



Data Discovery & Documentation PROCEDURE

Document Version: 1.0
Date of Issue: June 28, 2013



Table of Contents

1. Introduction	3
1.1 Purpose.....	3
1.2 Scope.....	3
2. Option 1: Current Process – No metadata available	4
3. Option 2: Manual Process – Metadata Repository Template	6
3.1 SEARCH / RESEARCH Procedures	8
3.2 INSERT and UPDATE Procedures.....	13
3.3 DELETE (EXPIRE) Procedures.....	23
4. Option 3: Automated Process – Metadata Repository Tool	32
4.1 SEARCH / REPORT	33
4.2 CREATE.....	33
4.3 CHANGE MANAGEMENT.....	35
4.4 OTHER.....	35
Appendix A – Current Application Delivery Process Diagram	34
Appendix B – Metadata Repository Logical Model (LDM).....	35
Appendix C – Metadata Repository Template Glossary.....	36
Appendix D – Proposed Application Delivery Process Diagram.....	42

1. Introduction

1.1 Purpose

The purpose of this document is to describe a suggested set of procedures to be utilized for the ongoing creation and maintenance of metadata (i.e. *data about data*) within the Application Delivery and Database Services processes.

1.2 Scope

This document identifies and describes 3 separate options for addressing and implementing metadata discovery and documentation initially intended to enable the Application Delivery and Database Services personnel involved in the development and maintenance of Applications to collect and maintain metadata that is non-existent today. The 3 options described are:

- a) Option 1:** Metadata is not available (i.e. no repository)
- b) Option 2:** Metadata is collected and housed in an excel spreadsheet designed for this purpose (i.e. an inactive, manual repository)
- c) Option 3:** Metadata is collected and housed in a functioning repository tool (i.e. an active, automated repository)

At the time of this writing, the metadata housed, collected, and maintained will be limited to basic information about the following entities and the relationships that exist between them where that information currently exists and is available to be collected. (see *Appendix B – Metadata Logical Data Model*)

- Agencies and Agency Divisions
- Applications (*i.e. process structures*)
- Databases (*i.e. data structures*)
- Business Subject Areas, Entities, and Attributes (*currently unavailable*)
- Master Lists of Reference Data
 - Database Types
 - Program Languages
 - Object Types

2. Option 1: Current Process – No metadata available

Each Project Request requires labor intensive research to determine what data and process or system may be in production. All must be found and identified so as to determine what the impact will be regarding what is to be added, changed, or deleted by the Project. It also needs to be determined what may potentially be impacted outside as well as inside the scope of the current project (i.e. *other* dependent system applications and data bases). Since this information is currently not documented in any shareable, repeatable, and accessible way, whatever information is gathered in the process of discovery is currently lost upon the completion of the Project. Therefore, each project request requires the same due diligence and manual research be repeated and does not leverage prior research in any repeatable or reusable fashion.

In other words, the discovery cycle must be repeated by each project team for each project request. The pertinent metadata produced by the Project is only documented to the extent necessary to facilitate implementation of that specific project. That documentation is not recorded in such a way as to be retained and made readily accessible for future project teams.

Each Business Analyst, DBA, and Developer must rely on their own acquired knowledge, inquire of others, and search for data structures, programs, documents, etc. in a very time consuming and tedious way. Negative impacts are often only discovered during systems testing and sometimes only after the system has gone into production. The implications of the problems this could (and does) cause should be of serious concern to all. (See Appendix A – Current Application Delivery Process). This current process does not lend itself to meeting the ultimate objective of agency interoperability through data sharing. Metadata type information is needed and/or should be created and captured during each phase of the application development cycle as well as referenced for assessing impact during the initial project request intake stage.

Project Request Phase

Business-defined Subject Areas, Entities, and Attributes should be identified and/or referenced during this phase. However, currently there is no enterprise level conceptual or logical model in place that would define and identify these for the State, across all Agencies, or even within one Agency. Therefore, the connection between the business and system applications is not and cannot be documented at this time.

Project Requirements

This would be the time to identify and enhance any logical data models that are in place to understand how the requested enhancements fit in, work with, and/or impact the enterprise. However, as already noted, at the time of this

writing there are no logical models for the State and/or its Agencies in place and therefore, no strategic plan to use as the guide.

Detailed Design

This phase requires that current applications and data structures that could be involved or impacted by this project are sourced and identified. Currently, there is no one way to accomplish this task. Each business analyst, developer, and DBA involved must perform their own search/research through database queries, application documentation, previous project documentation (if available), files, etc. in the hope of discovering any and all of the relevant items to be involved and/or impacted. Unidentified impacts are often only discovered during systems testing and sometimes only after the application has moved into production.

During the detail design phase, the new items that are created and named should be documented for future reference. All new databases, tables, columns, programs, and objects need to be uniquely identified, named, and described and relationships (i.e. links) between them. However, there are neither standards for performing documenting of this kind nor any centralized repository to facilitate this effort.

Development

Any changes or additions to the items that were identified and defined in the design phase should be noted and the documentation adjusted accordingly.

Testing

Any changes or additions to the items that were identified and defined in the design phase should be noted and the documentation adjusted accordingly. Also, any other items that are discovered to be impacted need to also be noted and documented.

Implementation

All information, process items, data items, documents, instructions, etc. needs to be documented and packaged in such a way as to be accessed and referenced by future projects. At this point in time, this is in the hands of individuals (e.g. configurations managers) and not readily known or available to future projects teams or other interested parties. Pulling together all the "pieces" that were presented for implementation after the fact has proven to be cumbersome at best, but essentially impossible to document.

3. Option 2: Manual Process – Metadata Repository Template

This process defines the steps for utilizing and, in turn, building upon a common, reusable, and shareable Metadata Repository to facilitate data discovery efforts. The Metadata Repository is based on a template that is designed to “inventory” the major components of all Applications and Databases and their relationships or links. Documentation disciplines must be adhered to as described below in order to build out the information to be contained in this Repository. This will allow for a robust wealth of metadata to be collected with integrity of data maintained for use in the fulfillment of ongoing project requests through application and database development. It requires some adjustments to the current Application Delivery (and Database Design) process to incorporate the use and capture of the relevant metadata (**See Appendix D – Proposed Application Delivery Process**).

The Logical Data Model (i.e. LDM - see **Appendix B**) is the blueprint for the Repository Template. Understanding this LDM, is key to understanding the Metadata Repository and how it needs to function and be maintained. The LDM depicts the Entities and Attributes that make up the Repository and most importantly the relationships between them (see **Appendix C** for definitions). It is necessary to understand the dependencies that exist between these entities in order to understand how to successfully manage the Metadata Repository to keep it functioning and to endure the ongoing integrity of its information.

***For example:** A COLUMN entry cannot exist before the TABLE entry exists in the Repository. A TABLE entry cannot exist before a DATABASE entry exists in the Repository, etc.*

Also, by understanding the LDM, you will understand how items are “linked” to one another thereby defining a research “trail” for you to follow for a data discovery effort.

***To illustrate:** A project request is asking that a change be made to a Table. By employing the Repository, we first find the Table in question. Then we can see what Columns are in the Table, what Programs are using this Table, etc. For each Program, we can find all the Objects (Stored Procedures, Triggers, Data Windows, etc.) associated with this Program. In short, we can identify all the process and data items that could be impacted by the proposed change to the Table and plan accordingly.*

It will be **important to follow** the described order of the steps as described in performing each function to maintain the integrity of the metadata repository and also to ensure identification or discovery of all related items to assess possible impacts.

In the following pages, each of the function options are outlined and accompanied by a process flow diagram to clarify the order of the steps to be taken.

It should be noted that rather than creating a dynamic application and database development environment, this manual repository and documentation process as described here can only provide a common reference facility for metadata discovery and access. It will be necessary to establish governance e standards and procedures in order to manage its ongoing use and to secure the integrity of the information to be housed within the repository. The repository should be implemented on a centralized, web-accessible server that will allow at least browse access to ALL authorized users (e.g. Sharepoint).

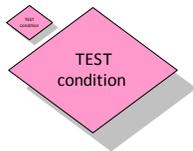
Guide for reading/understanding the Process Flow diagrams:



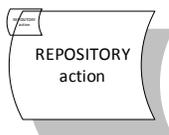
Start of the repository function process.



Action step.



Test result of action step.



Action to be taken in the Repository based on test result.



End of the process.



Alternate to END. The process continues on to another process.



Alternate to START. The process continues here from another process.

3.1 SEARCH / RESEARCH Procedures

Data items are found by searching for known Columns, Tables, and/or Databases in each of their respective TABS within the Metadata Repository Template. Once the item is found, the relationships to the others can be identified.

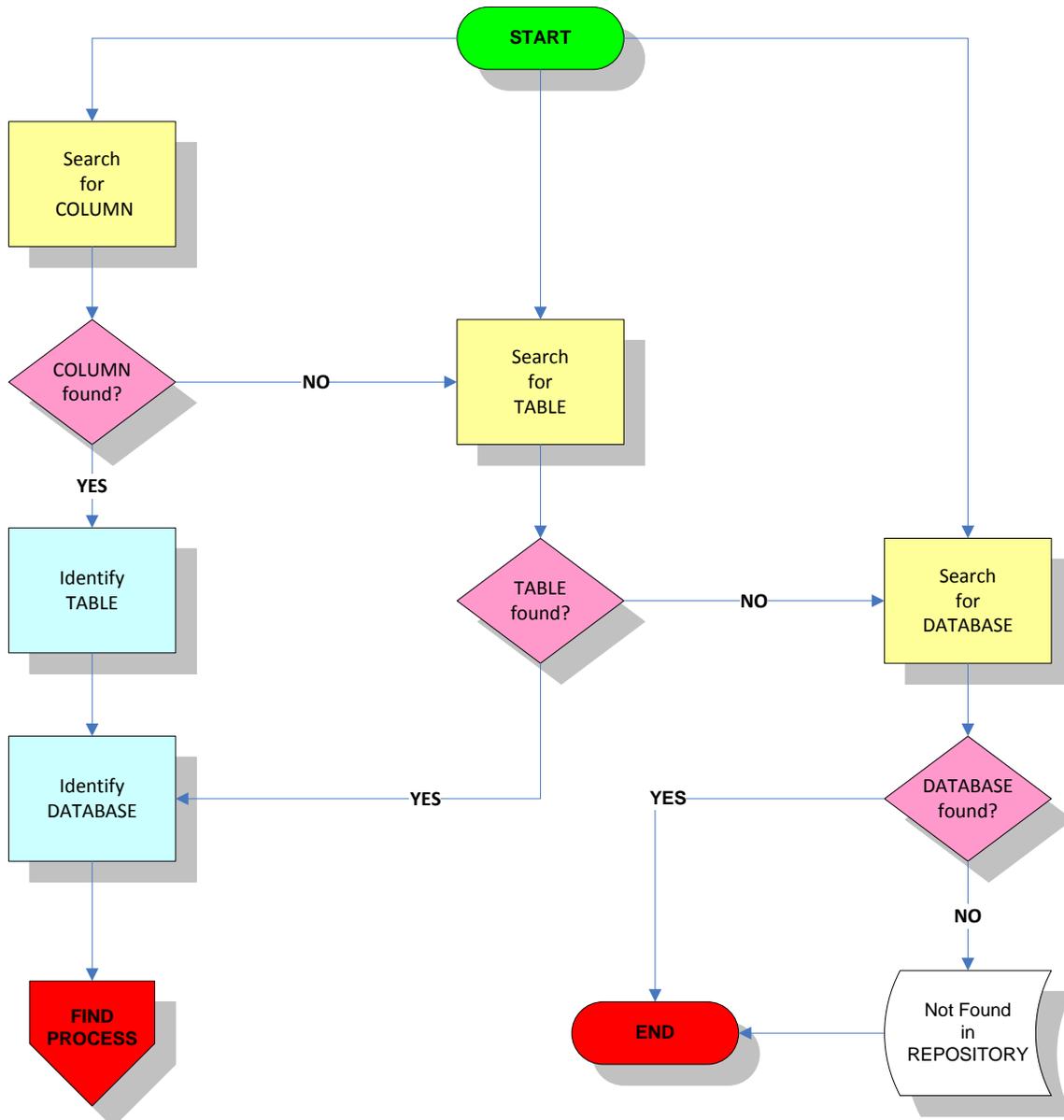
For example: Find a specific column name in the COLUMN worksheet. Once that column is found, then you will discover to what TABLE the Column belongs. On the TABLE worksheet, you will then, in turn, be able to discover in what DATABASE the TABLE belongs, etc.

Process items are found by searching for known Programs, Applications, and/or Objects in each of their respective TABS within the Metadata Repository Template. Once the item is found, the relationships to the others can be identified.

For example: Find a specific program name in the PROGRAM worksheet. Once that program is found, then you will discover to what APPLICATION the Program belongs. On the APPLICATION worksheet, you will then, in turn, be able to discover what DIVISION (and AGENCY) "owns" the APPLICATION. You will also, now be able to discover all the OBJECTS associated (i.e. used by) the Program, etc.

