Information-Centric Systems – The Future of Business Systems for the Cloud

Current Systems Overview

The world of information systems is about to enter a new generation of business application development which will see a transition out of the static, high-cost, limited world of the past 3 decades and into a dynamic, low-cost and flexible world for the future. Whether it's finally being able to transition out of the decades old legacy systems that have beset many parts of the industry (read eliminating the 220 billion lines of COBOL code that are still in operation) or being able to effectively automate the many evolving requirements from the interconnected, international world of today, **Information-Centric Architecture** is introducing the standard for that new generation.

Information is about the meaning of data. Yet, the current and 3-decades old solutions to building business applications seem to have forsaken information and turned their focus in two different directions instead.

1. There have been a series of attempts to attack the application development problem from a higher business-level by introducing new languages and graphic techniques, from older 4th generation languages to newer no-code / low-code interfaces. But all of them introduce a solution without addressing the heart of the problem, "the structure of information". The results are satisfactory solutions to simple problems that leave the rigorous heart of the industry untouched.
2. Most of the industry, particularly that directed at the rigorous systems that support major business automation, have focused on physical data. SQL has become a desirable standard, but building applications on top of a set of unstructured, fragmented tables that add meaning to data only through a system of "foreign keys" has been a costly, time-consuming and rigid solution.

For 3 decades the industry has been missing the question at the heart of the problem: **How do we structure data to create meaning.** What is clearly needed is a consistent, structured way of analyzing, understanding and documenting information that promotes an understanding of the whole business problem and generates production systems automatically from specifications of that understanding. This paper is a presentation of a system that finally meets these objectives.

The foundation for this new IS world is an *Information-Centric Architecture (ICA)* built around an extension of a data tree structure called an *Information Tree*, so designated because the data tree extensions produce meaning and understanding, as well as automation. An application development system called *Information-Centric Development Platform* (ICDP – the main topic

of this presentation) has been built around that architecture, which generates *Information-Centric Systems* for business applications into multiple technical environments (primarily for the cloud) from the design-level definitions of ICDP. This ICDP base then becomes another foundation on which the reusable component ICDP libraries are built, which in turn produce the huge gains in productivity, quality and flexibility of the business systems for this new world.

Though ICDP includes a set of design-level components, it begins with 2 major data structures: the Information Tree introduced above and a persistent data model, as follows.

1. The first structure is an IC Data Model, which is an extension of the ER Data Model first introduced over 3 decades ago. For a number of years that ER Data Model generated a great deal of enthusiasm for CASE tools (Computer-Aided Software Engineering). However, CASE fell short of its promises because it never found a way to build on the effectiveness of data modeling to take the next step needed to actually generate business systems. The IC Data Model utilizes the entities, attributes and relationships of the ER Data Model to document persistent data, but also introduces several extensions to that model which add integrity constraints, enhance reusability and generate the physical components necessary to convert the logical model into physical data both when creating databases and when generating physical I/O statements during execution.
2. The second technique is the major innovation of ICDP, an Information-Centric Object model, which uses a visual *Information Tree Structure* to reveal meaning and document data-in-context. This specification is called an *IC Object* (and also a Logical Object Definition because its focus is on the logical structure of data as opposed to the physical). Like the IC Data Model, an IC Object is made up of the logical entities, attributes and relationships of logical data. But while the IC Data Model represents ALL persistent information for a business, an IC Object represents a subset of that information within a particular business context. It is also extended with additional data to add meaning, along with the business rules that process the data within that context. It is particularly important to note that these rules are always defined directly against the information tree structure of the IC Object. In addition, an IC Object automates all work-in-progress functions, including generating the necessary SQL to read data into the logical structure of the IC Object at run-time and write out any work-in-progress changes to the database.

Building on this foundation is an additional set of logical building blocks that define work-flow, processing and human interfaces, which together generate business applications in a variety of technological environments. All of this together defines **ICA** and **ICDP**, *the architecture and*

development platform that will build the rigorous, demanding and flexible business solutions of the future.

It should be noted at this point that there is a critical characteristic/capability of ICA and ICDP that is necessary to enable this future, a characteristic that has proven elusive in CASE and in other ambitious automation efforts: For this vision to work, ALL rigorous executing business systems must be generatable completely from the logical business specifications at the heart of ICA. If the executables need to be modified, even tweaked in small ways, to realize the desired functionality, then the whole process breaks down. To repeat, **the desired executable systems must be able to be completely generated from the logical specifications of an ICA application.**

This application development system has huge value in itself. But it is important to understand that the ICDP system described above is also a **foundation: the standard on which the applications of the future will be assembled from reusable component libraries in a matter of hours and days instead of the months and years of current systems**. The concept of using reusable components has been part of practically every application automation solution of the past 3 decades. But that reusability has always been directed at programming or at non-information structured processes, limiting the power of that reusability. ICDP's reusable components are built on IC Data Models and IC Objects that keep the development process focused on logical information and on the processes and interfaces that use that information.

We will introduce the use of reusable component libraries here through an example: a Customer Relationship Management (CRM) Library, a standard set of customer communication and analysis functions that add important data analysis functionality to most business systems. The steps and techniques used in this example are common to the utilization of all ICDP reusable libraries.

Building Business Applications by Assembly from Reusable Component Libraries

A Reusable Component Library is a set of design-level definitions that specify all the data, processing and interface information necessary to generate a fully-functioning business application for the cloud. At the heart of each library is an IC Data Model and a set of IC Objects that set a consistent foundation for understanding and processing information at all stages in the life of the application. The key components of a library include:

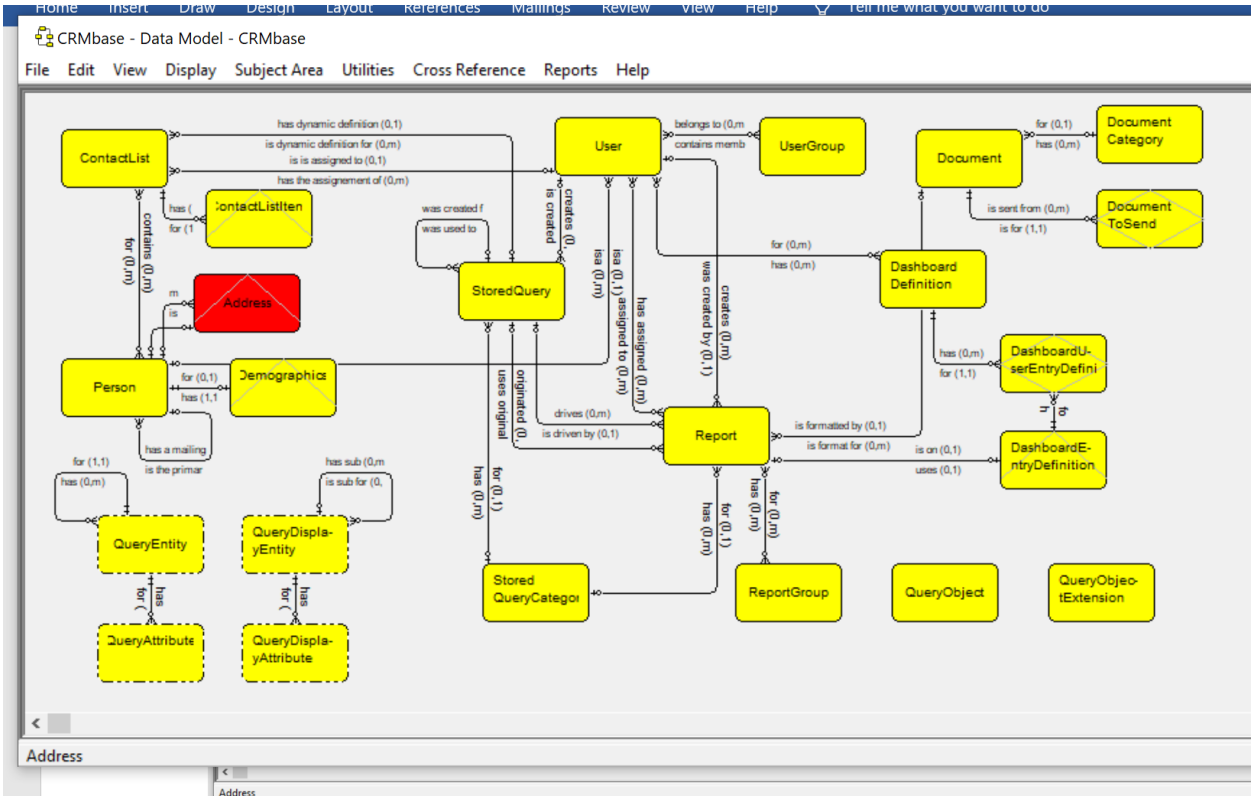
- IC Data Model
- IC Objects with Related Processing Rules
- Interactive Dialog Definitions, which Define Interface Components and Related Processing Rules
- Other Presentation Interface Definitions and Related Processing Rules

CRMBase Component Examples

Sample components for the CRMBase library follow.

