR(80),
FIELD21                 CHAR(80),
FIELD22                 CHAR(80),
FIELD23                 CHAR(80),
FIELD24                 CHAR(80),
FIELD25                 CHAR(80),
FIELD26                 CHAR(80),
FIELD27                 CHAR(80),
FIELD28                 CHAR(80),
FIELD29                 CHAR(80),
FIELD30                 CHAR(80),
FIELD31                 CHAR(80),
FIELD32                 CHAR(80),
FIELD33                 CHAR(80),
FIELD34                 CHAR(80),
FIELD35                 CHAR(80),
FIELD36                 CHAR(80),
FIELD37                 CHAR(80),
FIELD38                 CHAR(80),
FIELD39                 CHAR(80),
FIELD40                 CHAR(80),
FIELD41                 CHAR(80),
FIELD42                 CHAR(80),
FIELD43                 CHAR(80),
FIELD44                 CHAR(80),
FIELD45                 CHAR(80),
FIELD46                 CHAR(80),
FIELD47                 CHAR(80),
FIELD48                 CHAR(80),
FIELD49                 CHAR(80),
FIELD50                 CHAR(80)            );

CREATE TABLE Preabstracts   (
 Collection_ID          CHAR(22)            NOT NULL,
 Object_ID              NUMBER              NOT NULL,
 Keywords               CHAR(80)            );

CREATE TABLE Librarians     (
 User_ID                CHAR(35),
 Collection_ID          CHAR(22),
 Admin_Mask             NUMBER(38),
 Custom_Mask            NUMBER(38)          );

CREATE TABLE ClassFields    (
 ClassID                CHAR(22)            NOT NULL,
 Name                   CHAR(50)            NOT NULL,
 Prompt                 CHAR(50)            NOT NULL,
 Kind                   NUMBER(38)          NOT NULL,
 Visibility             NUMBER(38)          NOT NULL,
 Editable               NUMBER(38)          NOT NULL,
 Length                 NUMBER(38),
 NumColumns             NUMBER(38),
 NumLines               NUMBER(38),
 FieldUnique            CHAR(1),
```

```
 SeqNum                  NUMBER(38),
 Enum_ID                 NUMBER          );

CREATE TABLE ObjectClasses  (
 ClassID                 CHAR(22)        NOT NULL,
 ClassName               CHAR(80)        NOT NULL,
 NumFields               NUMBER(38)      );

CREATE TABLE Applications   (
 Name                    CHAR(40),
 Description             CHAR(80),
 AppType                 CHAR(20),
 Process                 CHAR(20),
 Path                    CHAR(80),
 User_Mode               CHAR(5),
 Parameters              CHAR(240)       );

CREATE TABLE App_Formats    (
 Name                    CHAR(40),
 AppType                 CHAR(20),
 User_Mode               CHAR(5),
 Format                  CHAR(40)        );

CREATE TABLE Admin_Defs     (
 Name                    CHAR(35),
 Mask                    NUMBER(38)      );

CREATE TABLE Output_Queue   (
 Userid                  CHAR(35)        NOT NULL,
 Directory               CHAR(80),
 Command                 CHAR(240)       NOT NULL);

/* Create a sequence number that will be used for object IDs */
CREATE SEQUENCE NewObjectID INCREMENT BY 1 START WITH 101
 NOMAXVALUE MINVALUE 100 NOCYCLE CACHE 50 ORDER;

COMMIT WORK;
QUIT;
```

Appendix B SQL for NELS 1.2 Table Indices

```
/* Create index files for NELS tables */

CREATE UNIQUE INDEX iabstracts ON Abstracts(Collection_ID, Object_ID);

CREATE UNIQUE INDEX iadmin_defs ON Admin_Defs(Name);

CREATE UNIQUE INDEX iapplications ON Applications(Apptype, Name);

CREATE UNIQUE INDEX iapp_formats ON App_Formats(Name, Format);

CREATE          INDEX iauthors ON Authors(Collection_ID, Object_ID);

CREATE UNIQUE INDEX iclassfields ON ClassFields(ClassID, SeqNum);

CREATE UNIQUE INDEX ienum_types ON Enum_Types(Enum_ID);

CREATE UNIQUE INDEX ienum_values ON Enum_Values(Enum_ID, Seq_Num);

CREATE UNIQUE INDEX ihistory ON History(Acct_Date, Action);

CREATE UNIQUE INDEX ikeywords ON Keywords(Collection_ID, Object_ID, Keyword);

CREATE UNIQUE INDEX ilibrarians ON Librarians(User_ID, Collection_ID);

CREATE          INDEX ilinks ON links(Collection_ID, Object_ID, Name);

CREATE UNIQUE INDEX ilong_enum ON Long_Enum(Collection_ID, Object_ID, Name);

CREATE UNIQUE INDEX imaster ON Master(Collection_ID);

CREATE UNIQUE INDEX imaster2 ON Master(Node_Type, Collection_ID);

CREATE UNIQUE INDEX inotify ON Notify(User_ID, Time);

CREATE UNIQUE INDEX iobjectclasses ON ObjectClasses(ClassID);

CREATE UNIQUE INDEX iobjects ON Objects(Collection_ID, Object_ID, Class_ID);

CREATE UNIQUE INDEX iobject2 ON Objects(Object_ID, Class_ID, Collection_ID);

CREATE UNIQUE INDEX iobjects3 ON Objects(Node_Type, Class_ID, Collection_ID, Object_ID);

CREATE          INDEX ipreabstracts ON Preabstracts(Object_ID, Collection_ID);

COMMIT WORK;
QUIT;
```