Step 1 Search for DATA items

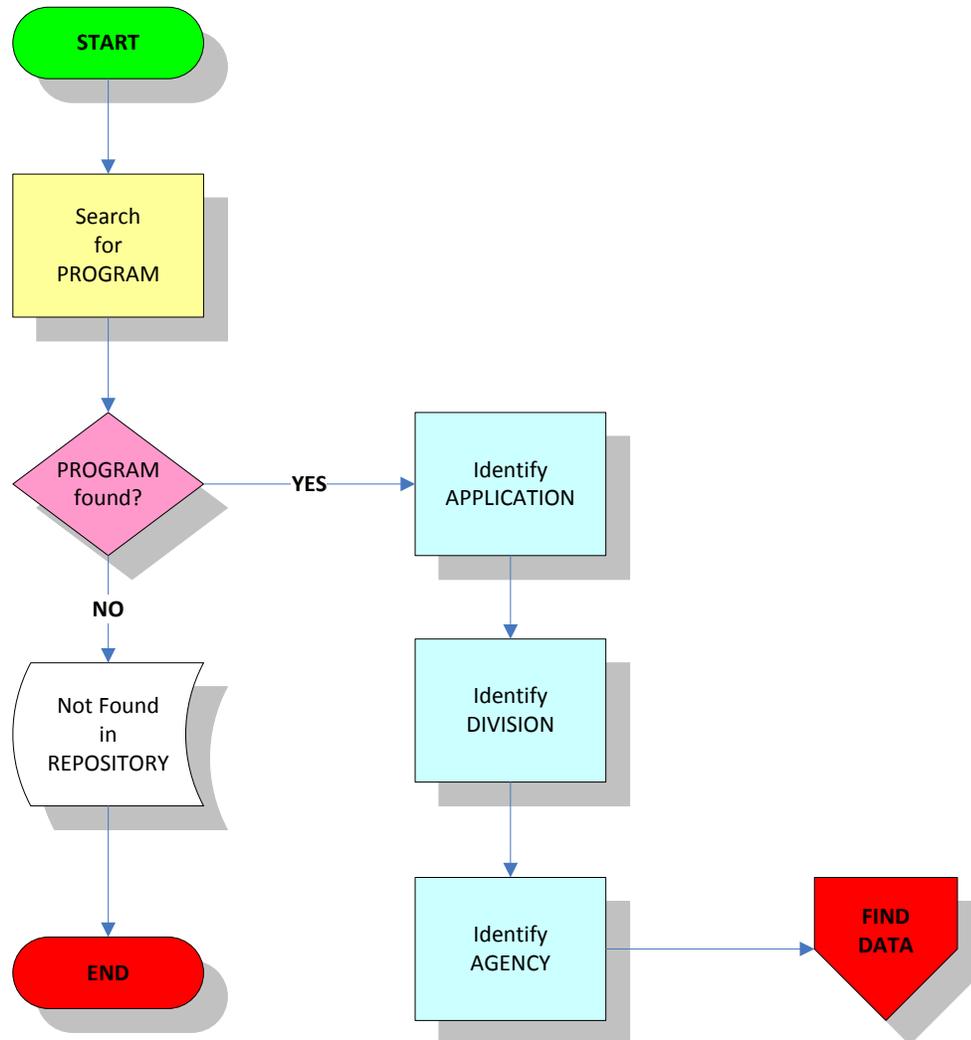
- a. Search by Column
- b. Search by Table
- c. Search by Database



Process Diagram A – Search for a Data items

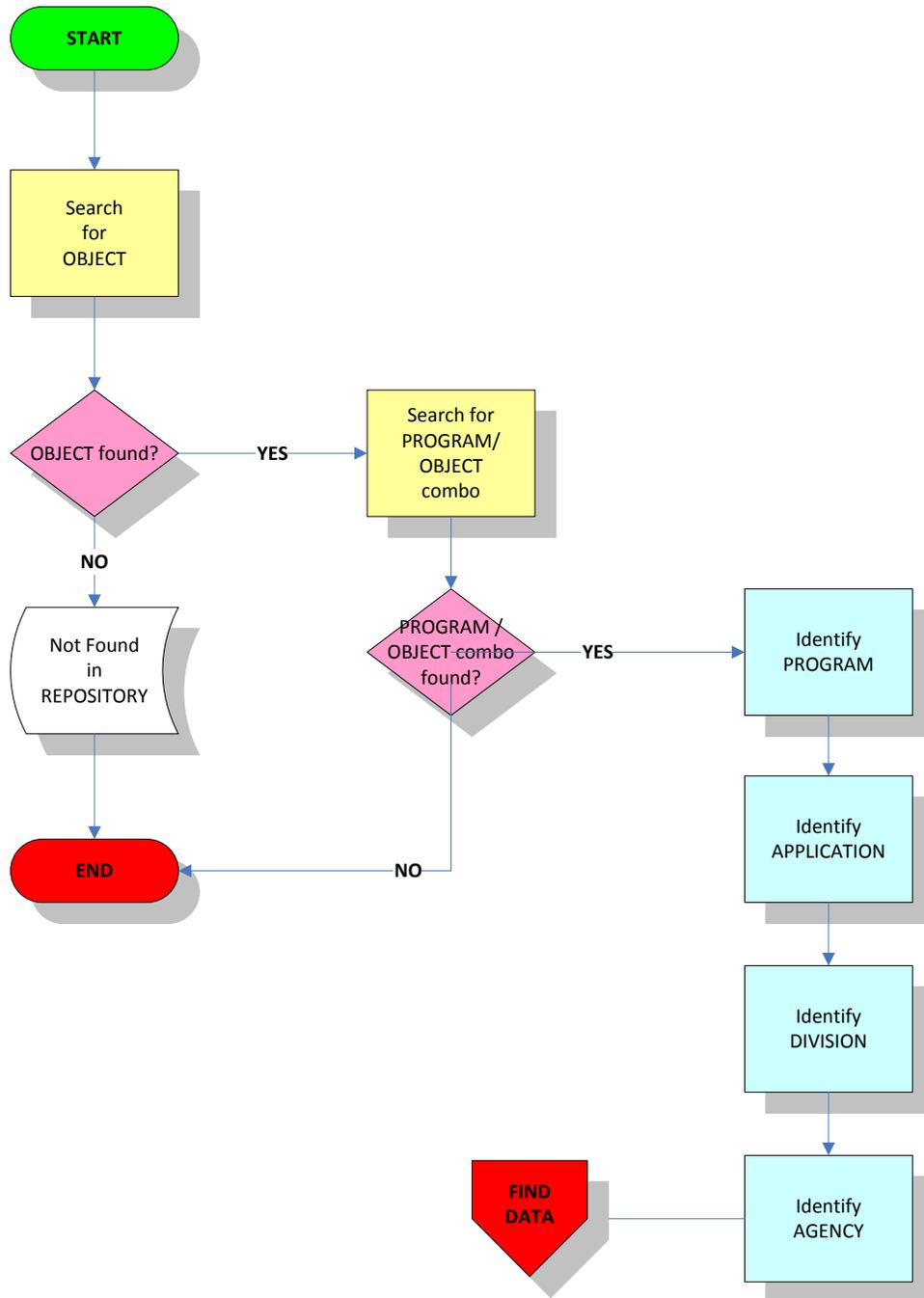
Step 2 Search for PROCESS items

a. Search by Program



Process Diagram B – Search for a Program

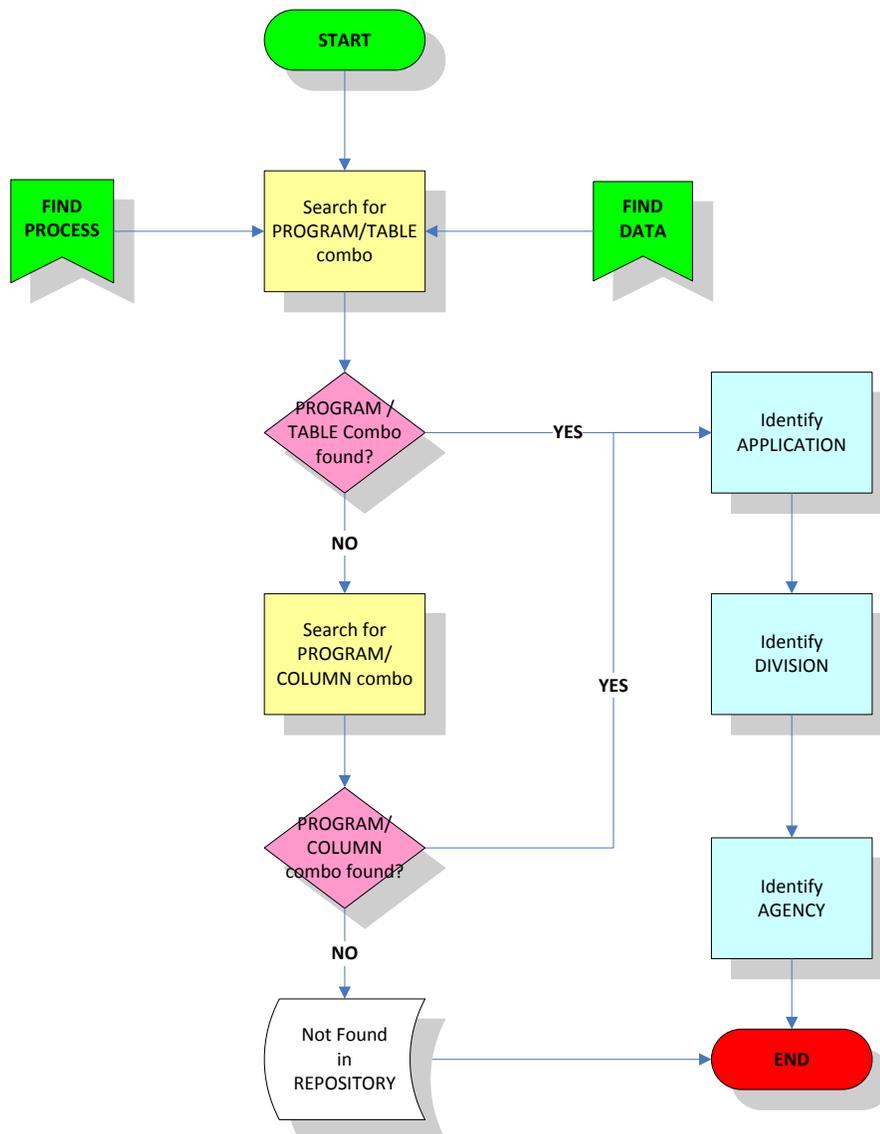
b. Search by Object (Process Diagram C)



Process Diagram C – Search for an Object

Step 3 Search for Relationships between DATA and PROCESS

- a. Search for Program/Table combination
- b. Search for Program/Column combination



Process Diagram D – Search for a Program/Data combination

3.2 INSERT and UPDATE Procedures

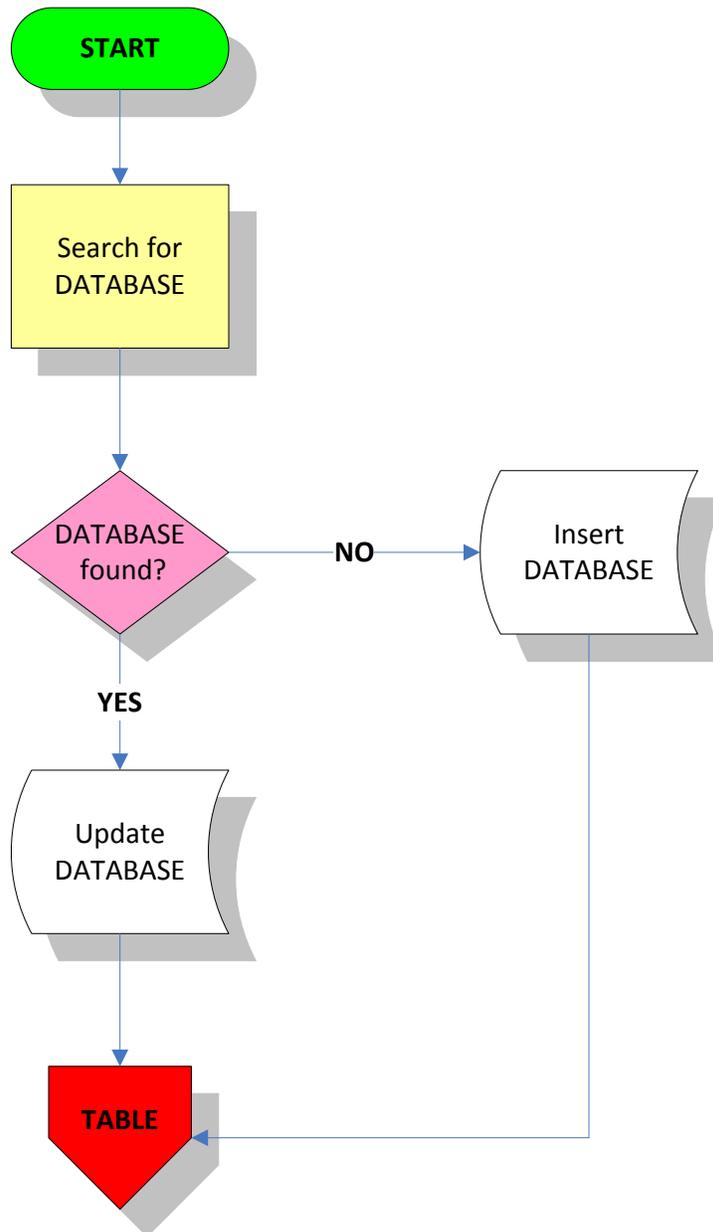
The rules of dependencies must be understood and adhered to when adding or inserting new entries into the Repository.

For example: A COLUMN entry should NOT be inserted unless the TABLE entry to which it belongs already exists in the Repository. If the TABLE entry does not yet exist, the TABLE entry must be inserted first, then the COLUMN entry can be inserted identifying the TABLE to which it belongs.

All attributes for any new entry should be filled in except for the Expiration Date and Expired By Name. This information is supplied when the item is to be deleted/expired (see 3.3 DELETE (EXPIRE) Procedures).