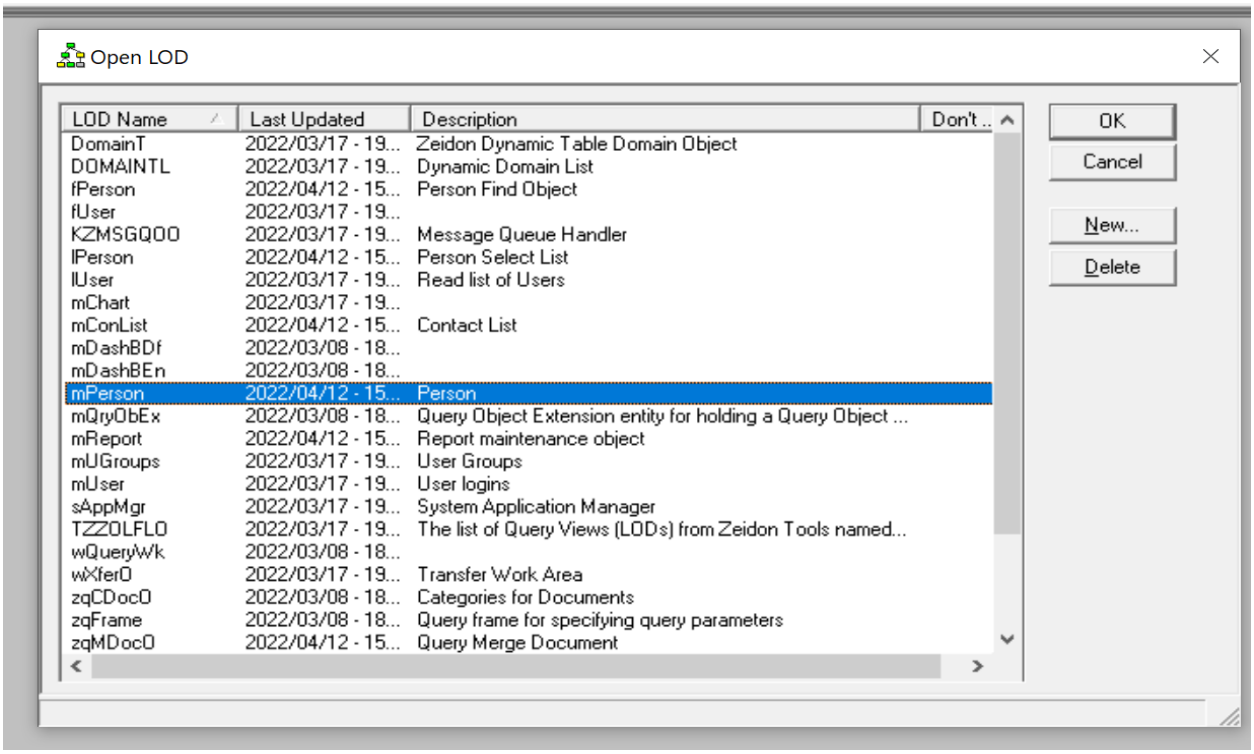
CRMBase IC Data Model

An IC Data Model defines the persistent data that will be stored in a database for an application. Though an IC Data Model can be defined for different kinds of persistent data, the current implementation supports standard SQL relational databases.



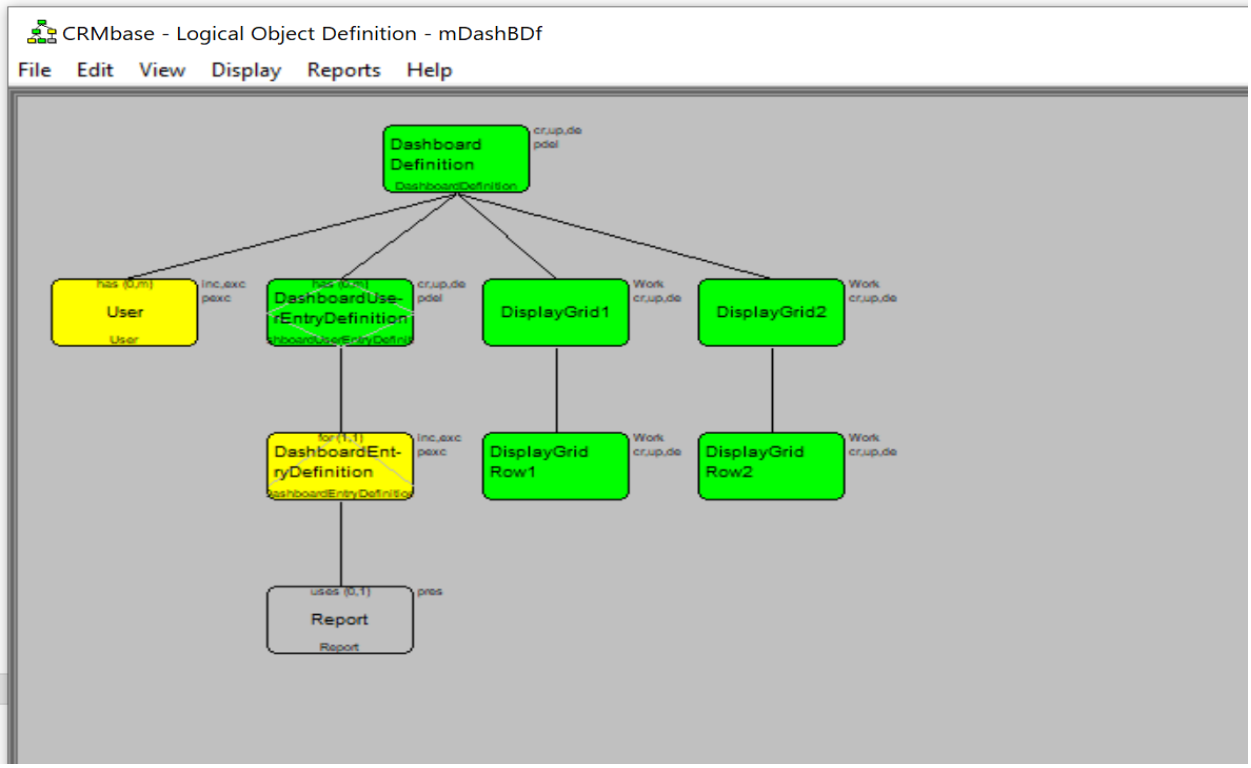
CRMBase IC Objects

At the heart of ICA is the IC Object, which uses a visual Information Tree structure to define the data that is used within a specific context. ALL data within an application is defined through IC Objects, with most of them comprised of a subset of data from the IC Data Model. The objects define, document and generate most of the data handling of the executing system. The object list below shows most of the IC Objects for the CRMBase component library.



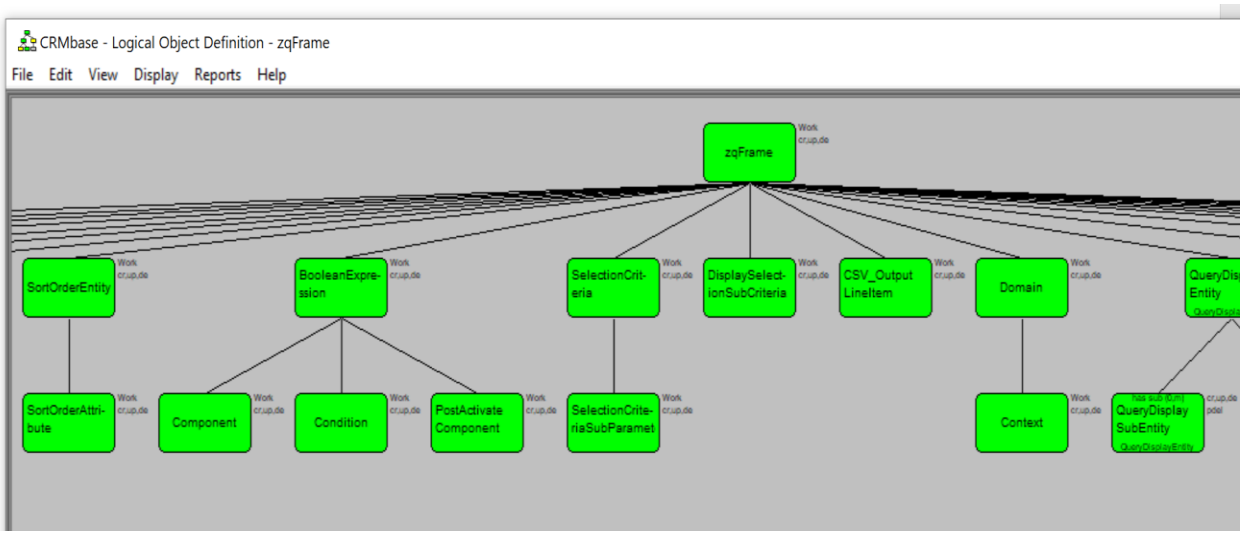
CRMBase Sample Object – Dashboard Definition

The following is a simple IC Object for carrying dashboard information. It is comprised of both database data and work data that is used in processing dashboards.



