Automating Data Collection and Location Tagging in Pansophy

Nicolas Crespo

High School Summer Honors Program

Thomas Jefferson National Accelerator Facility

Prepared in partial fulfillment of the requirements of the Department of Energy's Office of Science and Thomas Jefferson Accelerator Facility under the direction of Valerie Bookwalter in the SRFOPS division at Thomas Jefferson National Accelerator Facility.

Participant: ______________________
Signature

Mentor: ______________________
Signature

Abstract

Pansophy is an inventory management system utilized by the Superconducting Radio Frequency (SRF) Department at Jefferson Lab. Superconducting niobium cavities are essential to the near light speed movement and near zero energy loss that occurs in the linear sections of Jefferson Lab's particle accelerator. However, due to the specialized nature of the SRF technology, all SRF cavities are designed, built and tested in house. Though this allows Jefferson Lab to provide SRF cavities to particle accelerator facilities around the world, it also produces a challenge in inventory management. Pansophy, Jefferson Lab's internal inventory management system, allows for the manual tracking of parts as they move through the construction cycle with special reports called Travelers to be filled out by engineers. Although all data is securely stored in a collection of relational databases, invalid entries, human error and typos can lead to unexpected behavior and reduced efficiency in tracking parts. As such, an automated system has been created utilizing barcode scanners, which provide each item with a unique code, and location tracking cookies, which are stored on every engineer's computer. By properly interacting with Pansophy's databases, extracting the correct information and providing a friendly user interface, this system allows any engineer to directly access and create a Traveler with just one scan of a barcode.

Introduction

The superconducting radio frequency (SRF) cavities that line the linear particle accelerator at Jefferson Lab allow for near zero energy loss and enable particles to travel at close to the speed of light. However, the highly specialized nature of SRF technology means Jefferson Lab must design, construct and test SRF cavities in house. Though this allows the Lab to produce cavities for accelerator facilities around the world, it also presents an exceedingly difficult challenge in inventory and data management. Pansophy, an internal data management system is an all-encompassing solution to this problem.

Pansophy is an internal website that is only accessible inside of Jefferson Lab's firewall. Its front end is written in ColdFusion, JavaScript and CSS, with ColdFusion being used mainly due to its ability to seamlessly embed SQL (Structured Query Language) onto a webpage. ColdFusion is an HTML-like language with its own plethora of custom tags and a built-in scripting language, CFScript.

SQL is used to query the two main relation Oracle databases: the PRIMeS (Production and Research Inventory Management System) and Travelers. Each database houses a collection of tables each with a primary key and various foreign keys. By cross referencing the correct tables, relevant information about a part or traveler can be discerned even when provided limited information.

As parts make their way around the SRF Test Lab, they go through various stages, or actions, at different locations, or work centers. At each step, an engineer is tasked with creating a Traveler for the part they are dealing with. To streamline the creation of travelers into the Pansophy system and minimize errors due to typos, it is in Jefferson Lab's best interest to automate the entrance of part information as much as possible. As such, stickers containing unique barcodes (Figure 1) have been placed on most parts.

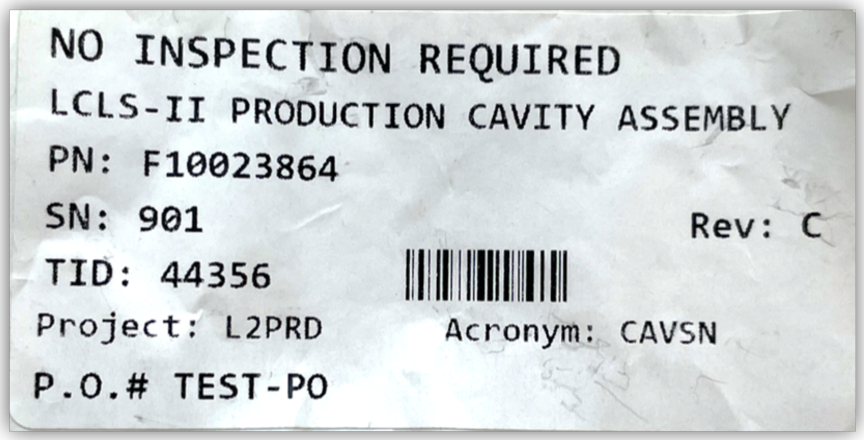
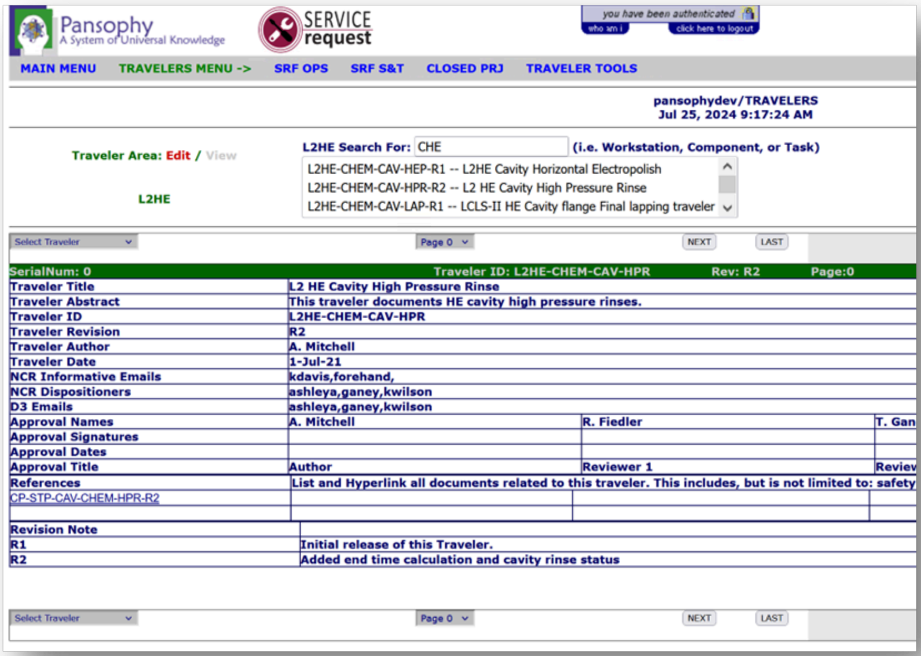