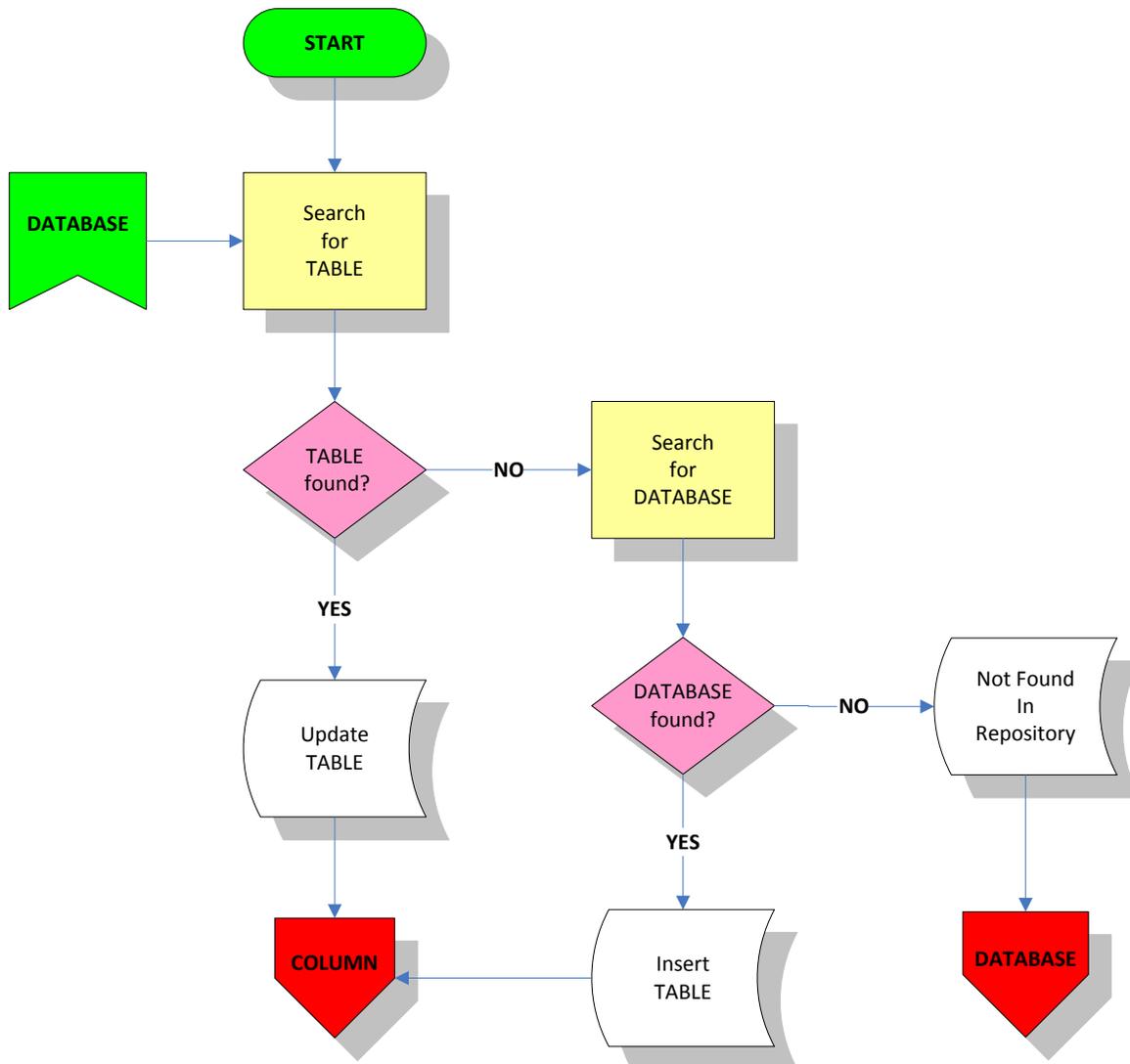
Updates are contingent on the item to be changed already exists in the Repository. If it is not, then the process for INSERTING that particular item needs to be followed.

Step 1 Insert/Update Database



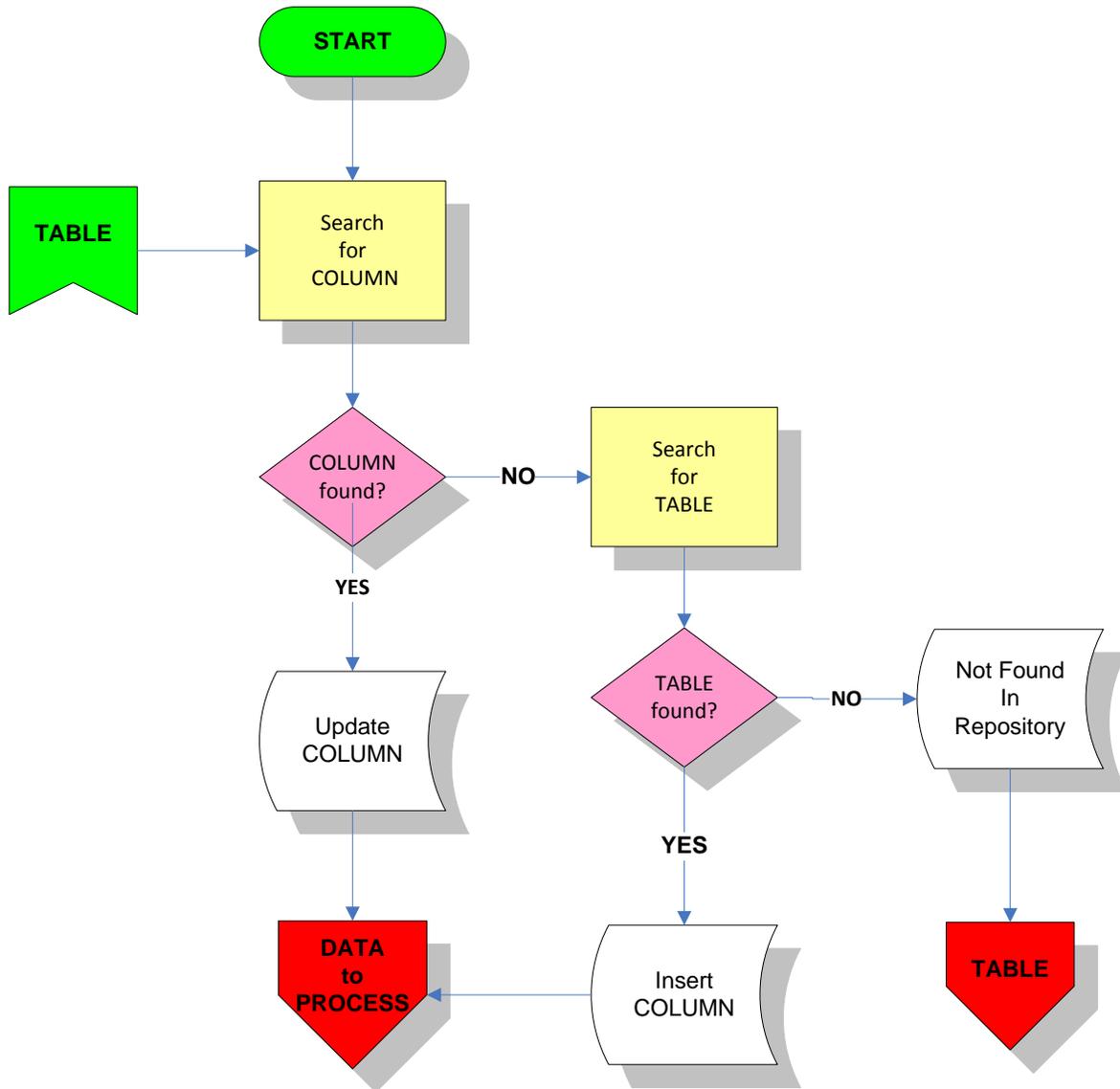
Process Diagram E– Insert/Update a DATABASE

Step 2 Insert/Update Table



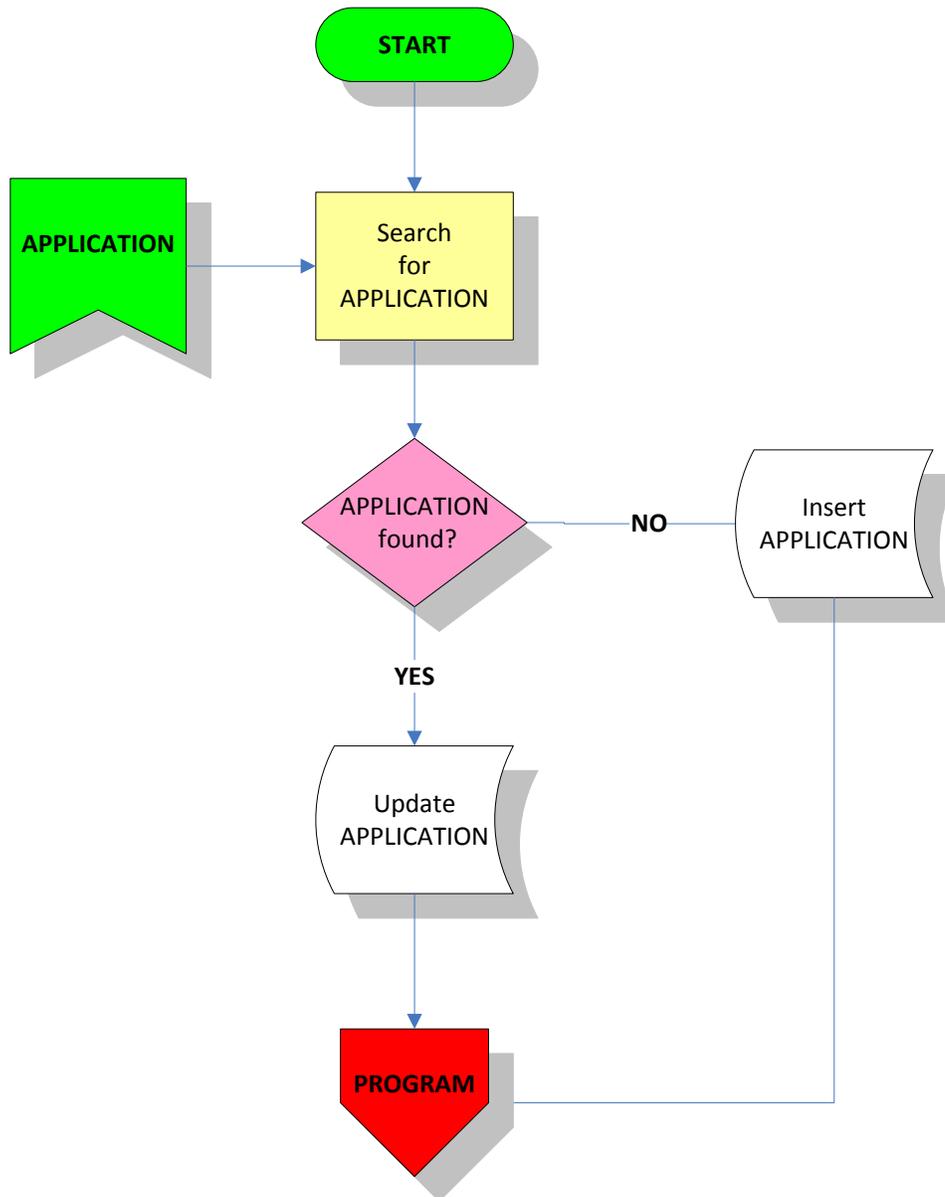
Process Diagram F– Insert/Update a TABLE

Step 3 Insert/Update Column



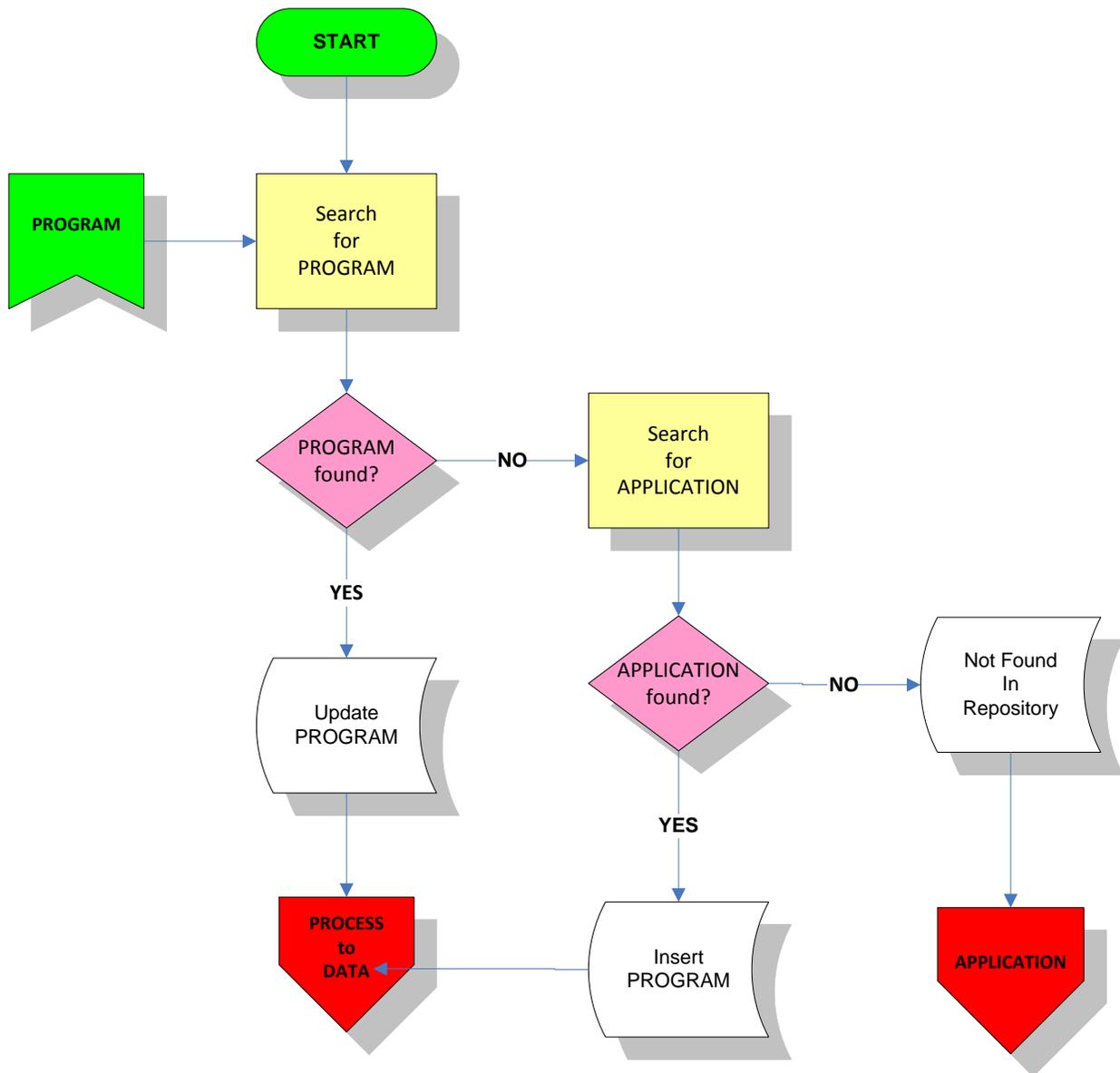
Process Diagram G– Insert/Update a COLUMN