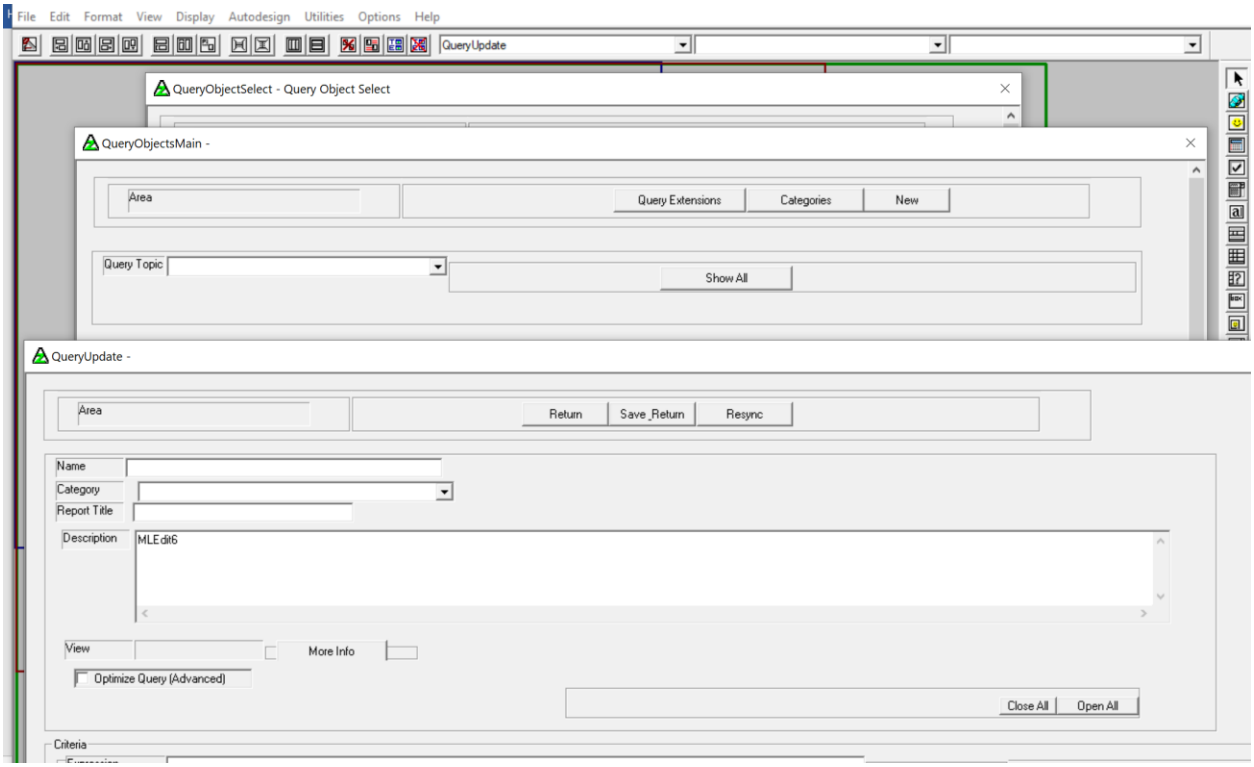
CRM Sample Work Object – Query Definition

The following is an example of an IC Object made up entirely of work data for defining complex database queries. Though the individual components are not stored as tables and columns in a database, instances of such objects are stored as a single blob in the database or as a stand-alone file. Thus, IC Objects can imitate much of the functionality of object-oriented databases.



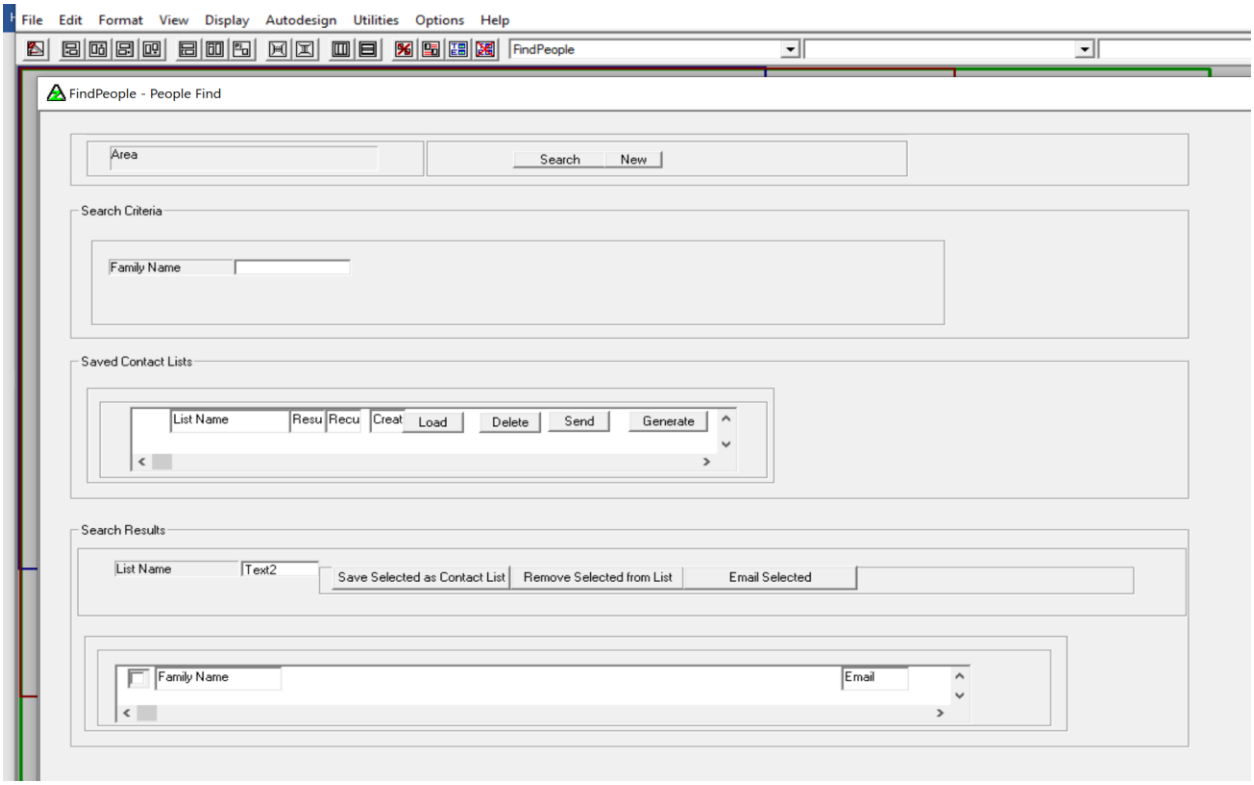
CRMBase Sample Dialog – Query/Reporting Page Specifications

The following is part of the CRM interactive dialog definitions for defining query and reporting functionality within CRM. A Dialog contains the “painted” logical layout of multiple pages that support the interactive query functionality. The Dialog interfaces and functionality are simply merged into the target application and generated into the executable system.



CRMBase Sample Object – Data Find Template to Be Copied and Modified

The following is part of a Dialog template which is the base for adding CRM find functionality to the target application. Target application data is added to the interfaces either through "painting" data on the interface or having it "auto-designed" on the interface from a selection of data attributes.

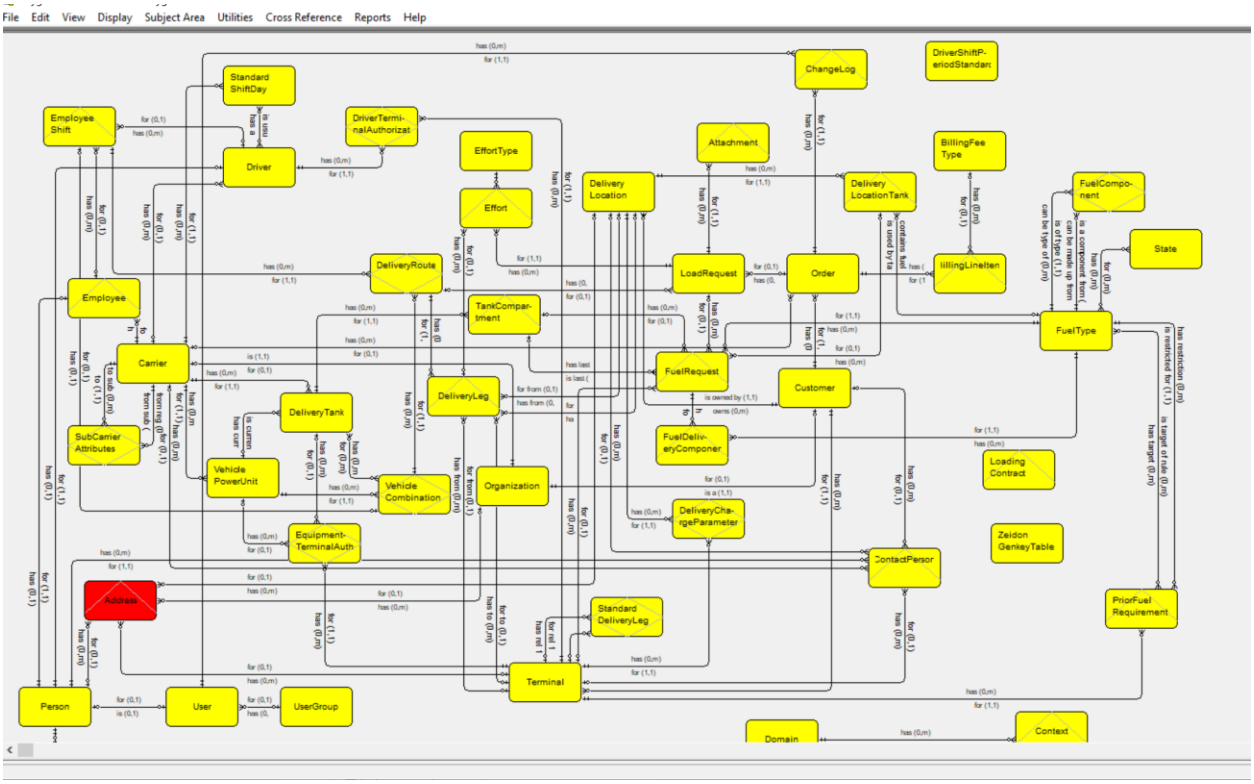


Merging CRM Functionality into a Target Application

The following are the steps which take all or part of the components in a CRM library and merge them into a target application. In this example, the target is an application for scheduling fuel deliveries and the CRM communication functionality will be extended to support customer contact people and delivery truck drivers.