NO INSPECTION REQUIRED
LCLS-II PRODUCTION CAVITY ASSEMBLY
PN: F10023864
SN: 901 Rev: C
TID: 44356
Project: L2PRD Acronym: CAVSN
P.O.# TEST-PO

Figure 1: A sticker containing a barcode to be used in the lab.

These barcodes, which produce Transaction IDs, can provide various important characteristics of a part. Combined with location information stored in cookies that is to be pre-set on every engineer's machine, a proper implementation of this system will instantly redirect an engineer to a Traveler page (Figure 2) with just a scan of a barcode (Figure 3).



pansophydev/TRAVELERS
Jul 25, 2024 9:17:24 AM

Traveler Area: Edit / View

L2HE

L2HE Search For: CHE (i.e. Workstation, Component, or Task)

L2HE-CHEM-CAV-HEP-R1 -- L2HE Cavity Horizontal Electropolish
L2HE-CHEM-CAV-HPR-R2 -- L2 HE Cavity High Pressure Rinse
L2HE-CHEM-CAV-LAP-R1 -- LCLS-II HE Cavity flange Final lapping traveler

SerialNum: 0	Traveler ID: L2HE-CHEM-CAV-HPR	Rev: R2	Page:0
Traveler Title	L2 HE Cavity High Pressure Rinse		
Traveler Abstract	This traveler documents HE cavity high pressure rinses.		
Traveler ID	L2HE-CHEM-CAV-HPR		
Traveler Revision	R2		
Traveler Author	A. Mitchell		
Traveler Date	1-Jul-21		
NCR Informative Emails	kdavis,forehand,		
NCR Dispositioners	ashleya,ganey,kwilson		
D3 Emails	ashleya,ganey,kwilson		
Approval Names	A. Mitchell	R. Fiedler	T. Gan
Approval Signatures			
Approval Dates			
Approval Title	Author	Reviewer 1	Reviev
References	List and Hyperlink all documents related to this traveler. This includes, but is not limited to: safety		
CP-STP-CAV-CHEM-HPR-R2			

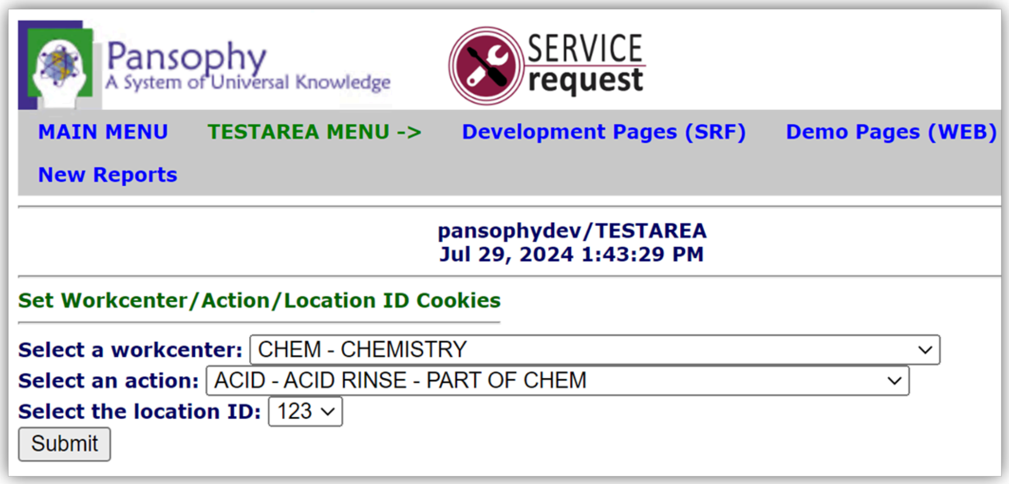
Revision Note	
R1	Initial release of this Traveler.
R2	Added end time calculation and cavity rinse status

Figure 2: A traveler page a user would reach after scanning a part.