Step 4 Insert/Update Application



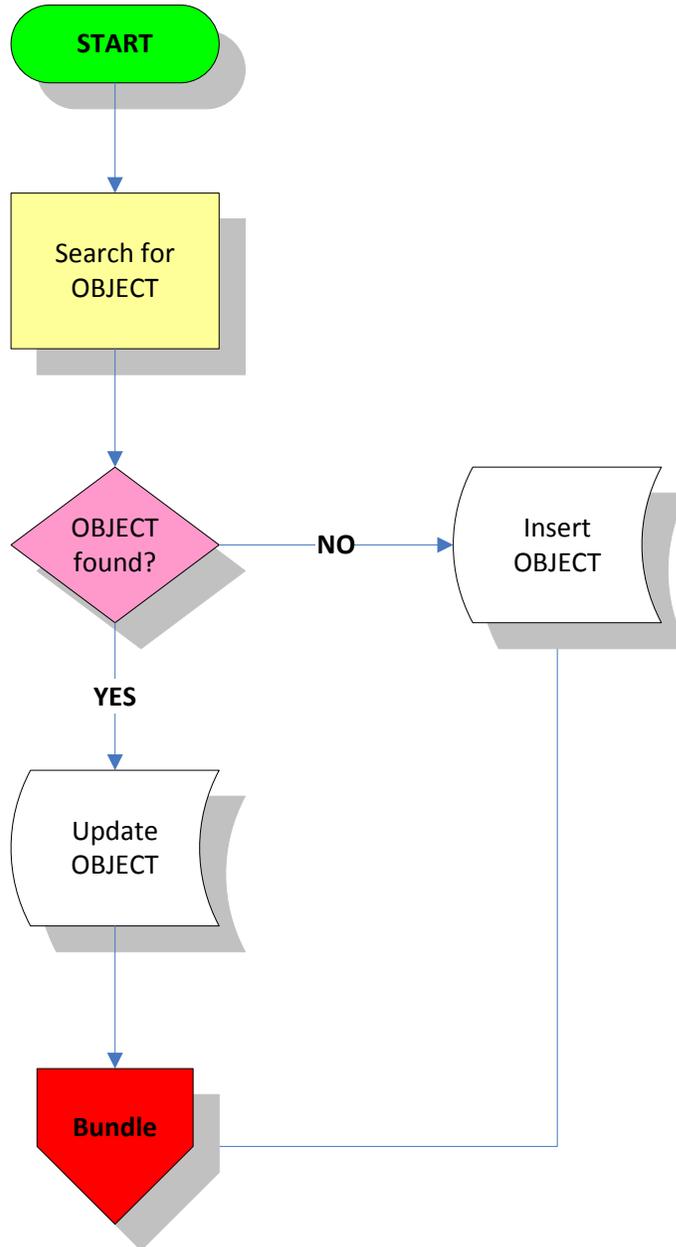
Process Diagram H– Insert/Update an APPLICATION

Step 5 Insert/Update Program



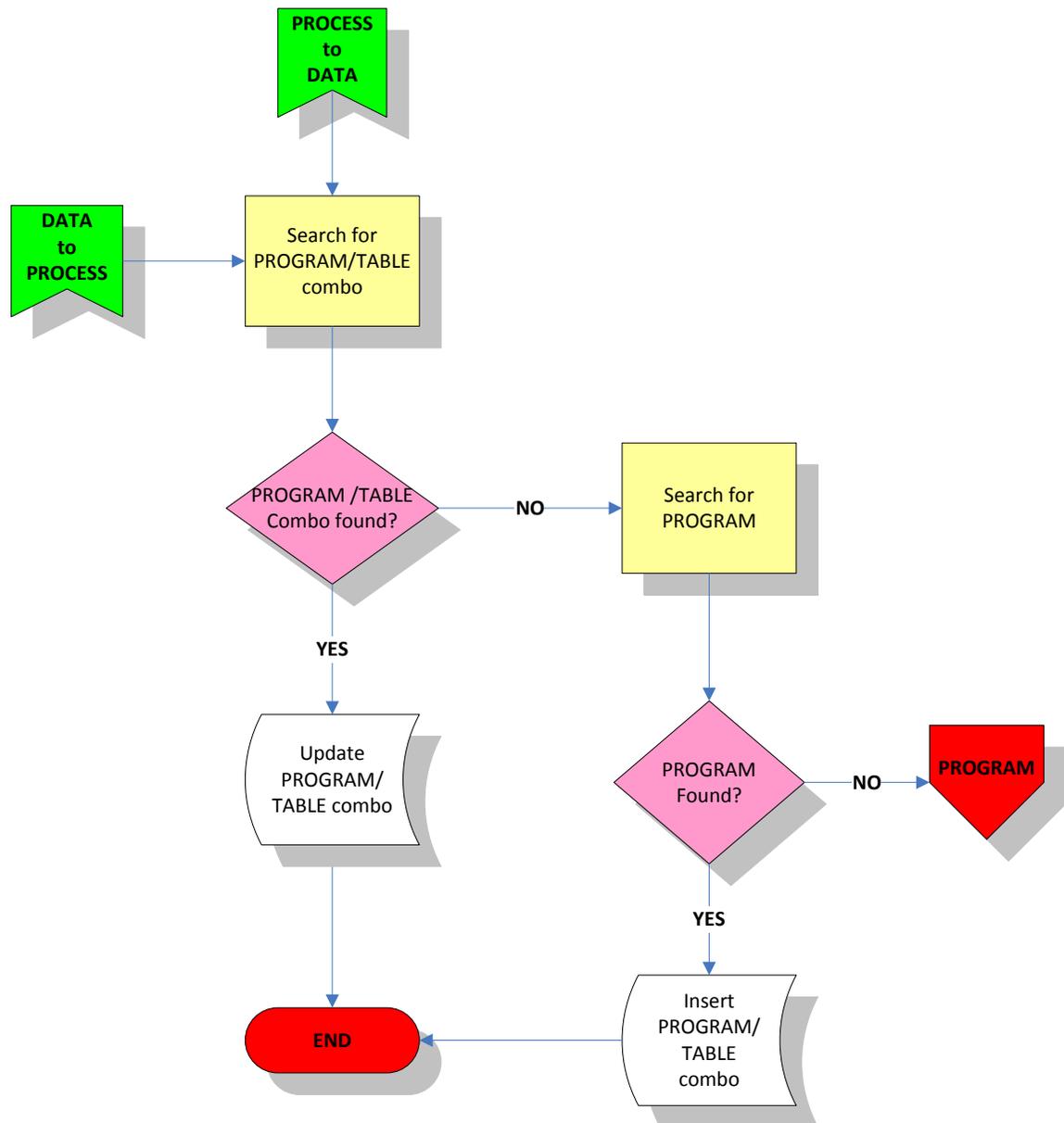
Process Diagram I– Insert/Update a PROGRAM

Step 6 Insert/Update Object



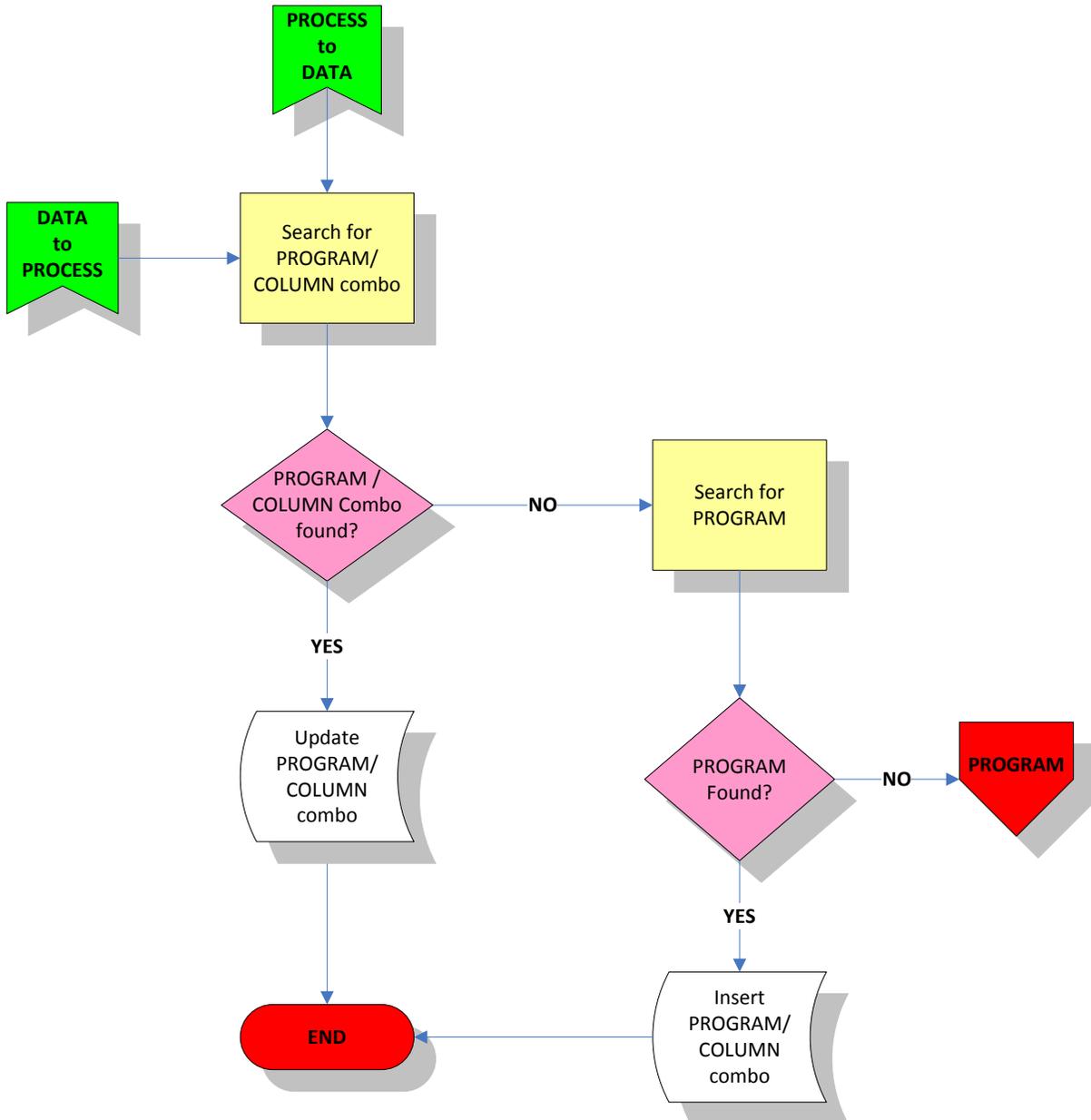
Process Diagram J – Insert/Update an OBJECT

Step 7 Insert/Update Program/Table relationship



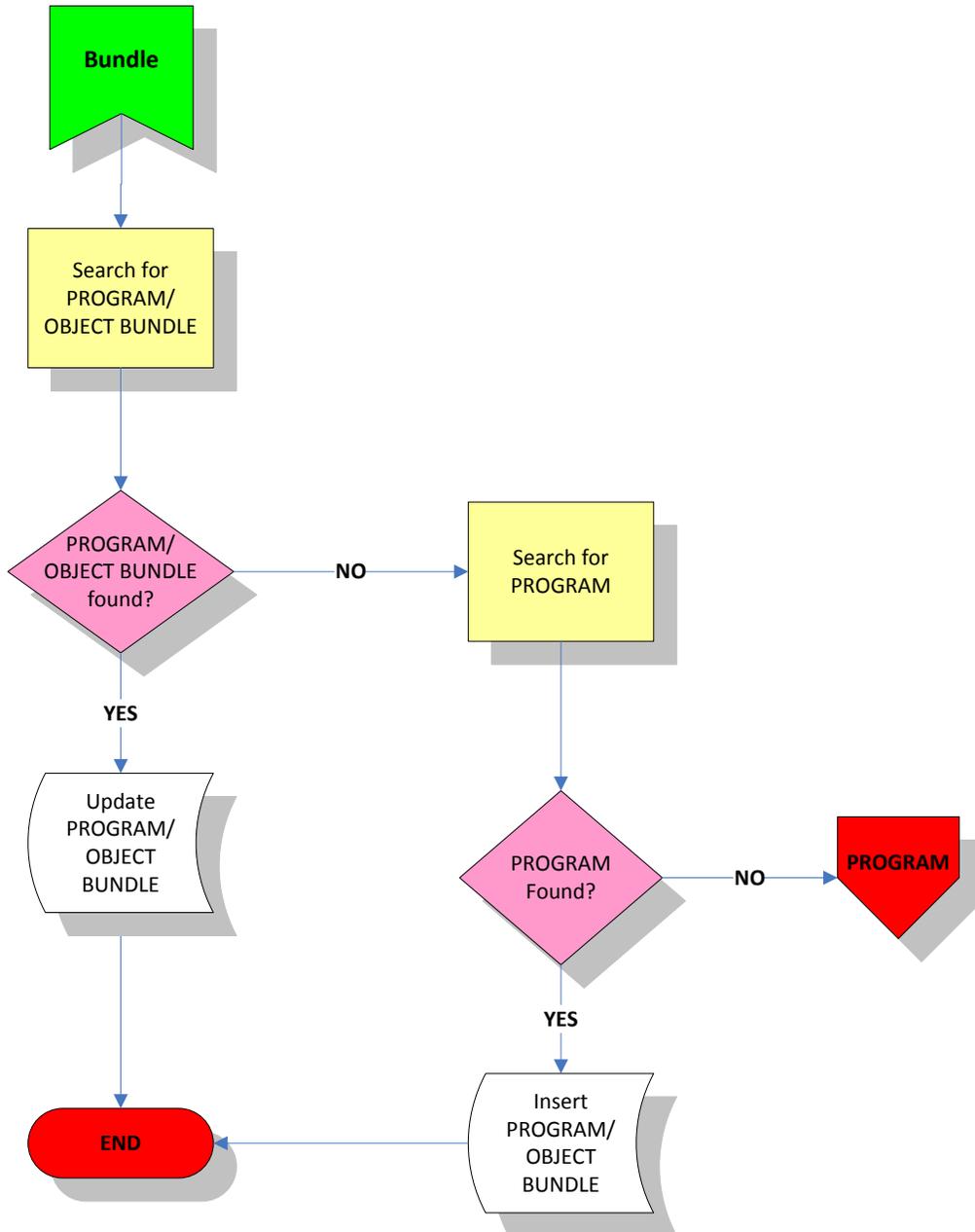
Process Diagram K– Insert/Update a PROGRAM/TABLE relationship

Step 8 Insert/Update Program/Column relationship



Process Diagram L– Insert/Update a PROGRAM/COLUMN relationship

Step 9 Insert/Update Program/Object relationship



Process Diagram L – Insert/Update a PROGRAM/OBJECT relationship

3.3 DELETE (EXPIRE) Procedures

Deletions of information from the Repository will be handled in a two-step process. When it has been determined that an item can be deleted the item will be first expired by inserting an Expiration Date and Expired By Name (the name of the person who is updating repository information).