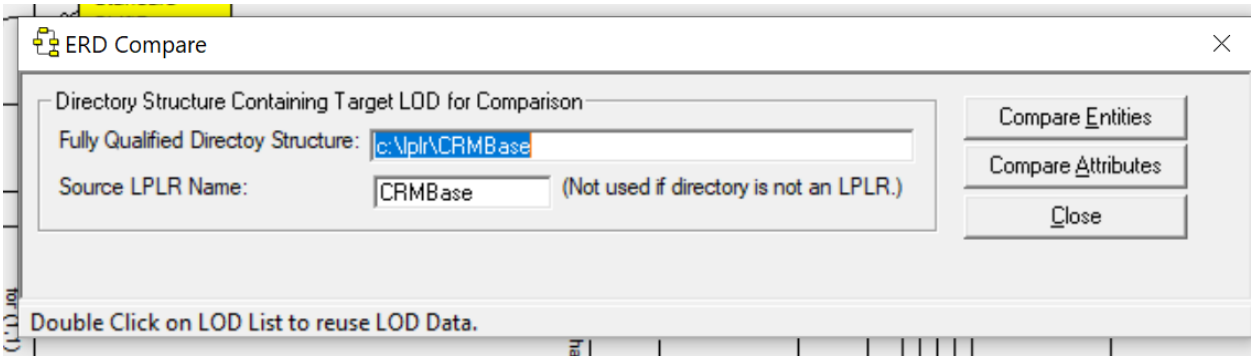
Target IC Data Model – Before Merging in CRM Functionality

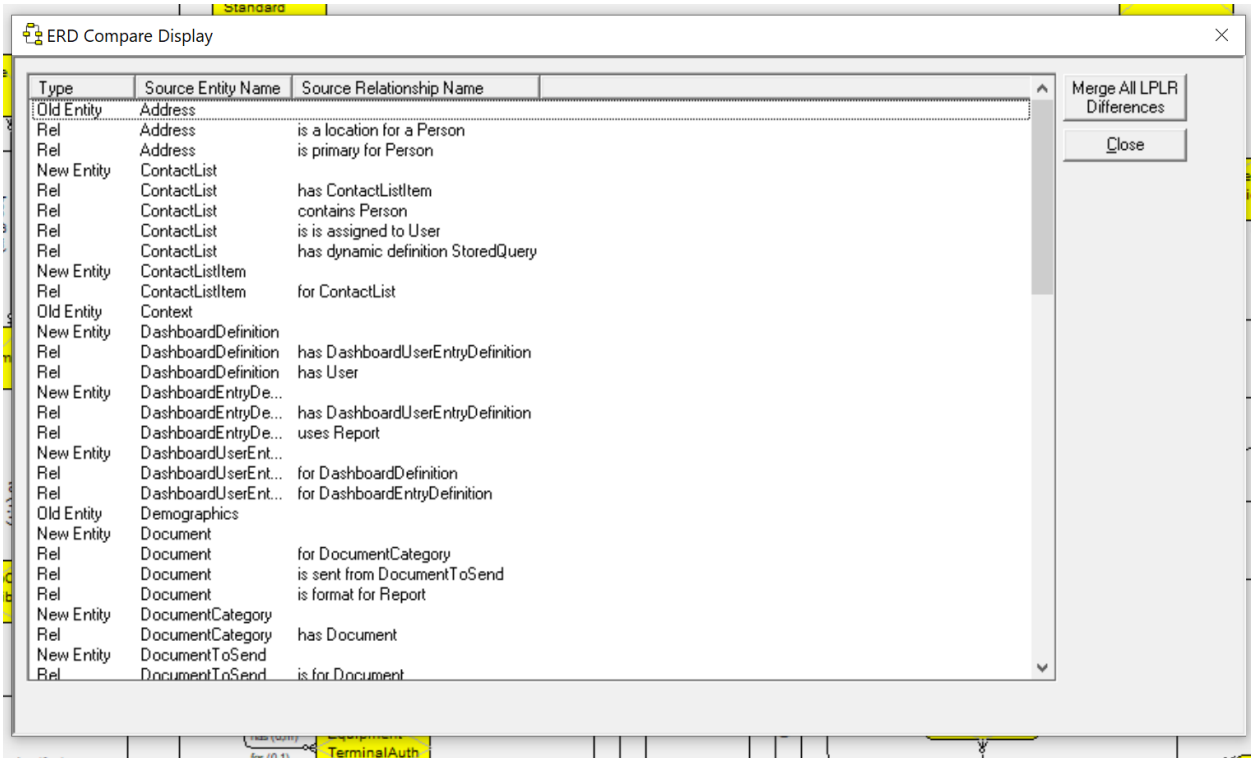
The following is the target IC Data Model before merging in CRM functionality.



Request to Merge CRMBase Data and Function

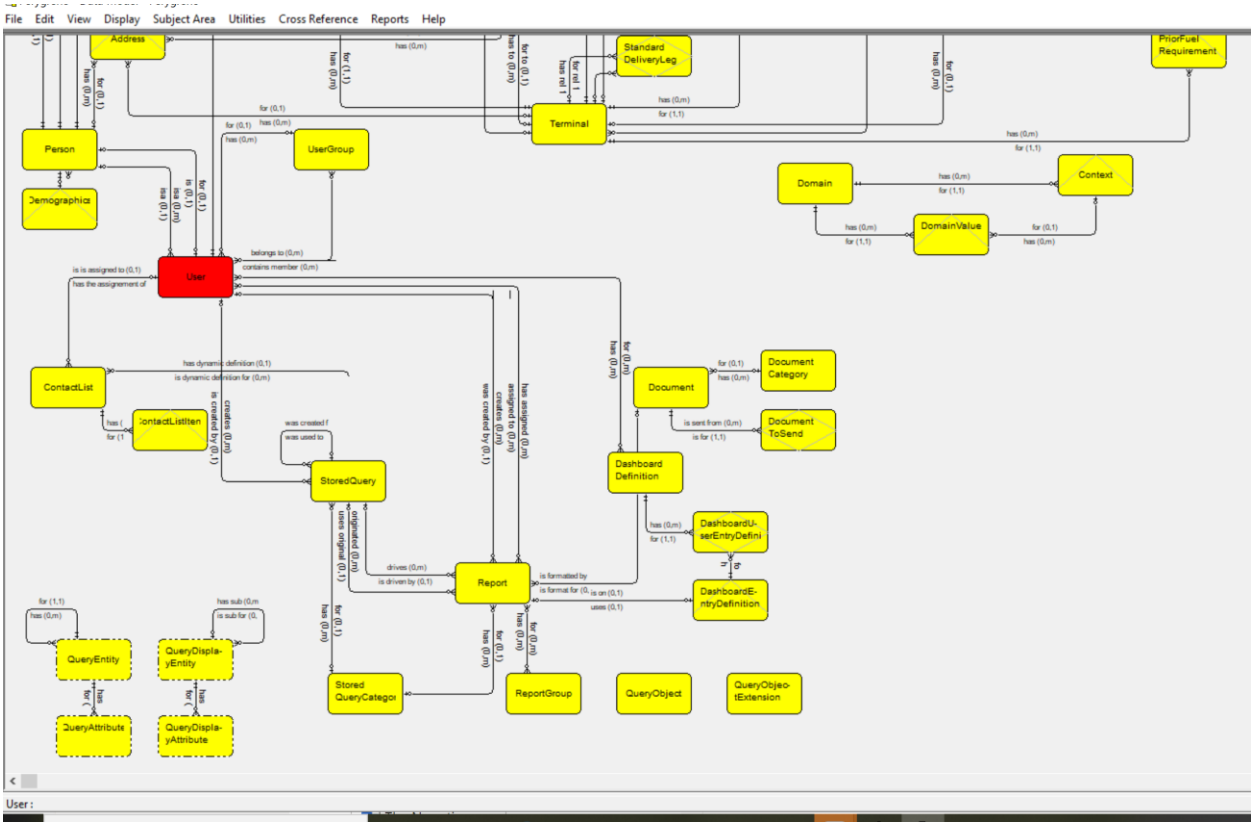
The user selects an “Edit/Compare/Merge” menu option which presents the following page to identify the source library for the merge. They then select the “Compare Entities” button, followed by the “Merge All LPLR Differences” on the returned page, as shown below. In a few seconds, all CRMBase component definitions are merged into the target library. The user could have chosen instead to only merge specifically selected entities.