Figure 3: A barcode scanner to be used in the lab.

Methodology

Before the scanning of a barcode and thus introduction of a Transaction ID, some preliminary information is required. By allowing an admin user to set session-persistent cookies on every computer on the floor of the Test Lab, this system can retain the work center, action, and location ID values (Figure 4) for every engineer's computer.



pansophydev/TESTAREA
Jul 29, 2024 1:43:29 PM

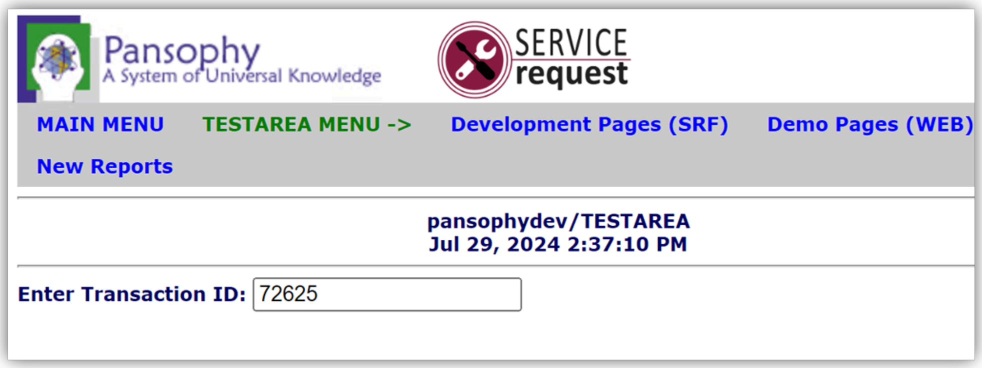
Set Workcenter/Action/Location ID Cookies

Select a workcenter: CHEM - CHEMISTRY
Select an action: ACID - ACID RINSE - PART OF CHEM
Select the location ID: 123

Figure 4: The admin only page used to set computer specific cookies.

The work center variable contains the precise location of the user's machine, the action variable defines the procedure the user is doing at that location, and the location ID is used to more precisely identify the current location. This information is crucial to selecting the correct traveler page to display to the user and for updating the inventory and transaction logs correctly.

Since the barcode scanner inserts the transaction ID as if it was typed manually, a single text entry box is all that is needed for acquiring the Transaction ID (Figure 5).



pansophydev/TESTAREA
Jul 29, 2024 2:37:10 PM

Enter Transaction ID: 72625

Figure 5: The transaction ID entry box. Redirects users to the traveler page once submitted.

Once entered, the program is able to create a Traveler ID in the format:

```
{PROJECT NAME}-{WORK CENTER}-{ACRONYM/PART NAME}-{ACTION}
```

Using this Traveler ID, a corresponding URL can be built that will redirect the user to the appropriate traveler page.