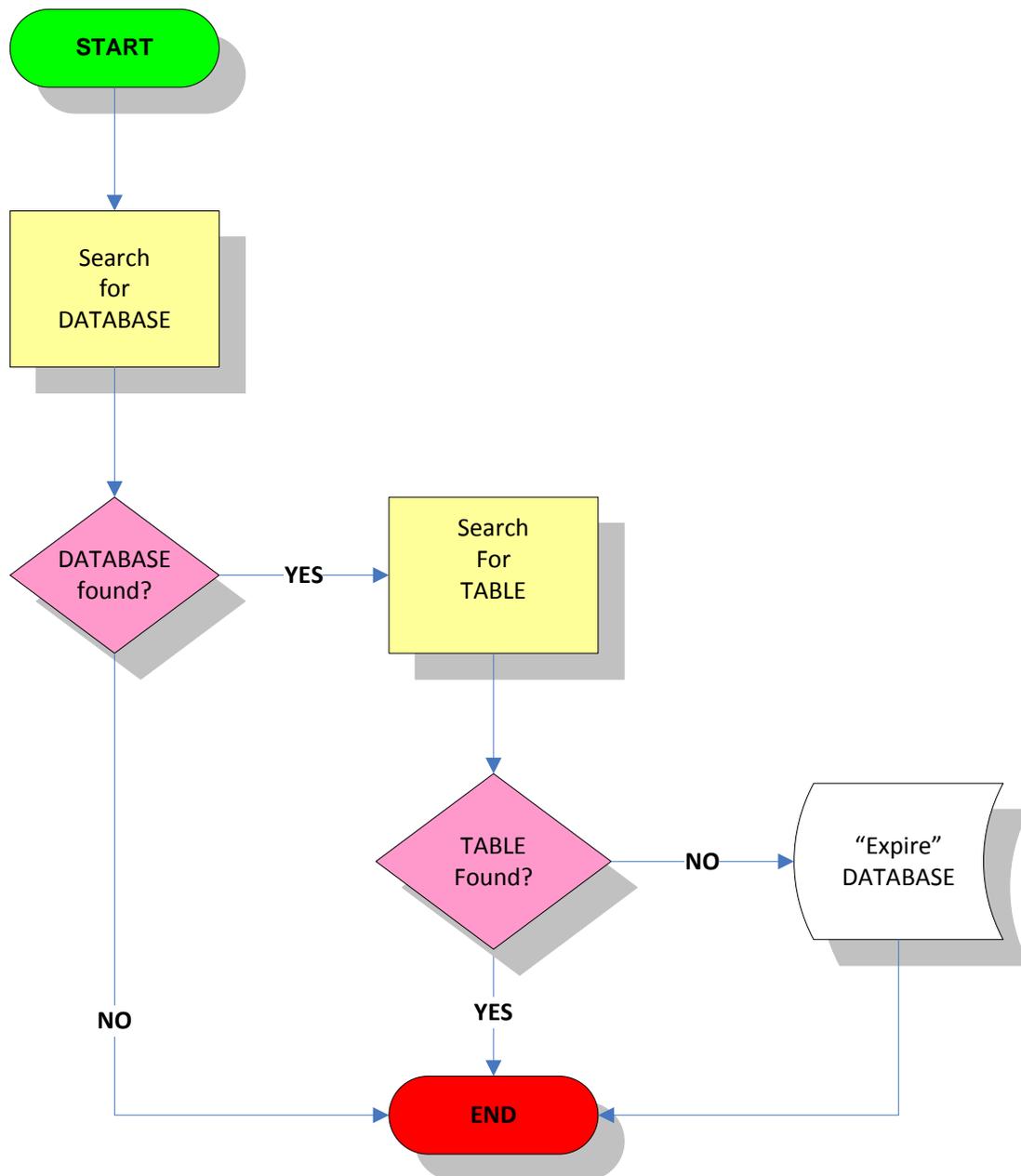
At regular intervals a “Repository Administrator” should comb through the Repository and “hard” delete any rows that have an expiration date older than what has been designated as a sufficient amount of time to have passed to delete the information in the interest of **keeping the repository “clean”** and uncluttered with obsolete information.

Understanding the relationship dependencies is critical here. The processes are defined in such a way as to help you to understand when an expiration/delete should be allowed and when it should not.

For example: Expiring/Deleting a TABLE should not happen (i.e. be allowed) as long as there are COLUMN entries still in the Repository related to the TABLE. The COLUMNS must all be expired/deleted FIRST before the TABLE can be expired/deleted.

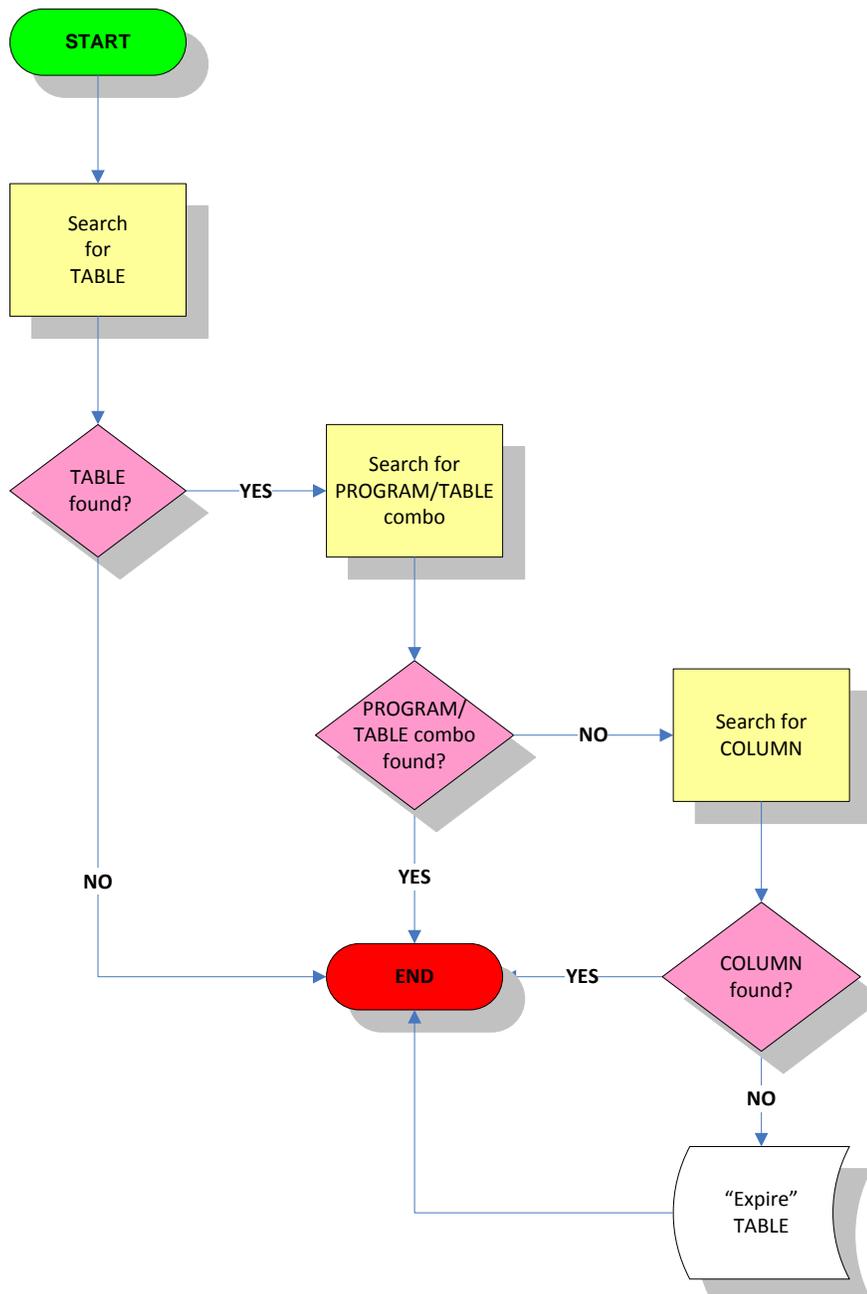
The following procedures are for adding the expiration information (i.e. updating) to the repository entries according to these rules of dependency.

Step 1 To expire a DATABASE



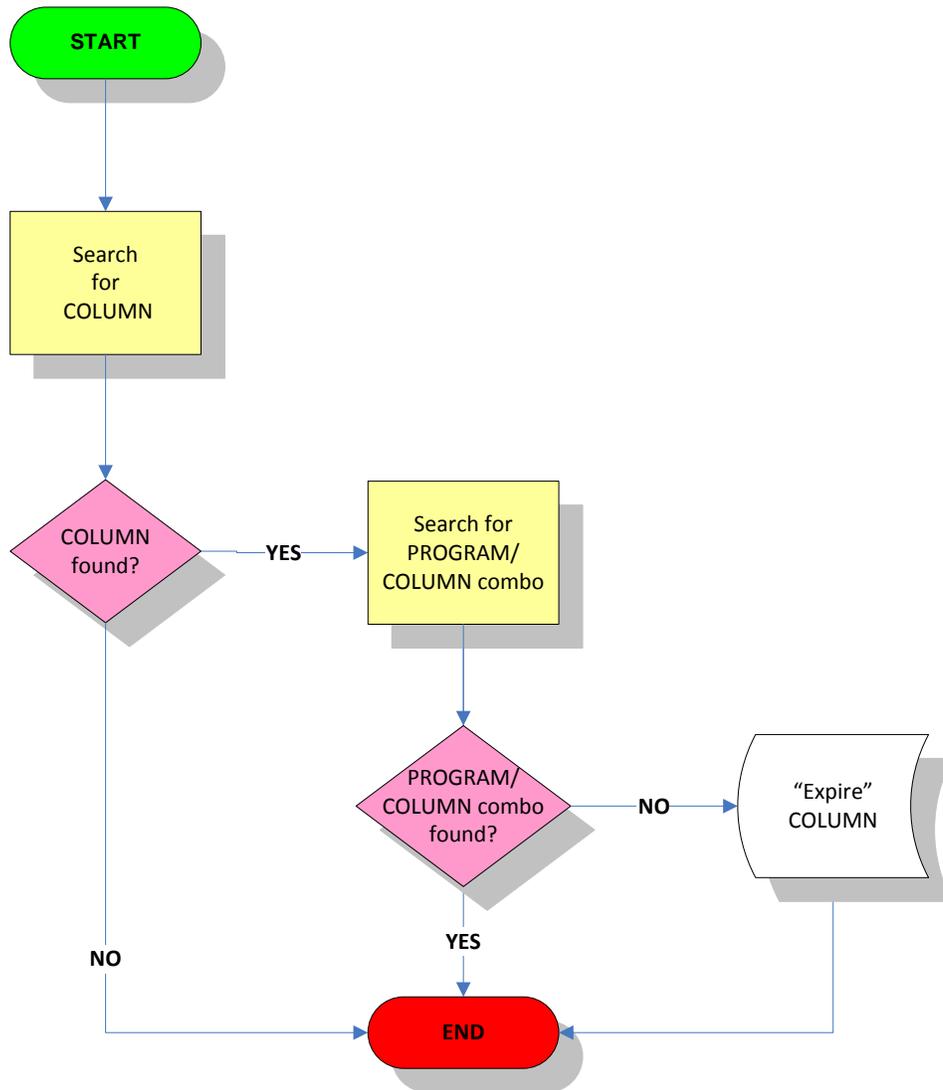
Process Diagram M – Expiring a DATABASE entry