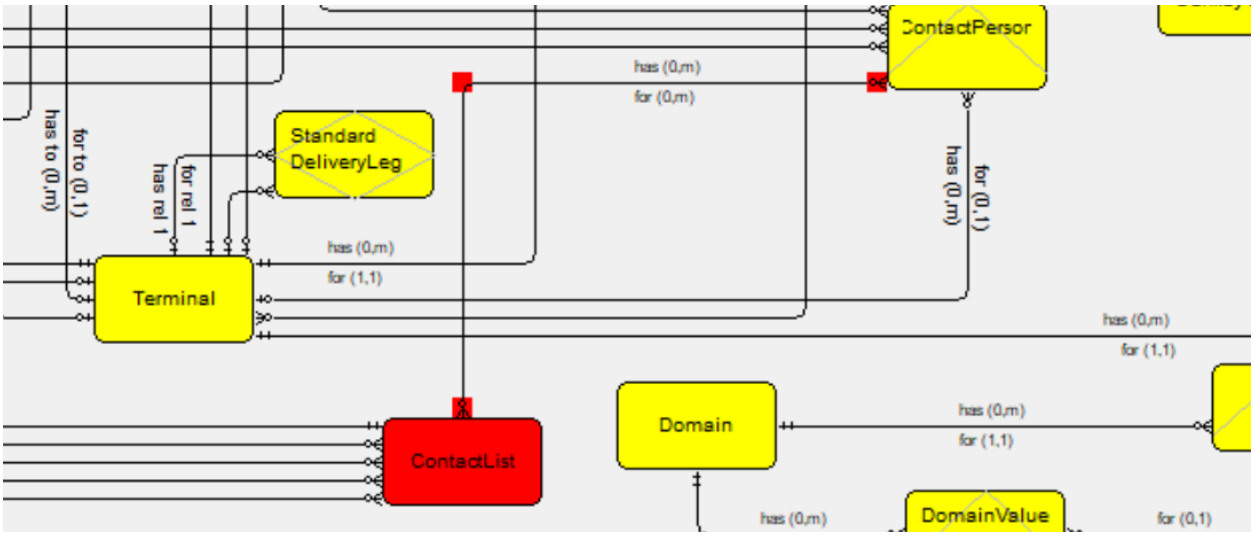




Target IC Data Model – After Merging in CRM Functionality

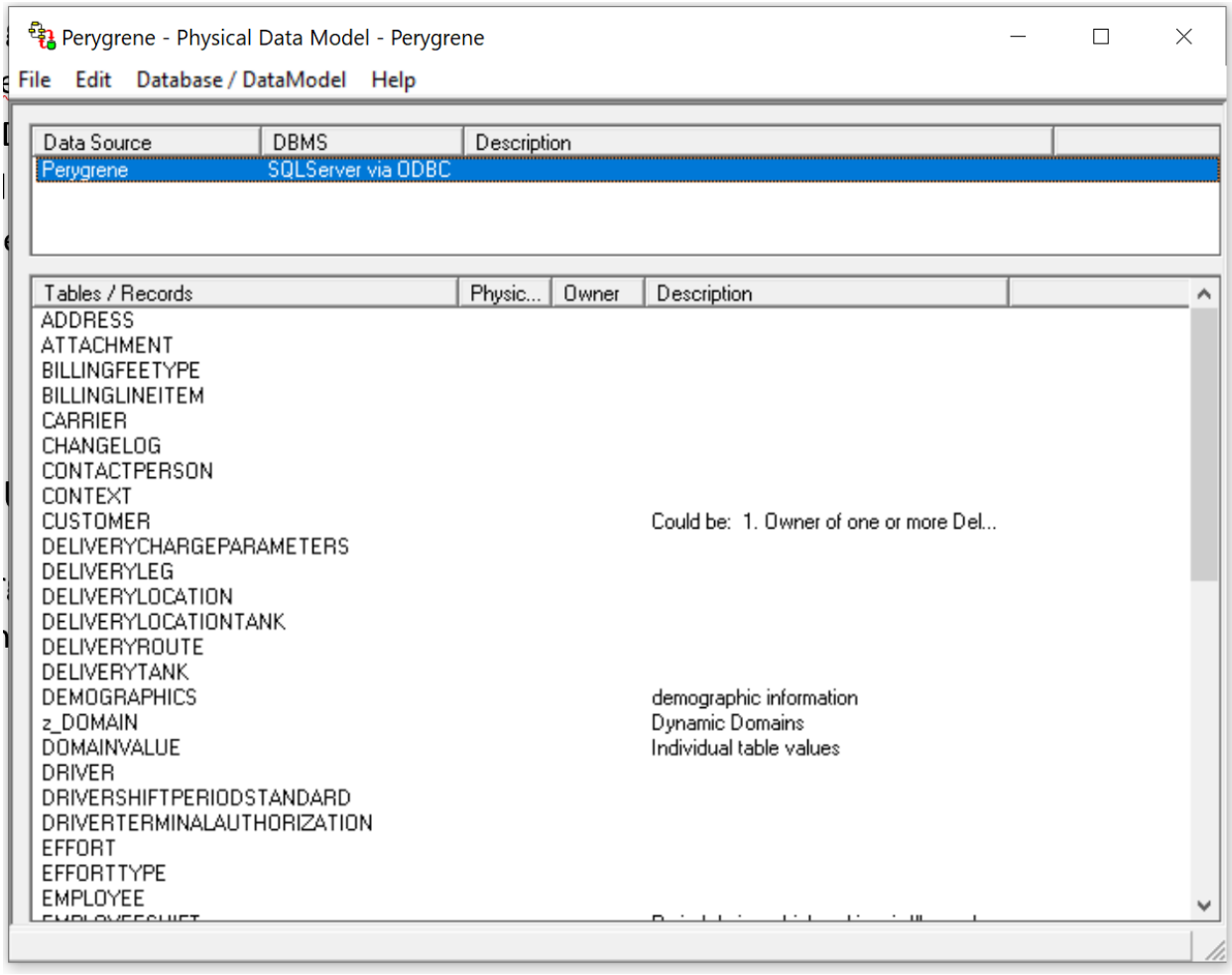
The following is the target IC Data Model after CRMBase data content has been merged into the bottom of the original data model diagram. An m-to-m relationship is then drawn between the ContactList entity and the ContactPerson and Driver entities for tying customer contact personnel and drivers into the CRM functionality, as shown below.





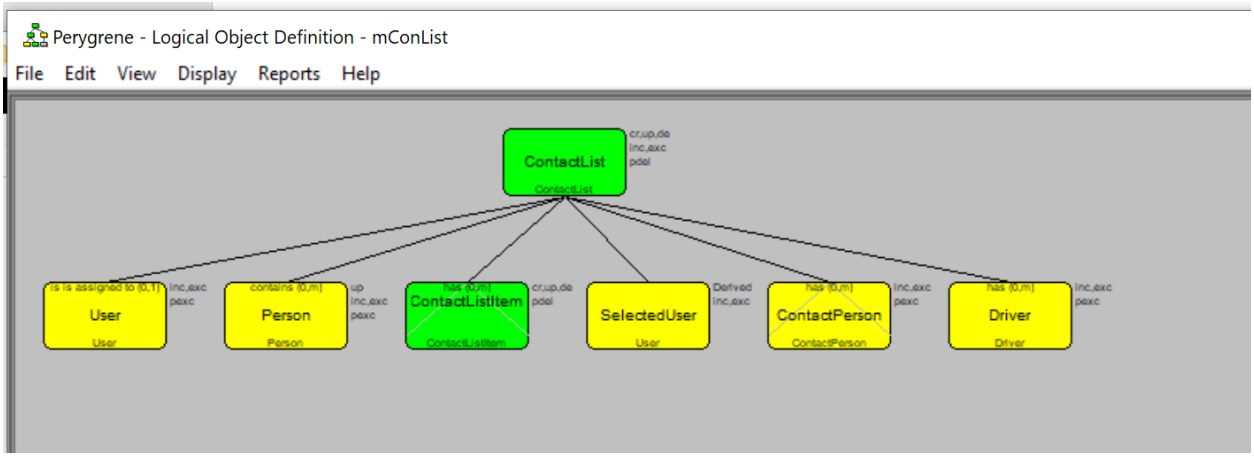
Regenerating Physical Data Definition and Generating Modification DDL

The next step is to bring up the Technical Environment specification tool, run the menu option “Data Base / DataModel / Build/Rebuild Table/Rels….”, which generates SQL physical definition entries, and then the “Data Base / DataModel / Build Sync DDL”, which generates change DDL, which is then executed on the appropriate target database. The following shows the resulting physical data as specified in the Technical Environment tool, with the menu options across the top.