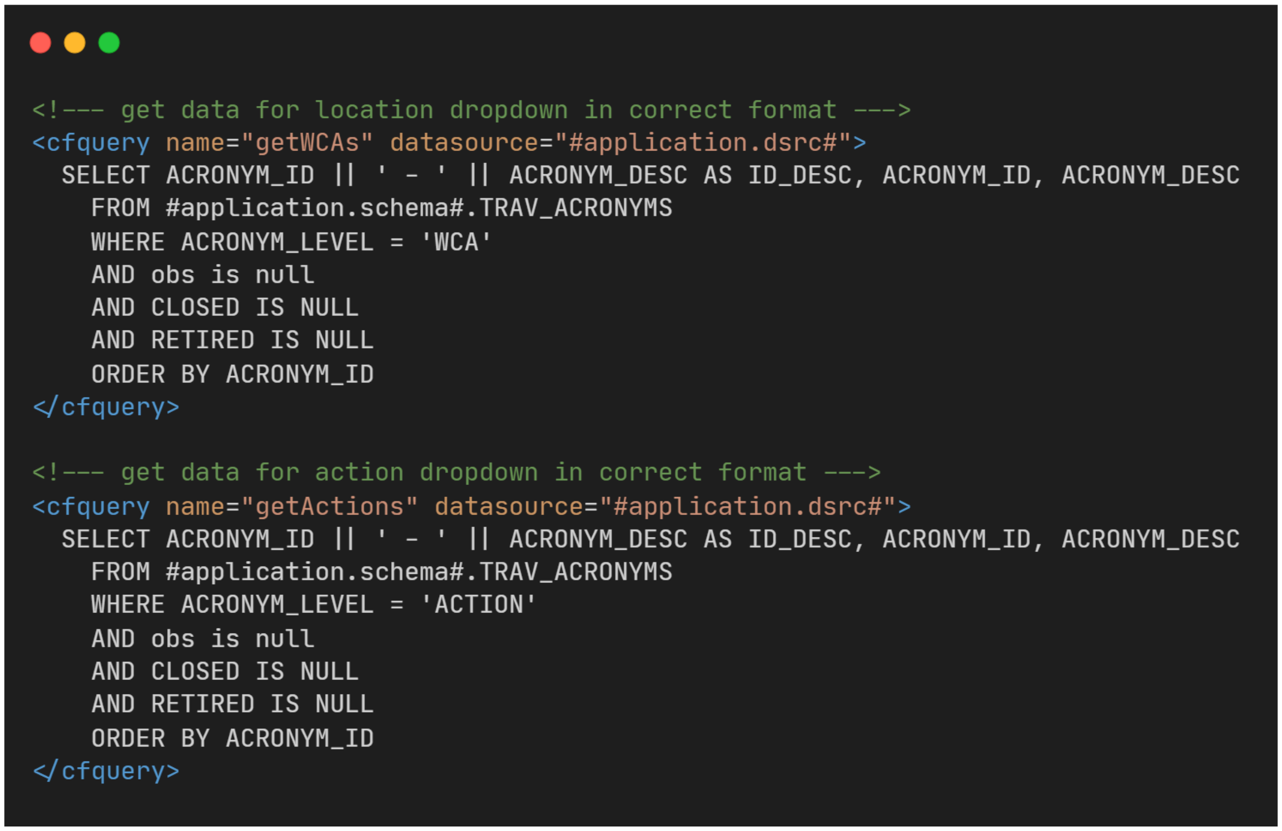
Furthermore, proper error checking must be implemented to account for bad entry, unexpected query results and undefined cookies. For example, if there is a possibility of multiple travelers fitting the criteria prescribed by the user, the system should be able to present the user with a list of relevant Traveler IDs and allow the user to search and select one.

Results

Though the required functionality could have been established in various different ways, by encapsulating core functionality within custom components with specialized functions, the code can be reusable, much easier to debug and more readable.

Almost all queries used rely on the careful matching of primary and foreign keys across tables. Primary keys are keys that are unique to a table, while foreign keys are values that must be matched with their corresponding values in another table to be used. For example, every traveler is given a unique TRAVELER_ID which is a primary key in the INV_TRANSACTIONS table. This means each record in INV_TRANSACTIONS has a unique TRAVELER_ID. However, each record also has a field called LOCATIONID, which is a foreign key. Inside of the INV_TRANSACTIONS table the LOCATIONID is just a number, but a separate table, in this case INV_LOCATIONS, can be queried using that LOCATIONID to retrieve the correct location. The LOCATIONID is a primary key in the INV_LOCATIONS table, where each record has a unique LOCATIONID and a specified name, description, and more.

On the page to set cookies (Figure 4), two initial queries to the TRAV_ACRONYMS table in the Travelers database return details about all possible work centers and actions in a dash-separated format. By specifying certain fields like obs, for obsolete, CLOSED, and RETIRED as null, any work centers or actions that are no longer in use or are invalid are not included in the query (Figure 6).



```
<!--- get data for location dropdown in correct format --->
<cfquery name="getWCAs" datasource="#application.dsrc#">
  SELECT ACRONYM_ID || ' - ' || ACRONYM_DESC AS ID_DESC, ACRONYM_ID, ACRONYM_DESC
    FROM #application.schema#.TRAV_ACRONYMS
    WHERE ACRONYM_LEVEL = 'WCA'
    AND obs is null
    AND CLOSED IS NULL
    AND RETIRED IS NULL
    ORDER BY ACRONYM_ID
</cfquery>

<!--- get data for action dropdown in correct format --->
<cfquery name="getActions" datasource="#application.dsrc#">
  SELECT ACRONYM_ID || ' - ' || ACRONYM_DESC AS ID_DESC, ACRONYM_ID, ACRONYM_DESC
    FROM #application.schema#.TRAV_ACRONYMS
    WHERE ACRONYM_LEVEL = 'ACTION'
    AND obs is null
    AND CLOSED IS NULL
    AND RETIRED IS NULL
    ORDER BY ACRONYM_ID
</cfquery>
```

Figure 6: Two SQL queries used to gather information for the cookie setting dropdowns.