Step 2 To expire a TABLE



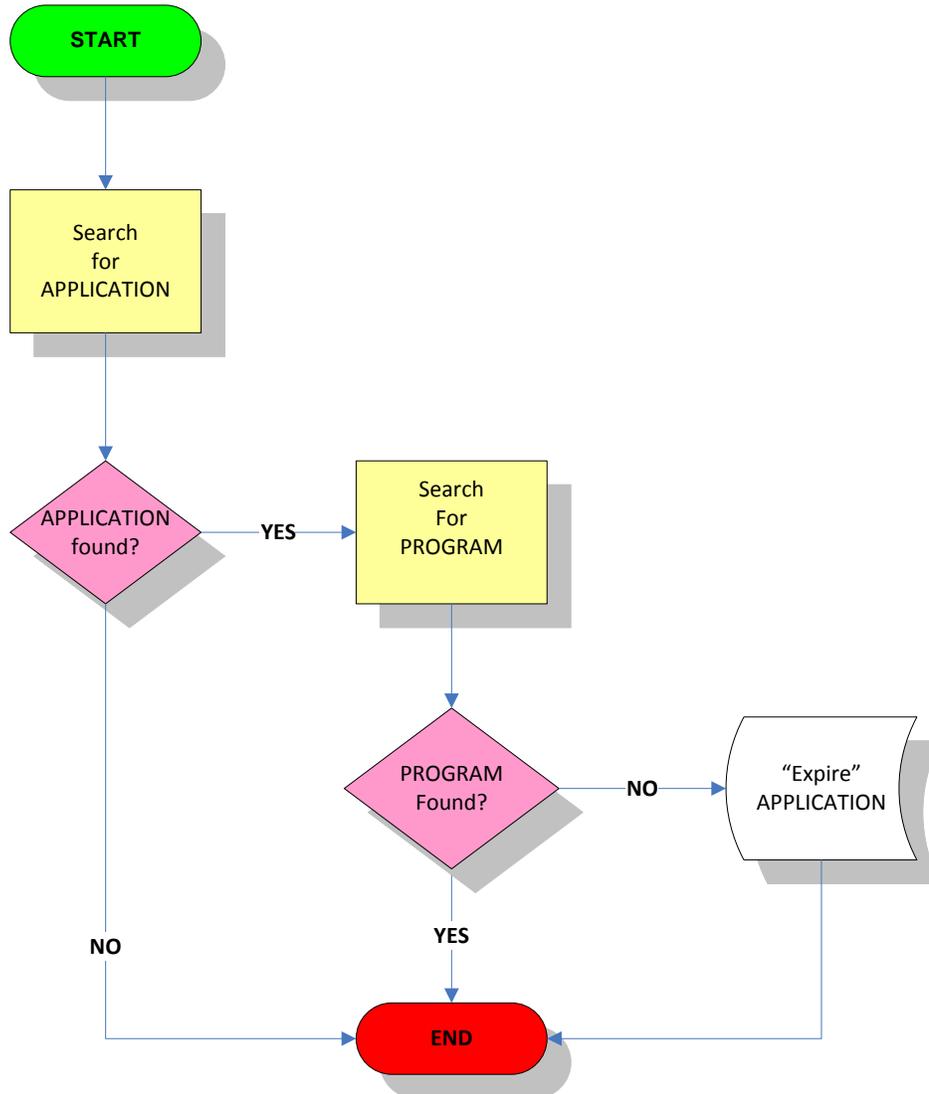
Process Diagram N – Expiring a TABLE entry

Step 3 To expire a Column



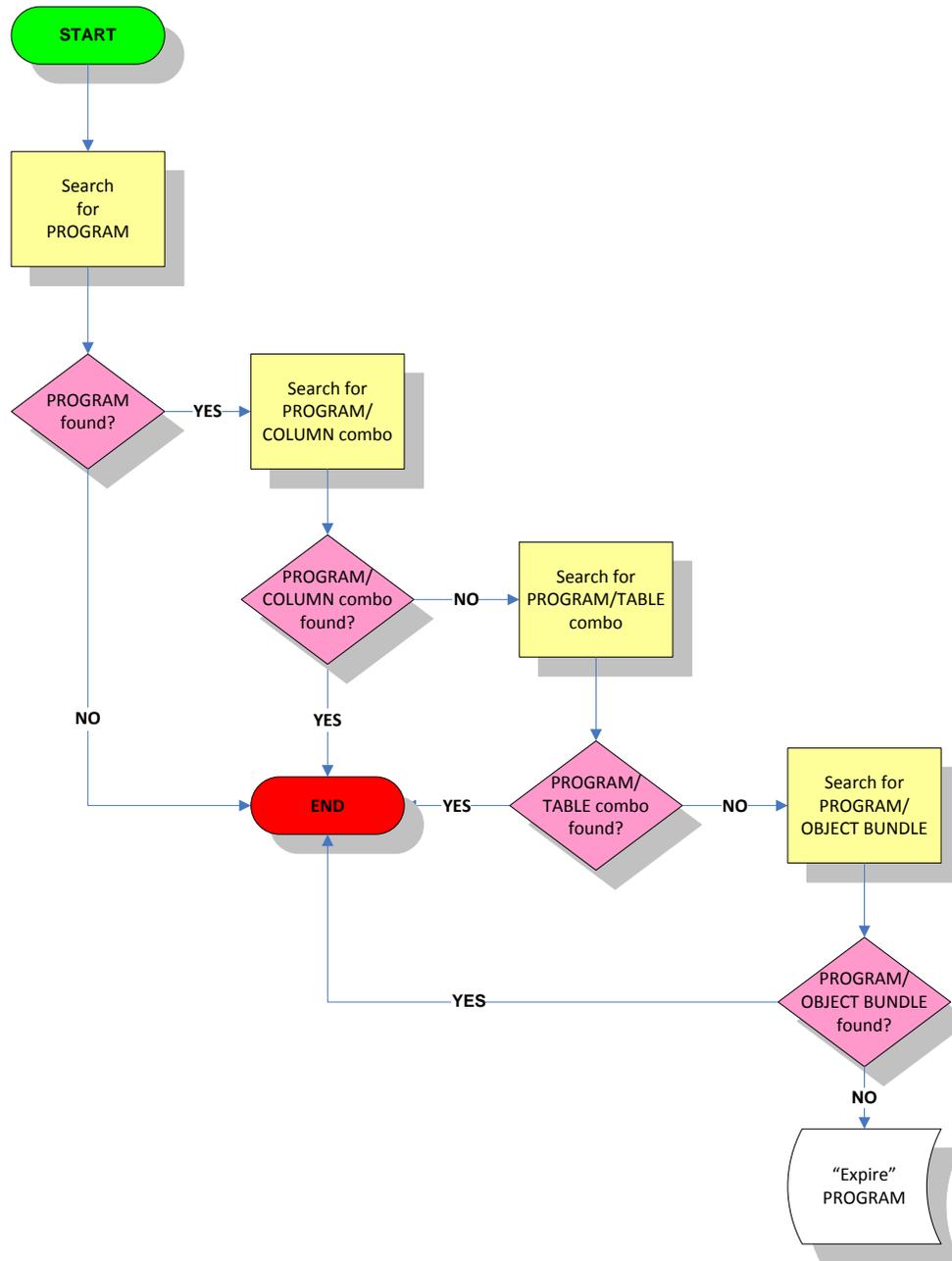
Process Diagram O – Expiring a COLUMN entry

Step 4 To expire an Application



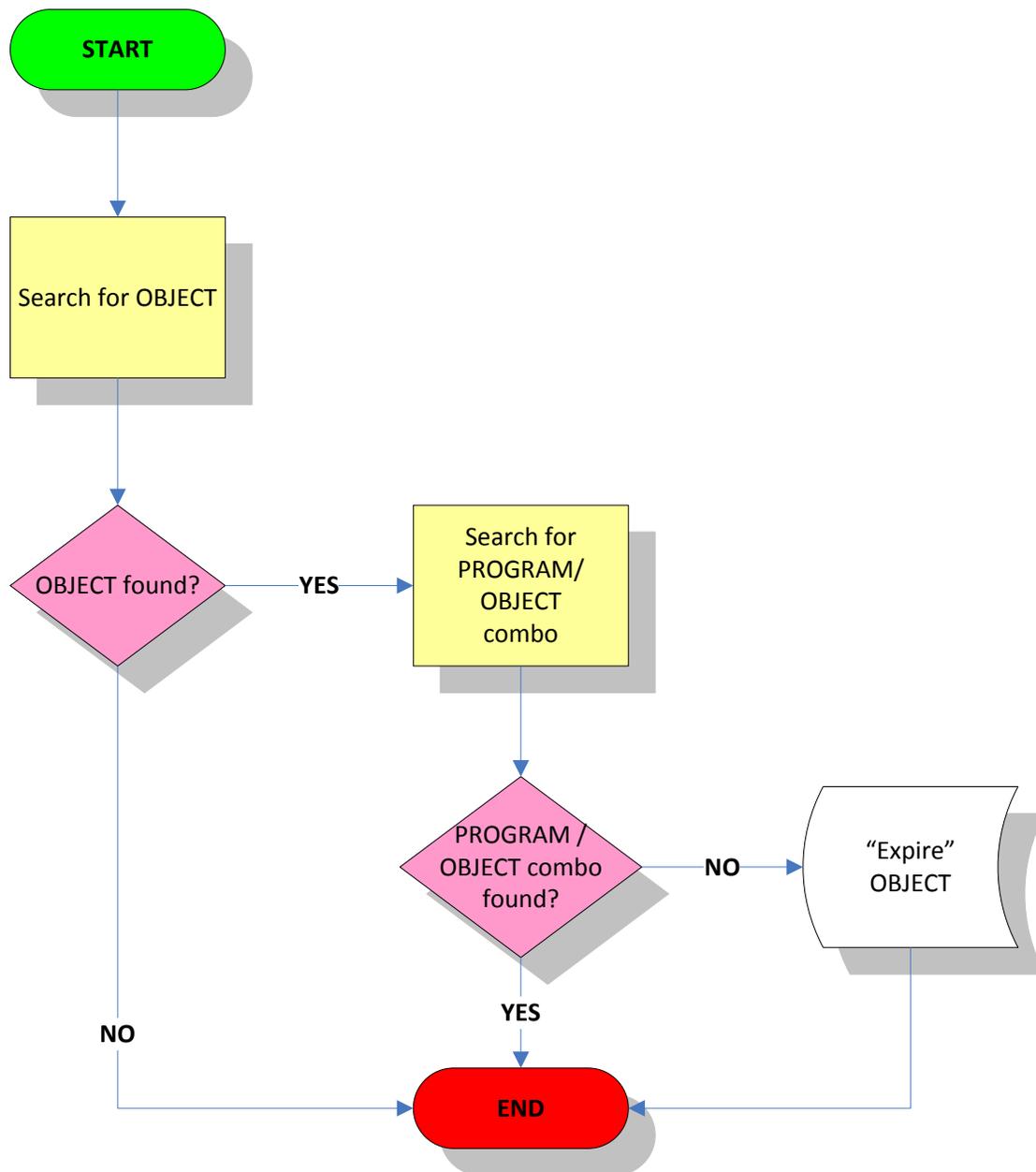
Process Diagram P – Expiring a APPLICATION entry

Step 5 To expire a Program



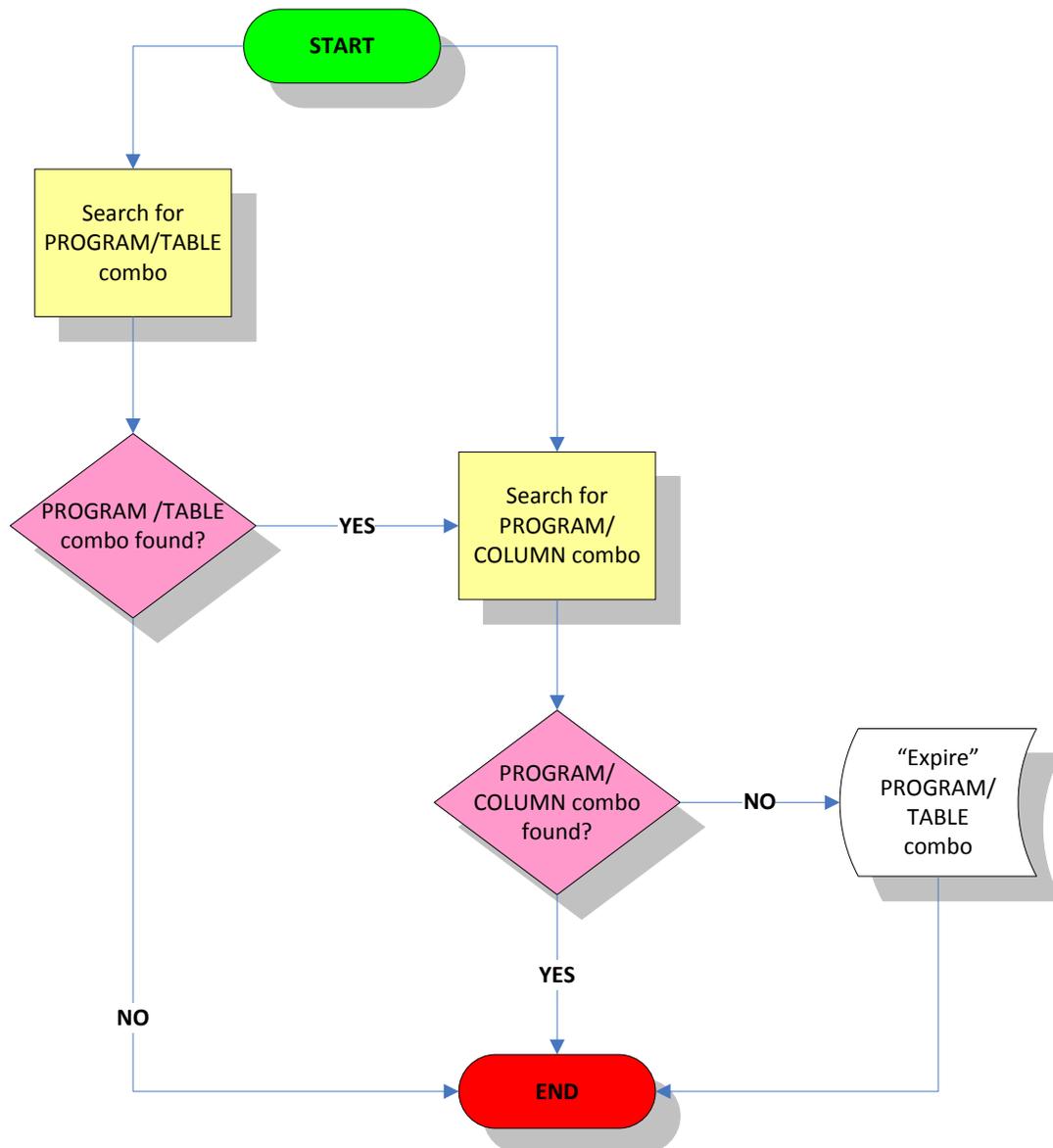
Process Diagram Q – Expiring a PROGRAM entry