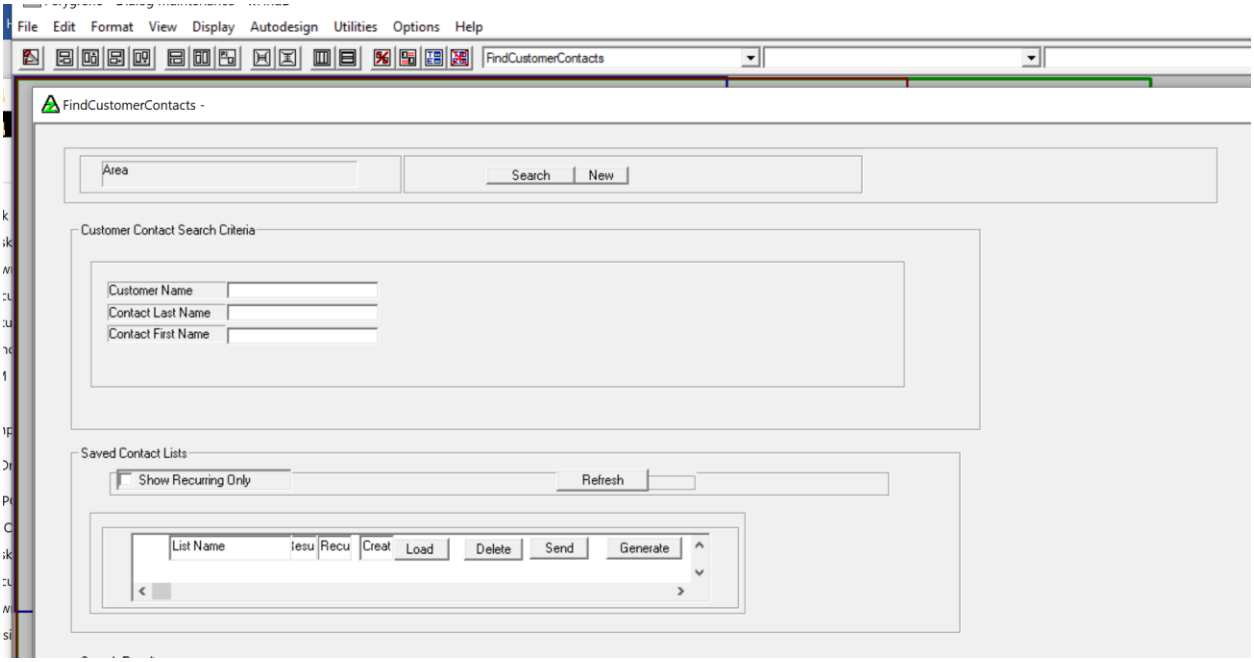
Contact List Object Update – Adding ContactPerson and Driver to the Object

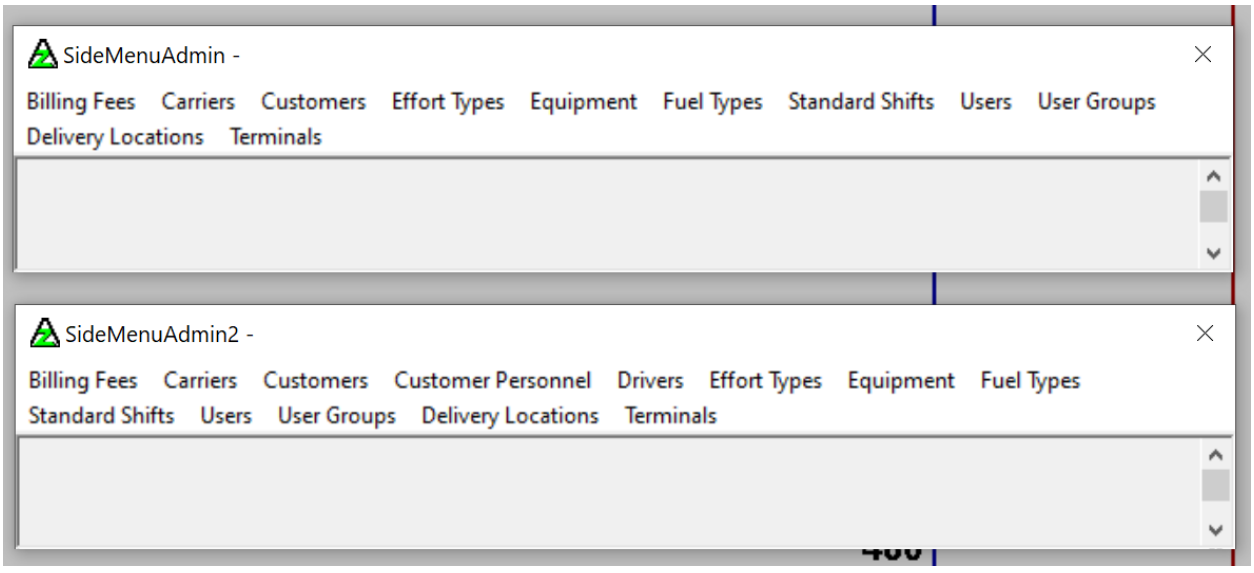
After the physical data is generated, the next step is to simply add the ContactPerson and Driver to the ContactList object, resulting in the following.



Contact List Dialog Update – Adding ContactPerson and Driver Find Pages

All of the necessary data and base interface functionality has at this point been merged into the target application library. The next step is to take the template Find page, make a copy of it and then trigger the find function from the desired menu option in the target library, as shown in the following examples. The first menu definition is the original definition, followed by the update where “Customer Personnel” and “Drivers” are added to link to the new CRM Find pages.



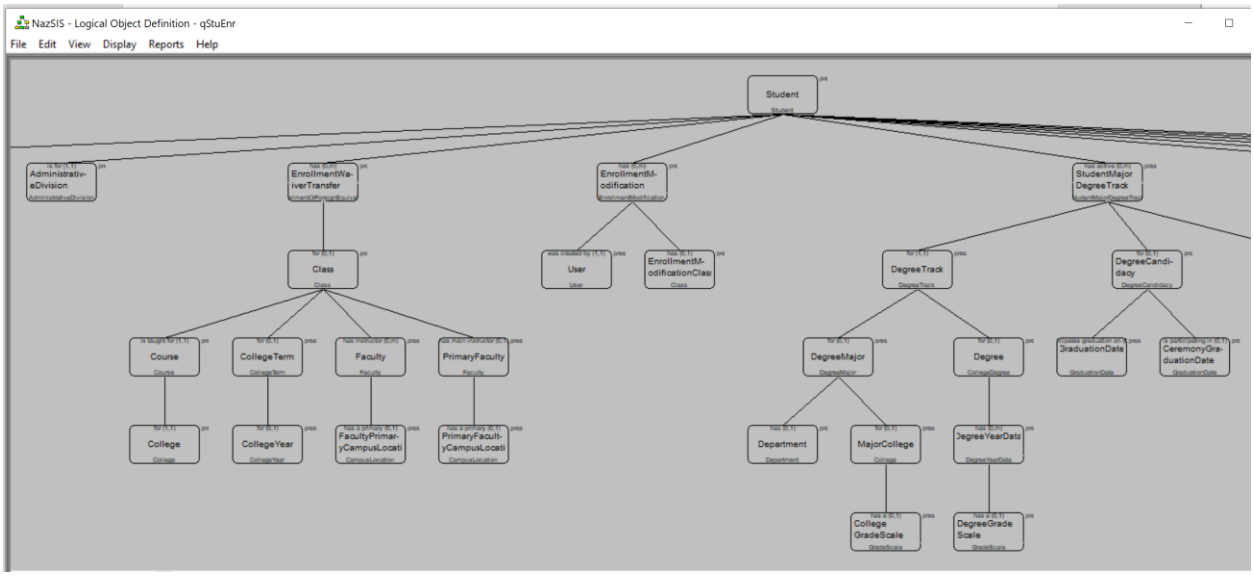


Building Query Objects to Define the Information for Communication

The last step in the merge process (and one that continues throughout the life of the application as business changes require additions and modifications) is to build the query objects that contain all the application information to be communicated through the various kinds of presentation media, including reports, documents, emails and graphic presentations. The same IC Object structure is used to provide a visual understanding of complex data relationships and automate processing. The following are a series of examples from an international Student Information System that embodies the requirements for rigorous cloud applications and heavily utilizes the CRM paradigm and reusable CRMBase components.