A third query to the INV_LOCATIONS table in the PRIMeS database (Figure 6) collects all valid location IDs.

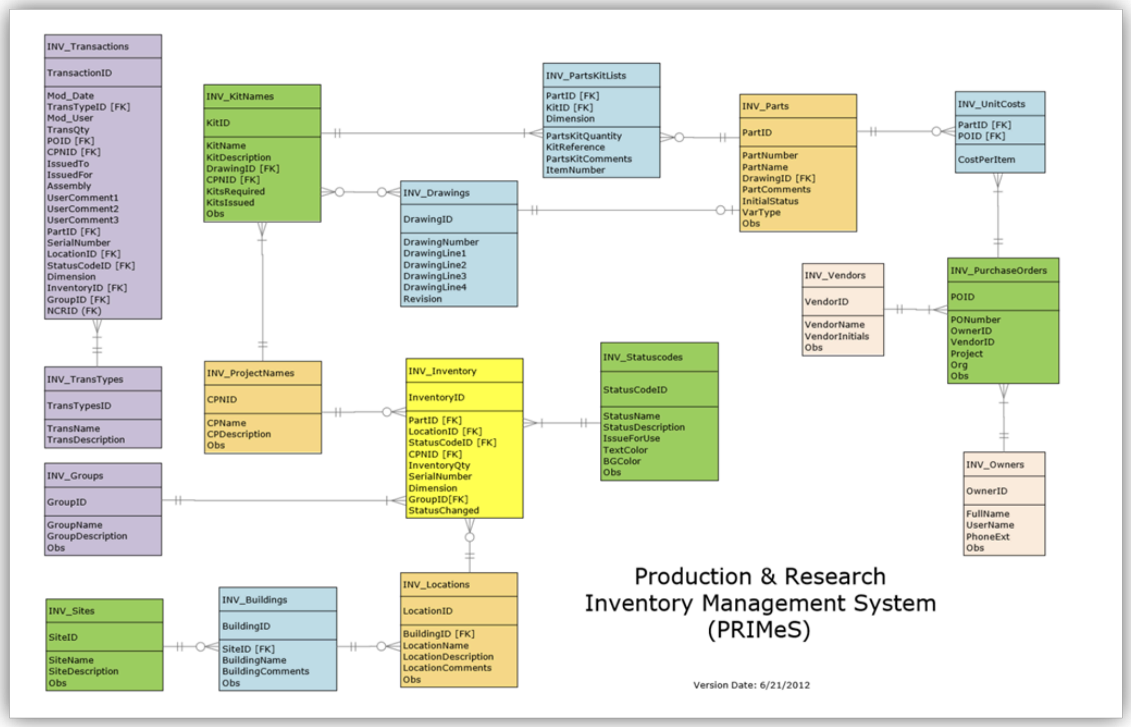
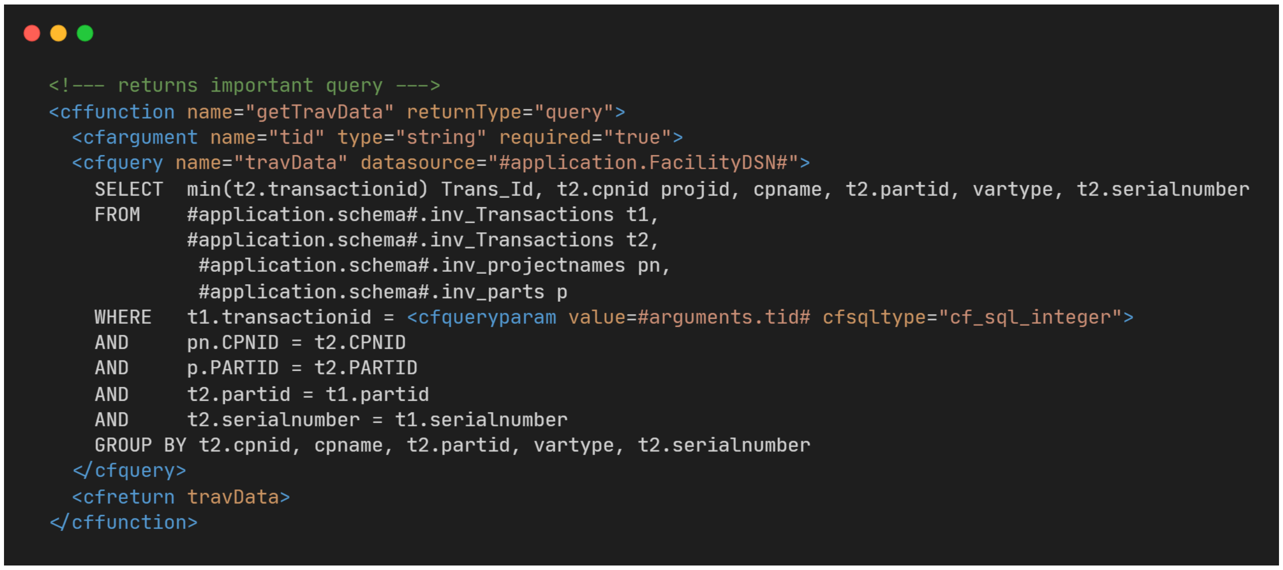


Figure 7: An entity entity relationship diagram describing the PRIMeS database.