Step 6 To expire an Object



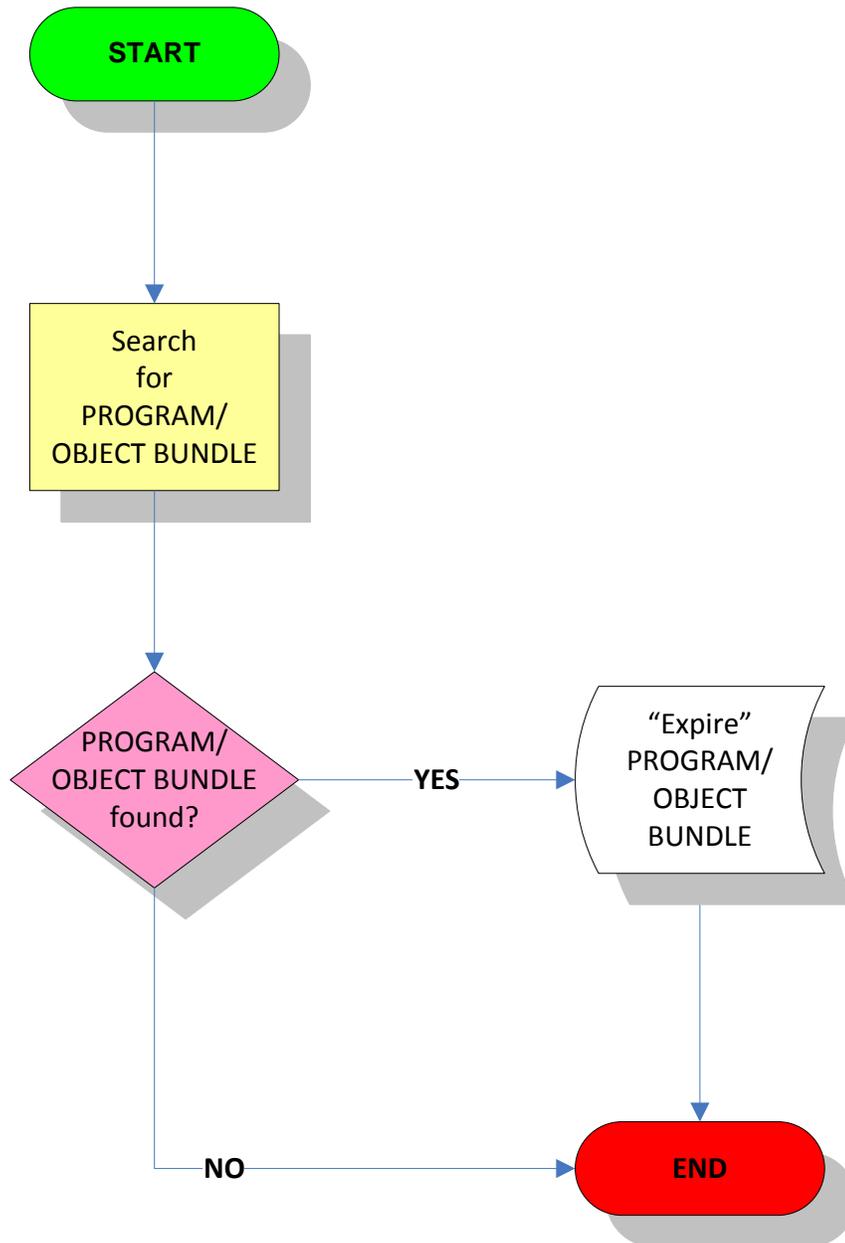
Process Diagram R – Expiring an OBJECT entry

Step 7 To expire a Data & Process relationship



Process Diagram S – Expiring a DATA and PROCESS relationship

Step 8 To expire a Program & Object relationship



Process Diagram T – Expiring a PROGRAM and OBJECT relationship

4. Option 3: Automated Process – Metadata Repository Tool

Choosing and implementing an enterprise-enabled Repository tool will eliminate not only the need for repeated, cumbersome manual processes, but also significantly reduce the administrative oversight required for the manual metadata management process. The automated environment that a repository tool would provide can effectively facilitate identifying and eliminating data redundancy, creating an enterprise (shareable) view of data assets, secure the integrity of those data assets, as well as greatly reduce the application and database development / maintenance cycles, and more.

Currently, getting a “lay of the land” understanding of the entire database environment (even within a single Agency) is nearly impossible. Database Administrators and Database Developers are finding it increasingly difficult to manage the database environment without a blueprint or roadmap to understand important object dependencies. It is becoming unmanageable and critical.

A repository tool offers crucial and cost-saving capabilities starting with the ability to reverse engineer all existing databases and thereby provide database administrators (DBAs) and/or Developers with important ‘physical’ data models in seconds instead of months! These models can be used as powerful and efficient change management platforms, allowing users to update a model with the required changes which need to be implemented at the database level and automatically generate DBMS specific, syntactically correct, alterations or database DDL.

Productivity gains and enforcement of organizational standards are achieved with strong collaboration capabilities offered by a Repository tool. A browser-based Repository provides easy access to metadata information stored in the Repository, enabling everyone to easily search, browse, and report on the information contained there, as well as provide powerful reporting and metadata exchange mechanisms throughout the enterprise.

All the many steps painstakingly described in Option 2 of this document would literally reduce most database and many application development efforts down from days or weeks to minutes or seconds. The integrity of the information would be enforced and maintained by the tool and the functionality it provides would greatly reduce the amount of metadata administration activities required. A sampling of these functions and the ease to perform are described in the following.

4.1 SEARCH / REPORT

Different tools allow different types of search and reporting capability. This will allow listing of metadata, object whereabouts and data usage stored in the metadata tool repository. In the case of ER Studio, there are a number of options to perform this task.

1. **ER Studio Data Architect** can be used to generate a fully navigable HTML report with dynamic data model images and detailed reports.

Steps: Tools -> Generate Reports -> Report Type (eg. HTML) -> Select Objects, Data Lineage, Data Dictionary, etc. -> Generate Images for sub-models -> Image Type/zoom/quality -> Put Corporate Logo / Links -> set other formatting options -> finish

Users can also write custom script to create and export their own reports (e.g. into Excel)

2. **ER Studio Portal**

Portal allows users to browse, search and report upon models that exist in the repository.

- Drill through explore tree
- Quick preview OR drill through models and sub models
- Drill through further detail of each entity (definition, attributes, where located in the repository, etc.)
- Business Architect Models allow documentation and exploration of conceptual models, processes and details of each diagram
- Search across entire portal (data models, process models, and conceptual models), apply filters
- Run canned Reports or use reporting wizard to generate custom reports
- Add report labels and description for further reference, classification, etc.

4.2 CREATE

Reverse Engineer

Overview: This approach allows automated documentation of existing database, data model (e.g. CA ERwin) or data transformation jobs (e.g. Informatica PowerCenter job).

Requirements:

- Industry standard metadata documentation tool
- Connectivity to desired database, files, etc.

Steps:

Actual steps provided are specific to ER Studio however most tools use very similar approach for reverse engineering an existing database:

1. Create new file or project
2. Specify target repository
3. Choose reverse engineering or import option
4. Specify source system/database and connection credentials
5. Choose types and objects to reverse engineer
6. Perform reverse engineering
7. Review results, and append additional / custom metadata
8. Save
9. Exit

ER Studio Steps:

1. File -> New -> Reverse Engineering -> Select Database Type (e.g. Oracle, SQL Server, Sybase, ODBC, etc.)
2. Provide Login Information
3. Select what types of objects, databases and owners that need to be reverse engineers
4. Based on database types, different object types will be available for reverse engineering
5. Select which tables, decided on further options (e.g. infer PKs that do not exist, create/define domains, choose a layout (can be changed later))
6. Review summary and click Finish.

This will create a logical and physical database model.

Create New Model

This option is used when a new model is being created and is very similar to other tools such as CA ERwin. Typical actions include:

1. Create Logical Data Model
2. Create Physical Data Model

4.3 CHANGE MANAGEMENT

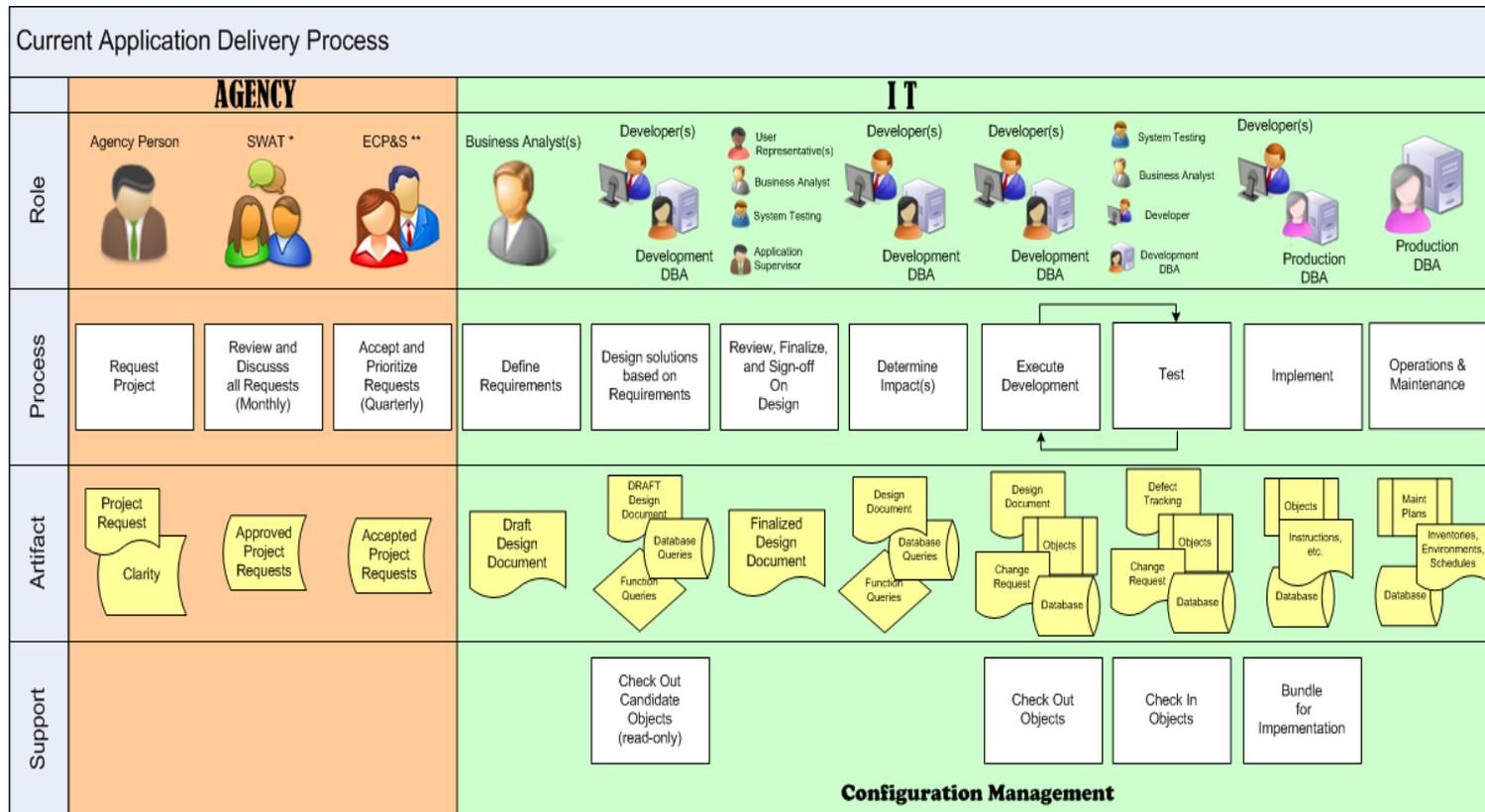
- Compare / Merge Utility
 - o Allows comparing models against database
 - o Review changes
 - o Implement changes to database (tool creates DDL change script)
 - o OR rollback changes from data model
 - o Comparisons include configuration, data model, data content, etc.
- Snapshots of database can be taken and stored at any point in time and used later
- Teams can work on data models by checking out / checking in using built in version control feature.

4.4 OTHER

- Integrate Tool (e.g. ER Studio) into existing enterprise applications such as Microsoft SharePoint Server
- Build custom automation scripts that feed manual and routine processes (e.g. database structure scan reports, current-state snapshots, etc.)

This page left intentionally blank.