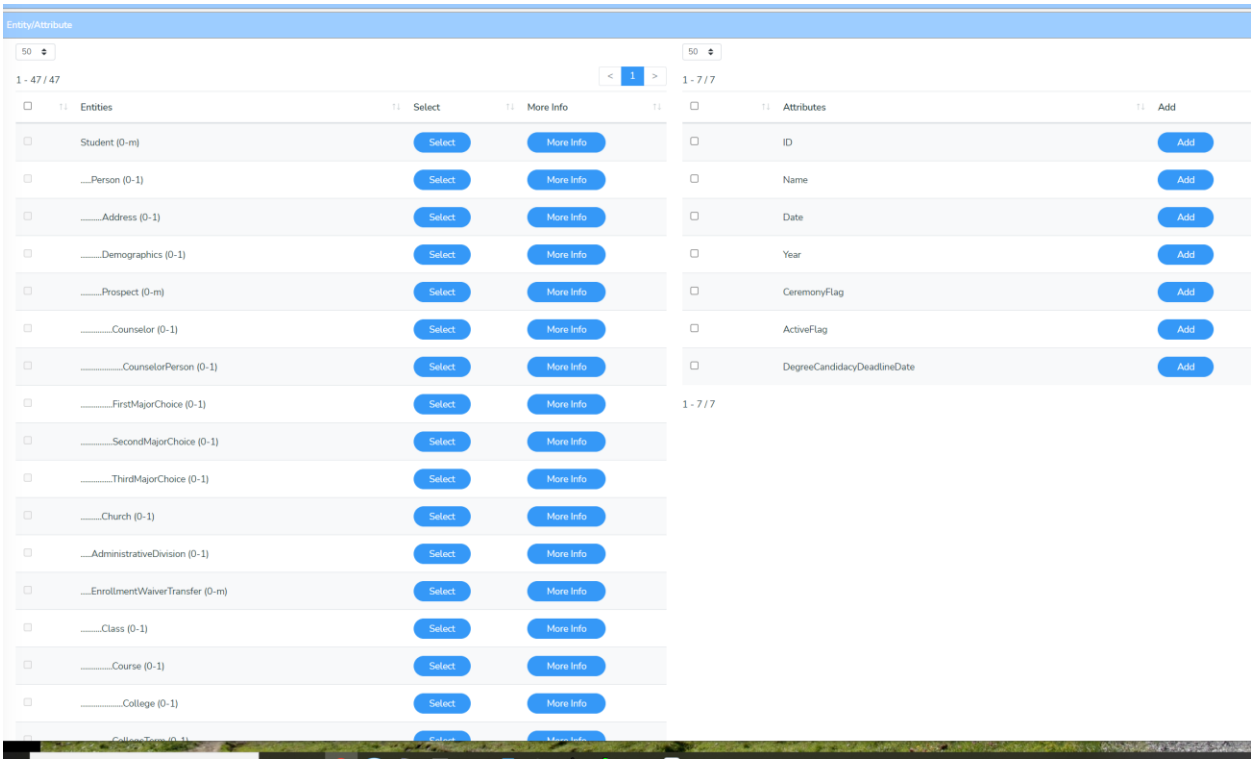
Sample Student Academic Query Object – Graphic Form

A rigorous SIS includes a huge quantity of student academic information. The student academic query object, which is shown in part below, includes nearly 50 database entities. This amount and data complexity is almost impossible to manage without a clear visual structure and the automated functionality that goes with it.



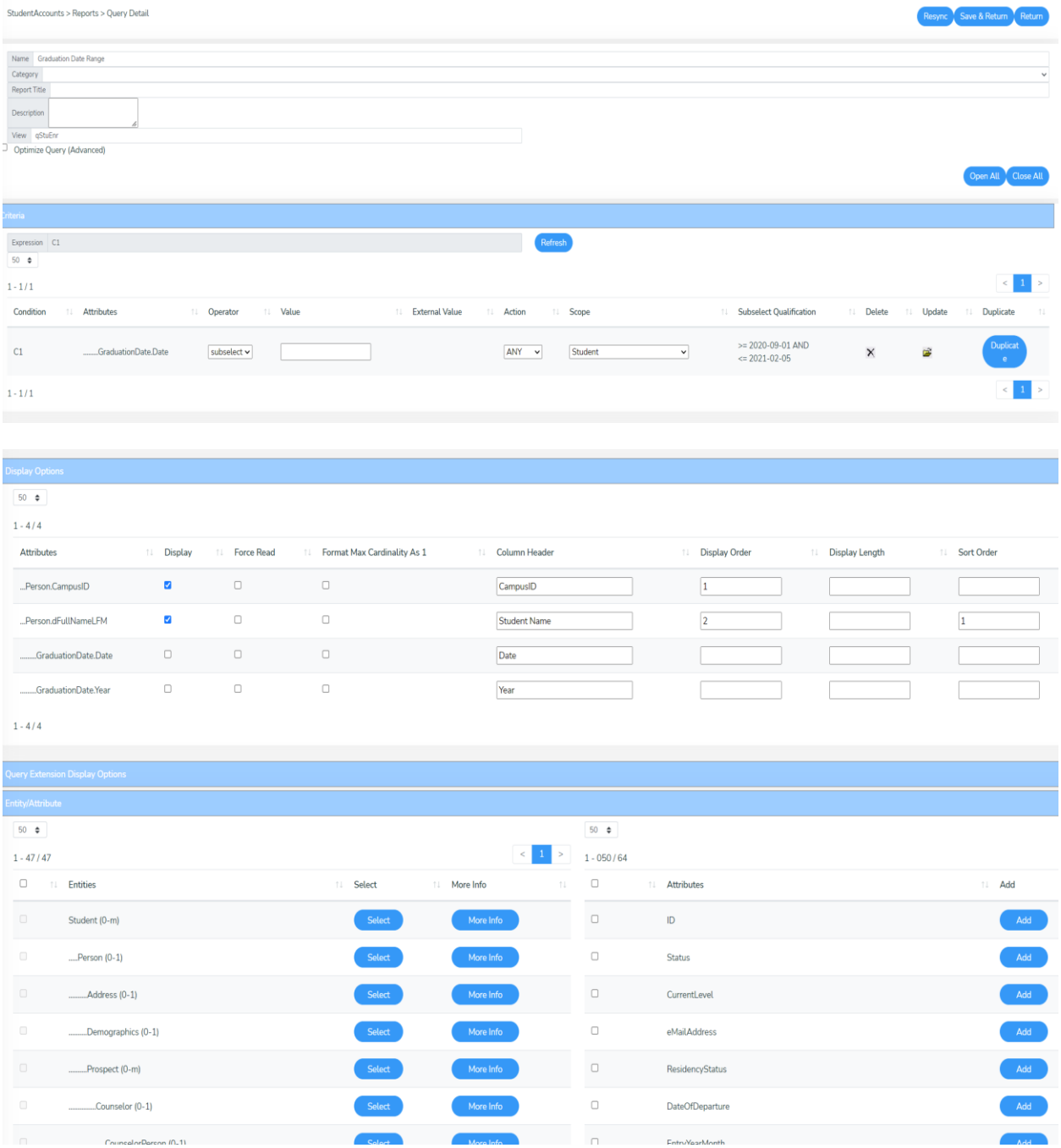
Sample Student Academic Query Object – Listbox Form

During execution, a listbox form of the above object structure is used for selecting object components in displaying data and qualifying criteria for the data returned. The tree structure of the object is represented by the indented entity names on the left, with the attributes for a selected entity listed on the right.



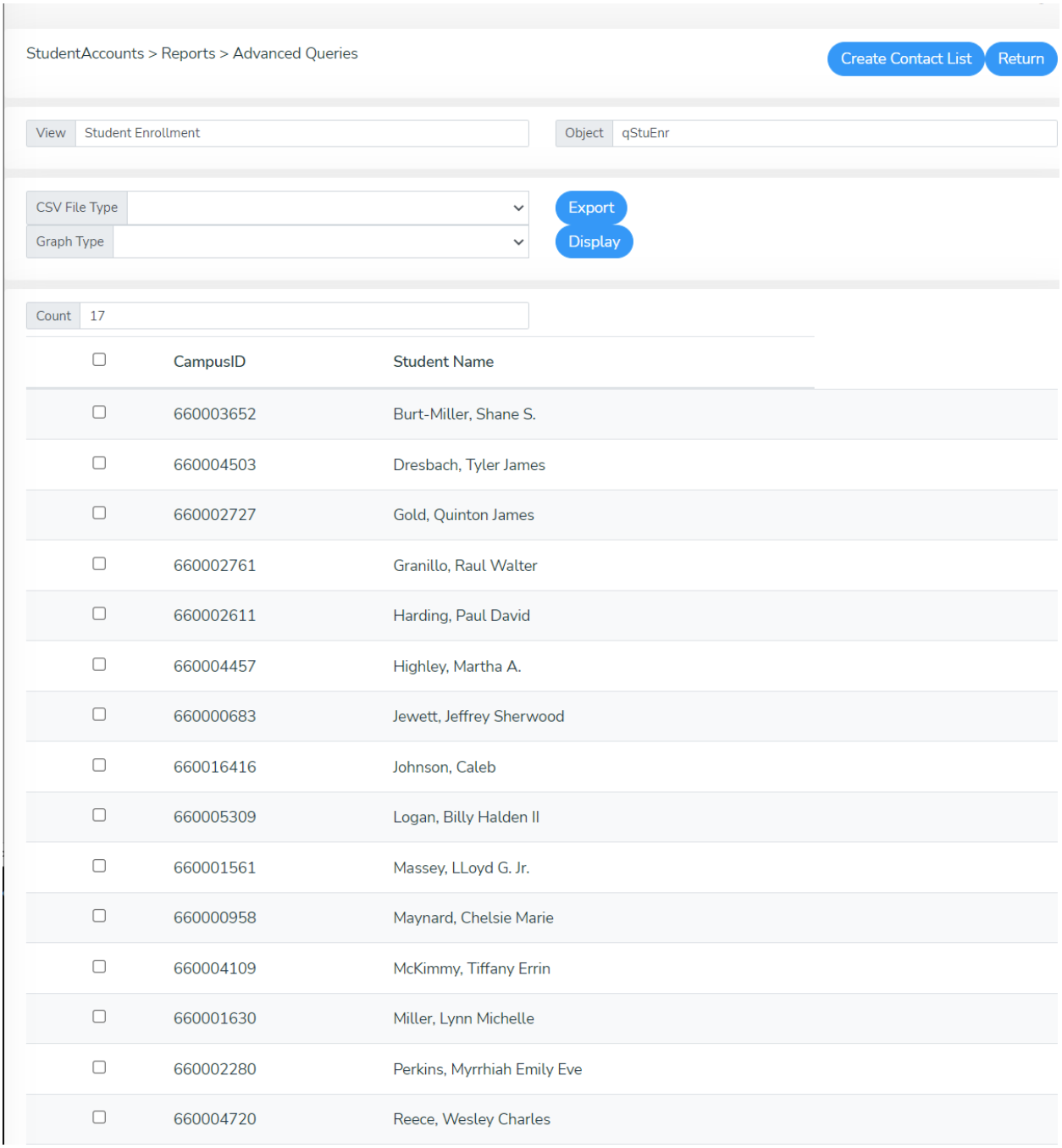
Query Definition Containing Selection Criteria and Presentation Data

The following page components show a query with its current selection criteria and display data, followed by the full object entity and attribute lists for selecting that criteria and display data. The collapsed “Query Extension Display Options” group can be used for selecting derived summary data for summary reports and dashboards.