These three queries are fed into <cfselect> tags inside of a form to populate three dropdowns with their results. A NULL action was also hard coded into the Location ID drop-down box due to certain travelers omitting that field. Finally, the action= field of the form specifies a separate ColdFusion script that sets the cookies named wca, action and loc_id to their selected values by invoking a method in a custom CookieBuilder component.

On the Transaction ID entry page (Figure 5), almost all functionality is dependent on a conditional statement that requires all cookies to be defined properly and the entry box to have been submitted with a valid integer value. The text entry stores any submitted value to the variable tid which is first used in a large getTravID function from the custom TravIDBuilder component. The getTravID function immediately calls the getTravData function, which includes a query that returns important data about the transaction ID. The TID is sent to this function through the tid parameter, then introduced into the query using a <cfqueryparam> tag (Figure 8).



```
<!--- returns important query --->
<cffunction name="getTravData" returnType="query">
  <cfargument name="tid" type="string" required="true">
  <cfquery name="travData" datasource="#application.FacilityDSN#">
    SELECT  min(t2.transactionid) Trans_Id, t2.cpnid projid, cpname, t2.partid, vartype, t2.serialnumber
    FROM    #application.schema#.inv_Transactions t1,
            #application.schema#.inv_Transactions t2,
             #application.schema#.inv_projectnames pn,
             #application.schema#.inv_parts p
    WHERE   t1.transactionid = <cfqueryparam value=#arguments.tid# cfsqltype="cf_sql_integer">
    AND     pn.CPNID = t2.CPNID
    AND     p.PARTID = t2.PARTID
    AND     t2.partid = t1.partid
    AND     t2.serialnumber = t1.serialnumber
    GROUP BY t2.cpnid, cpname, t2.partid, vartype, t2.serialnumber
  </cfquery>
  <cfreturn travData>
</cffunction>
```

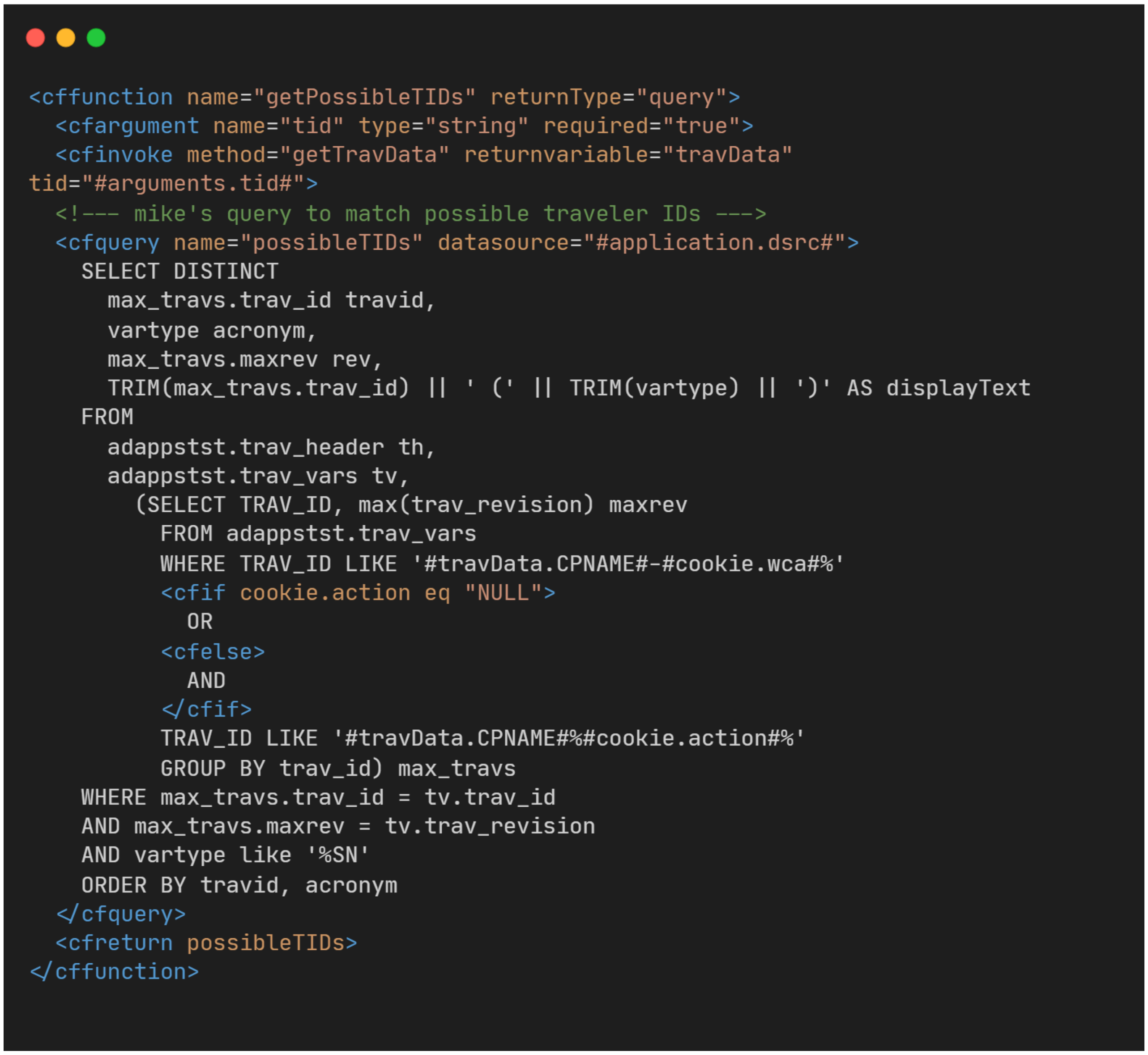
Figure 8: A SQL query that retrieves information about the transaction ID by matching primary and foreign keys.