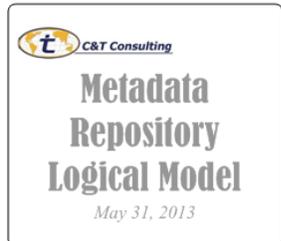
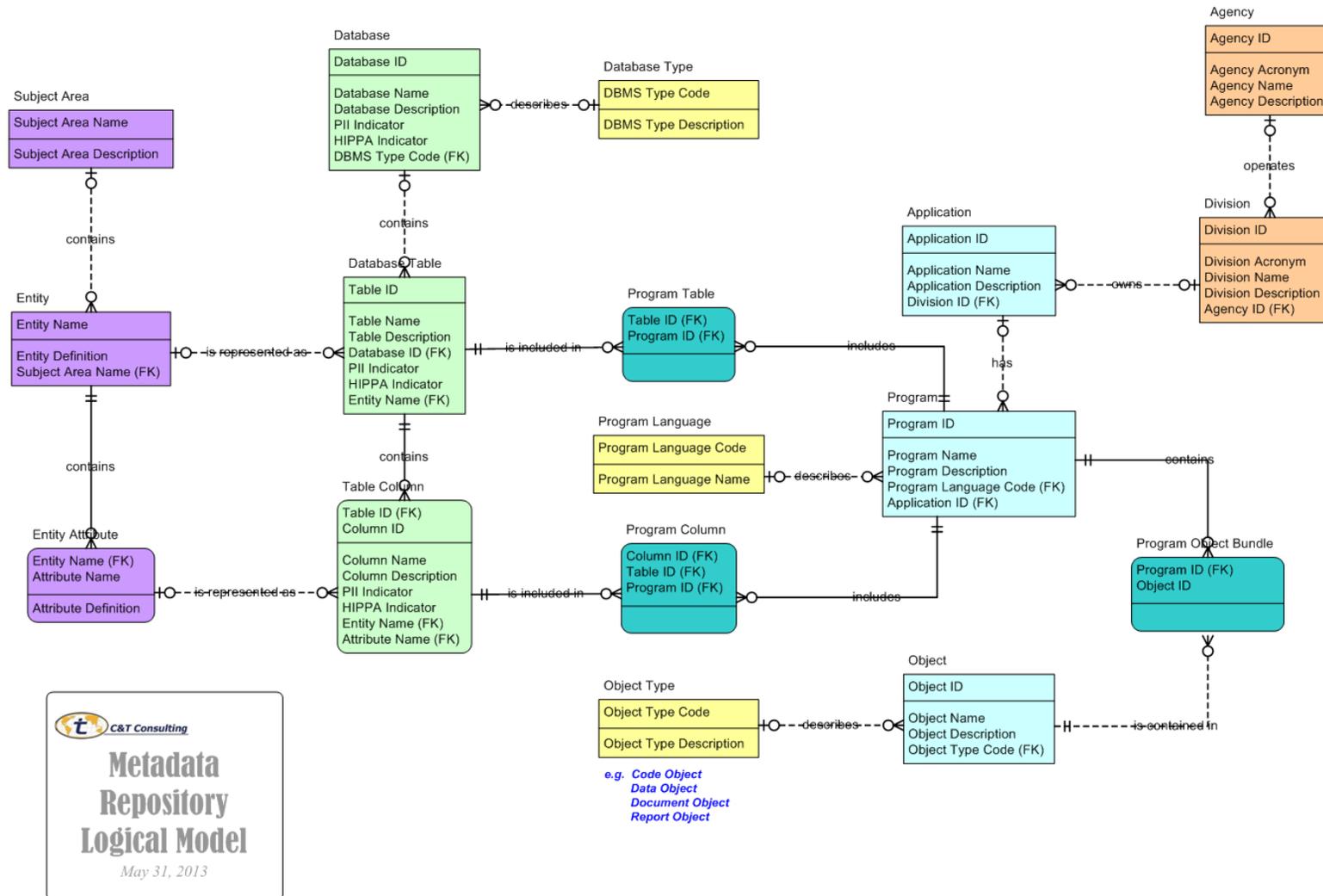
Appendix A – Current Application Delivery Process Diagram



* SWAT = System-Wide Analysis Team

** ECP&S = Executive Committee Prioritization & Selection

Appendix B – Metadata Repository Logical Model (LDM)



Appendix C – Metadata Repository Template Glossary

Entity	Entity Description	Entity Attribute	Entity Attribute Description
Agency	A permanent or semi-permanent organization within the state government that is responsible for the oversight and administration of specific functions, such as a human services agency. Agencies can be established by legislation or by executive powers.	Agency ID	The unique identifier for the Agency.
		Agency Acronym	The recognized acronym or initials associated with the Agency. i.e. Short Name.
		Agency Name	The descriptive name or title of the Agency.
		Agency Description	The text that describes the Agency and its charter.
Application	An application is a program or group of programs designed for end users.	Application ID	The unique identifier for an Application or System.
		Application Name	The descriptive name or title of the Application.
		Application Description	The text that describes the application and what it is about.
		Division ID	Identifies the Division that has ownership and responsibility for the Application.
Database	A database is an organized collection of data. A database is not generally portable across different DBMS, but different DBMSs can inter-operate by using standards such as SQL and ODBC or JDBC to allow a single application	Database ID	The unique identifier for the Database.
		Database Name	The descriptive name or title of the Database.

Entity	Entity Description	Entity Attribute	Entity Attribute Description
	to work with more than one database.	Database Description	The text that describes the Database and the general information that it contains.
Database Table	In relational databases and flat file databases, a table is a set of data elements (values) that is organized using a model of vertical columns (which are identified by their name) and horizontal rows, the cell being the unit where a row and column intersect.[1] A table has a specified number of columns, but can have any number of rows[citation needed]. Each row is identified by the values appearing in a particular column subset which has been identified as a unique key index.	Table ID	The unique identifier for the Table.
		Table Name	The descriptive name or title of the Table.
		Table Description	The text that describes the Table and the information that it contains.
		Database ID	Identifies the database to which this Table belongs.
Database Type	A specific database platform a.k.a. database management system (DBMS) (e.g. MySQL, Microsoft SQL Server, Microsoft Access, Oracle, SAP, DB2).	Database Type Code	The code that uniquely identifies a type of Database Management System (DBMS).
		Database Type Description	The text that describes a type of Database Management System (DBMS).
Division	A subdivision of an a state agency set up to address a specialized set of services and/or functions. Example: Child Welfare is a division of the Colorado Department of Human Services.	Division ID	The unique identifier for the Division.
		Division Acronym	The recognized acronym or initials associated with the Division. i.e. Short Name.
		Division Name	The descriptive name or title of the Division.

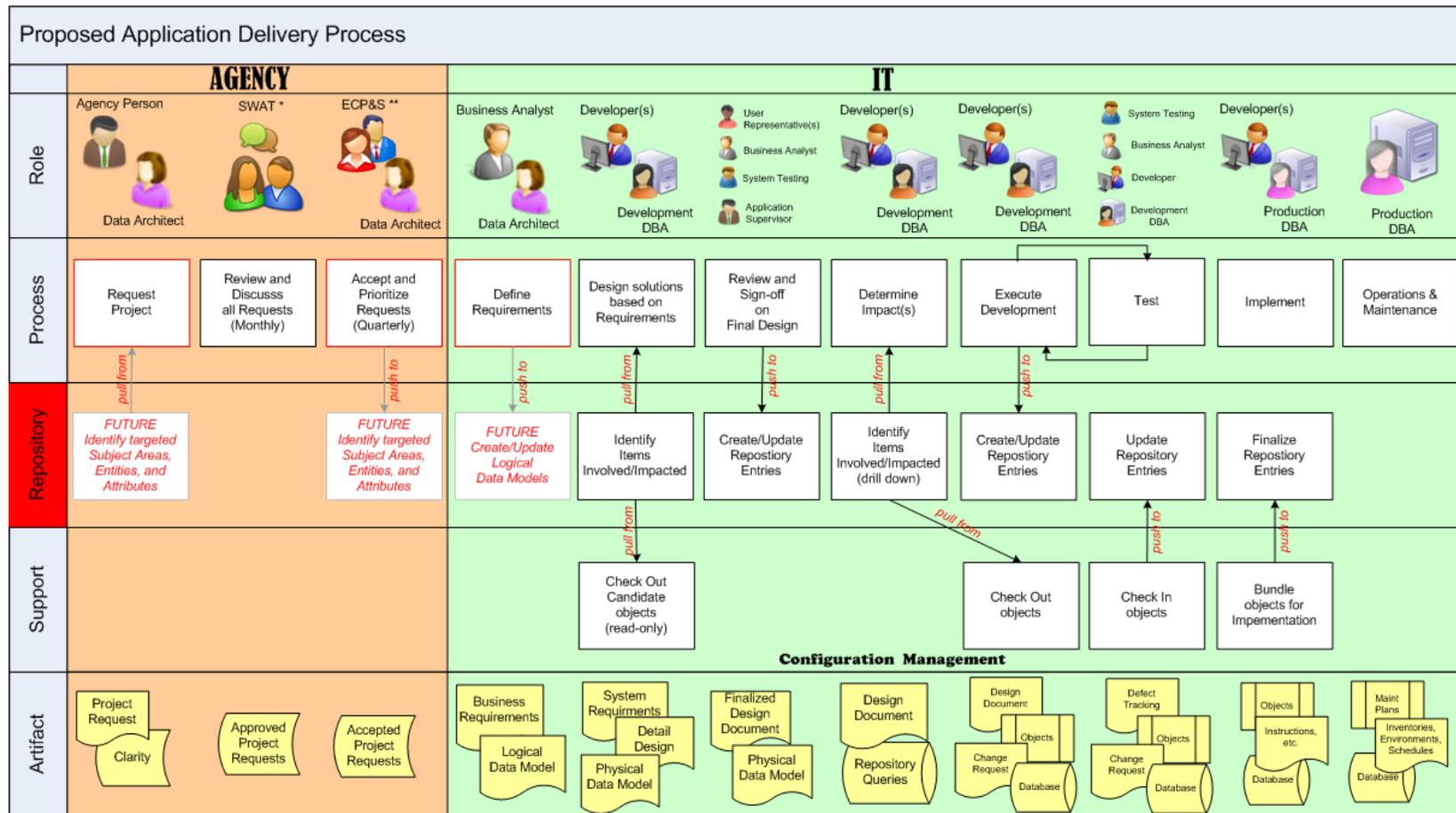
Entity	Entity Description	Entity Attribute	Entity Attribute Description
		Division Description	The text that describes the Division and its charter.
		Agency ID	Identifies the Agency of which the Division is a component.
Entity	A logical, business-oriented person, place, or thing that is uniquely identifiable and about which there are specific attributes. (see ENTITY ATTRIBUTE).	Entity Name	The name that uniquely identifies and names this Entity.
		Entity Definition	The dictionary-like definition of this Entity.
		Subject Area Name	The name that uniquely identifies the Subject Area to which this Entity belongs.
Entity Attribute	A logical, business-oriented piece of descriptive information about an Entity. (see ENTITY).	Entity Name	The name that uniquely identifies and names Entity to which this Attribute belongs.
		Attribute Name	The name that identifies and names this Attribute
		Attribute Definition	The dictionary-like definition of this Attribute.
Object	A uniquely identifiable and reusable component of a computer program. Object can be grouped together or bundled to create unique program instances for implementation.	Object ID	The unique identifier of an Object.
		Object Name	The descriptive name of an Object.
		Object Description	The text that describes an Object.

Entity	Entity Description	Entity Attribute	Entity Attribute Description
		Object Type Code	The code that identifies what type of Object this is.
Object Type	Describes a type of object or component that is part of the make-up of a Program.	Object Type Code	The code that uniquely identifies a type of Object.
		Object Type Description	The text that describes a type of Object.
Program	A computer program, or just a program, is a sequence of instructions, written to perform a specified task with a computer.[1] A computer requires programs to function, typically executing the program's instructions in a central processor.[2] The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled), enables a programmer to study and develop its algorithms. A collection of computer programs and related data is referred to as the software.	Program ID	The unique identifier for the Program.
		Program Name	The descriptive name or title of the Program.
		Program Description	The text that describes and/or names the Program and what it is about.
		Application ID	Identifies the Application of which this Program is a component.
Program Column	This identifies and describes a Database Table Column and its relationship to a specific Program. (See TABLE COLUMN and PROGRAM).	Program ID	Identifies the Program in which this Column is used.
		Table ID	Identifies the Table to which the Column utilized by this Program belongs.
		Column ID	Identifies the Column utilized by this Program.
		Input Indicator	Indicates if this Column is included as input to this

Entity	Entity Description	Entity Attribute	Entity Attribute Description
			Program.
		Output Indicator	Indicates if this Column is produced as output from this Program.
Program Language	The machine language in which a computer program can be written.	Program Language Code	The code that uniquely identifies a programming language.
		Program Language Description	The text that describes and/or names a programming language.
Program Object Bundle	A grouping of objects that define or make-up a computer program and or an implementation package of a computer program. (see also PROGRAM and OBJECT).	Program ID	Identifies the Program for which these Objects are bundled or grouped.
		Object ID	Identifies an Object that is included in this bundle or grouping.
Program Table	This identifies and describes a Database Table and its relationship to a specific Program. (See DATABASE TABLE and PROGRAM).	Program ID	Identifies the Program in which this Table is used.
		Table ID	Identifies the Table utilized by this Program belongs.
		Input Indicator	Indicates if this Table is included as input to this Program.
		Output Indicator	Indicates if this Table is produced as output from this Program.
Subject Area	A high-level, business-oriented, and conceptual designation that identifies and describes a key	Subject Area Name	The name that uniquely

Entity	Entity Description	Entity Attribute	Entity Attribute Description
	concept or area of interest to the State to be addressed and managed. (e.g. Citizens, Benefit Programs, Locations, etc.)		identifies a Subject Area.
		Subject Area Description	The text that describes a Subject Area.
Table Column	In the context of a relational database table, a column is a set of data values of a particular simple type, one for each row of the table.[1] The columns provide the structure according to which the rows are composed. The term field is often used interchangeably with column, although many consider it more correct to use field (or field value) to refer specifically to the single item that exists at the intersection between one row and one column. In relational database terminology, column's equivalent is called attribute.	Table ID	Identifies the Table to which this Column belongs.
		Column ID	The unique identifier for the Column.
		Column Name	The descriptive name or title of the Column.
		Column Description	The text that describes the Column; the definition of the Column.

Appendix D – Proposed Application Delivery Process Diagram



* SWAT = System-Wide Analysis Team ** ECP&S = Exective Committee Prioritization & Selection