This tag allows the programmer to specify what kind of data a value should be to avoid SQL injection attacks. By ensuring the tid that is inserted is a valid integer, a user with malicious intent cannot inject SQL code into the query and potentially compromise the database. The results from the call to getTravData are returned as a query, then used alongside preset cookie values in a series of string concatenations that build a valid Traveler ID.

Next, the shouldUpdateLogs method in the TravIDBuilder component is used to query the INV_TRANSACTIONS and INV_INVENTORY tables to determine if the part is still in inventory. The INV_TRANSACTIONS table houses a history of all travelers, while the INV_INVENTORY table houses a list of all parts in inventory. If the part is in the inventory and is not already in the current user's location, a new record is inserted into the INV_TRANSACTIONS table and the corresponding record in the INV_INVENTORY table is updated with the new location ID.

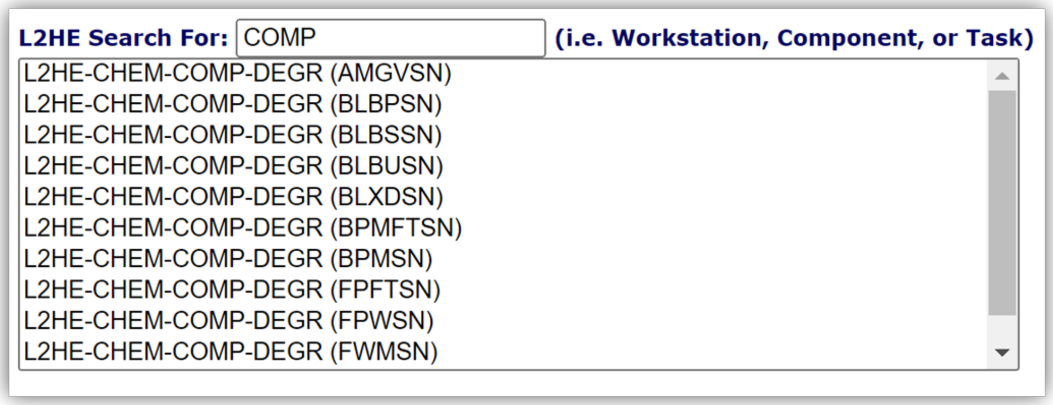
It is possible for a traveler to have multiple revisions added to it as certain actions are performed on it multiple times. A call to the getRevisionNumber method of the TravIDBuilder component returns the total number of revisions a traveler has received and, if valid, it is appended to the Traveler ID. The final piece of information needed for building the URL and redirecting the user is the maximum page number of the traveler, which is found by querying the TRAV_HEADER table in the PRIMeS database. Finally, all information found including the project name, work center, traveler ID, revision number and max page is concatenated to a URL the user is redirected to.

In the case of the getRevisionNumber function returning an empty string, meaning the Traveler ID is invalid, the getPossibleTIDs function is called. By attempting to match the traveler ID with a filtered set of options with the LIKE keyword in SQL and a nested query (Figure 9), this function returns a list of possible traveler IDs that the user can search through and select (Figure 7).



```
<cffunction name="getPossibleTIDs" returnType="query">
  <cfargument name="tid" type="string" required="true">
  <cfinvoke method="getTravData" returnvariable="travData"
tid="#arguments.tid#">
  <!--- mike's query to match possible traveler IDs --->
  <cfquery name="possibleTIDs" datasource="#application.dsrc#">
    SELECT DISTINCT
      max_travs.trav_id travid,
      vartype acronym,
      max_travs.maxrev rev,
      TRIM(max_travs.trav_id) || ' (' || TRIM(vartype) || ')' AS displayText
    FROM
      adappstst.trav_header th,
      adappstst.trav_vars tv,
        (SELECT TRAV_ID, max(trav_revision) maxrev
        FROM adappstst.trav_vars
        WHERE TRAV_ID LIKE '#travData.CPNAME#-#cookie.wca#%'
        <cfif cookie.action eq "NULL">
          OR
        <cfelse>
          AND
        </cfif>
        TRAV_ID LIKE '#travData.CPNAME#%#cookie.action#%'
        GROUP BY trav_id) max_travs
    WHERE max_travs.trav_id = tv.trav_id
    AND max_travs.maxrev = tv.trav_revision
    AND vartype like '%SN'
    ORDER BY travid, acronym
  </cfquery>
  <cfreturn possibleTIDs>
</cffunction>
```

Figure 9: A nested SQL query using the LIKE keyword to collect possible traveler IDs.



L2HE Search For: COMP **(i.e. Workstation, Component, or Task)**

L2HE-CHEM-COMP-DEGR (AMGVSN)
L2HE-CHEM-COMP-DEGR (BLBPSN)
L2HE-CHEM-COMP-DEGR (BLBSSN)
L2HE-CHEM-COMP-DEGR (BLBUSN)
L2HE-CHEM-COMP-DEGR (BLXDSN)
L2HE-CHEM-COMP-DEGR (BPMFTSN)
L2HE-CHEM-COMP-DEGR (BPMSN)
L2HE-CHEM-COMP-DEGR (FPFTSN)
L2HE-CHEM-COMP-DEGR (FPWSN)
L2HE-CHEM-COMP-DEGR (FWMSN)

Figure 10: A Search and Select form where users can search for their desired traveler ID.

As the user types in the provided text input box, the results are filtered with JavaScript to contain only those that match the user's input. When a valid option is selected, the submitTravID.cfm script is executed, which redirects the user to the correct traveler page with the now adequate amount of information.

Challenges

The disconnect between the PRIMeS and Travelers database, which is bridged by the construction of a Traveler ID, made it difficult to test the code. Oftentimes a query had to be reverse-engineered to ensure its validity, which took a considerable amount of time. Working in the development side of Pansophy's databases also proved problematic as all the dev data was years old and adhered to old standards. The development database had to be updated to allow for proper testing. Discrepancies in the format for traveler IDs also required edge cases to be accounted for and special functionality to be implemented to account for exceptions to the standard format. Lastly, ColdFusion's insistence on a <cfform> submission redirecting the user to a separate ColdFusion page rather than allowing a function to be executed on submit led to major code refactoring as the system scaled and required such functionality.

Future Work

This work is currently being implemented in the production Pansophy site where engineers can use it. Real world testing in the Test Lab is needed to gain feedback from real users and potentially change functionality to better fit their needs. As new projects, parts and processes come about at Jefferson Lab, this system may also need to be extended or altered to accommodate for any future changes. The modularization of the current system with custom components and functions should facilitate this process.

Acknowledgements

I would like to give a special thanks to Mike Dickey and Valerie Bookwalter for their excellent mentorship and guidance throughout this project, as well as Megan McDonald, Matthew Menia and the rest of the Pansophy Team for their support. Additionally, I would like to acknowledge Carol McKisson and Jalyn Dio for making the program possible, and my fellow interns for making my experience even more enjoyable.

References

Bookwalter, V., Dickey, M., McEwen, A., & Salvador, M. (2017). PANSOPHY, a JLAB SRF engineering data management system, supporting data collection, retrieval and analysis utilized by LCLS-II. *OSTI OAI (U.S. Department of Energy Office of Scientific and Technical Information)*. https://doi.org/10.18429/jacow-srf2017-mopb043

CFML reference. (2024, May 30). Adobe. Retrieved July 30, 2024, from https://helpx.adobe.com/coldfusion/cfml-tags-functions-reference-user-guide.html

Contributors, M. O. J. T. a. B. (n.d.). *Learn CF in a week*. https://learncfinaweek.com/

Pansophy — Process and data management. (n.d.). Thomas Jefferson National Accelerator Facility. https://www.jlab.org/accelerator/srf/pansophy