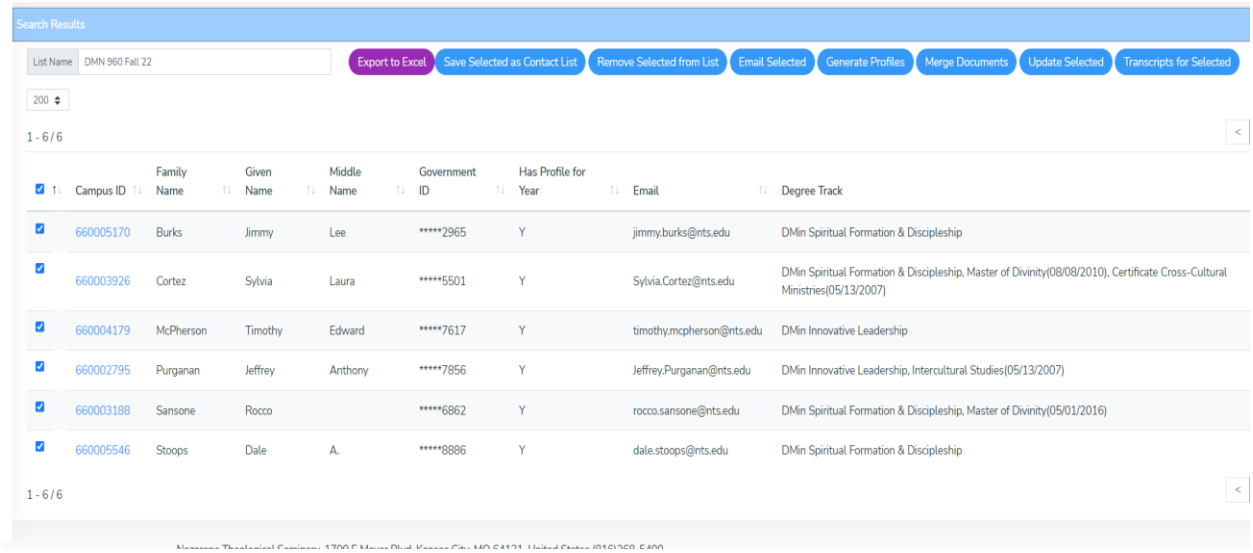
Query Execution Summary List for Creating CRM Contact Lists

When a query is run, the initial data returned is a summary of the selected entries, which can then be used to create a CRM Contact List for that data. A Contact List can also be defined against a particular query and triggered at any time to generate a list of desired customer data.



Merge Documents and Emails from Contact Lists and Query Data

A powerful CRM function in communicating with customers and other personnel is the generation of merge documents and emails that contain important custom data for that particular customer or person. The following is a sample contact list page from the SIS Financial Aid CRM interface that selects students for generation of a merged document. Striking the "Merge Documents" button takes the user to a list for selecting a Print or Email Template, which in this sample is an MS Word document template with static and dynamic mapping criteria for defining data content. Following the sample page are a sample template document and the resulting merged document. Note that the template document has IC Object entities and attributes imbedded in the formatted text, including the syntax for repeating data to be listed.



Financial Aid Award Document Template

[Z:CollegeYear.Year]
Award Notification

[Z:Person.dToday] **IMU ID#: [Z:Person.CampusID]**

[Z:Person.dFullName]
[Z:Address.dFullAddress]

Cost of the [Z:CollegeYear.Year] Academic Year

Estimated Institutional Costs ($[Z:AwardLetterFederal.CostOfAttendanceTotal])

Fall **Spring** **Full Year**

[Z:#S:AwardLetterFederalCOAItem][Z:AwardLetterFederalCOAItem.Description]
[Z:AwardLetterFederalCOAItem.FallAmount][Z:AwardLetterFederalCOAItem.SpringAmount]
[Z:AwardLetterFederalCOAItem.Amount]
[Z:#E]

Grants and Scholarships

Total Grants and Scholarship($[Z:AwardLetterFederal.GrantsAndScholarshipsTotal]

Fall **Spring** **Full Year**

[Z:#S:AwardLetterFederalGrant][Z:AwardLetterFederalGrant.Description]
[Z:AwardLetterFederalGrant.FallAmount][Z:AwardLetterFederalGrant.SpringAmount]
[Z:AwardLetterFederalGrant.Amount]
[Z:#E]

Net Costs ($[Z:AwardLetterFederal.NetCosts])

Cost minus grants and scholarships **Fall** **Spring** **Full Year**

[Z:AwardLetterFederal.NetCostsFall][Z:AwardLetterFederal.NetCostsSpring]
[Z:AwardLetterFederal.NetCosts]

Options to Pay Net Costs

Work Options ($[Z:AwardLetterFederal.WorkOptionsTotal])

Federal/Texas Work-Study or regular campus student wages (must work to earn)
($[Z:AwardLetterFederal.WorkOptionsTotal])

Fall **Spring** **Full Year**

[Z:AwardLetterFederal.WorkOptionsFall][Z:AwardLetterFederal.WorkOptionsSpring]
[Z:AwardLetterFederal.WorkOptionsTotal]

Loan Options ($[Z:AwardLetterFederal.LoanOptionsTotal])

Fall **Spring** **Full Year**

[Z:#S:AwardLetterFederalLoan][Z:AwardLetterFederalLoan.Description][Z:AwardLetterFederalLoan.FallAmount]
[Z:AwardLetterFederalLoan.SpringAmount][Z:AwardLetterFederalLoan.Amount]
[Z:#E]

Final Net Costs ($[Z:AwardLetterFederal.FinalNetCosts])

Cost of attendance minus scholarships, grants,
estimated work study and loans **Fall** **Spring** **Full Year**

[Z:AwardLetterFederal.FinalNetCostsFall][Z:AwardLetterFederal.FinalNetCostsSpring]
[Z:AwardLetterFederal.FinalNetCosts]

Other Options

Imaginary U
125 years educating for success

"For I know the thoughts that I think toward you, says the Lord, thoughts of peace and not of evil, to give you a future and a hope."

Jeremiah 29:11

Knowledge. Faith. Service.

Financial Aid Award Document Sample Generated Document

IMAGINARY U

2021-2022
Award Notification

August 13, 2021 **IMU ID#: 269778**

William Martin
9644 Nothing Road
Houston , TX 77088

Cost of the 2021-2022 Academic Year

Estimated Institutional Costs ($31,608.00)

	Fall	Spring	Full Year
Tuition	11,424.00	11,424.00	22,848.00
SA Fee	110.00	110.00	220.00
Technology Fee	220.00	220.00	440.00
Room and Board - On Campus	4,050.00	4,050.00	8,100.00

Grants and Scholarships

Total Grants and Scholarship ($7,000.00)

	Fall	Spring	Full Year
Achievement Award	1,000.00	1,000.00	2,000.00
Achievement Award+	500.00	500.00	1,000.00
Leadership Scholarship- Church	500.00	500.00	1,000.00
Imaginary U - Freshman Scholarship	1,500.00	1,500.00	3,000.00

Net Costs ($24,608.00)

Cost minus grants and scholarships	Fall	Spring	Full Year
	12,304.00	12,304.00	24,608.00

Options to Pay Net Costs

Work Options ($2,200.00)

Federal/Texas Work-Study or regular campus student wages (must work to earn) ($2,200.00)

	Fall	Spring	Full Year
	1,100.00	1,100.00	2,200.00

Loan Options ($5,500.00)

	Fall	Spring	Full Year
Federal Unsubsidized Direct Loan	2,750.00	2,750.00	5,500.00

Final Net Costs ($16,908.00)

Cost of attendance minus scholarships, grants, estimated work study and loans	Fall	Spring	Full Year
	8,454.00	8,454.00	16,908.00

Other Options

- Payment Plan offered by ImaginaryU
- Military and/or National Service benefits

Imaginary U
125 years educating for success

"For I know the thoughts that I think toward you, says the Lord, thoughts of peace and not of evil, to give you a future and a hope."

Jeremiah 29:11

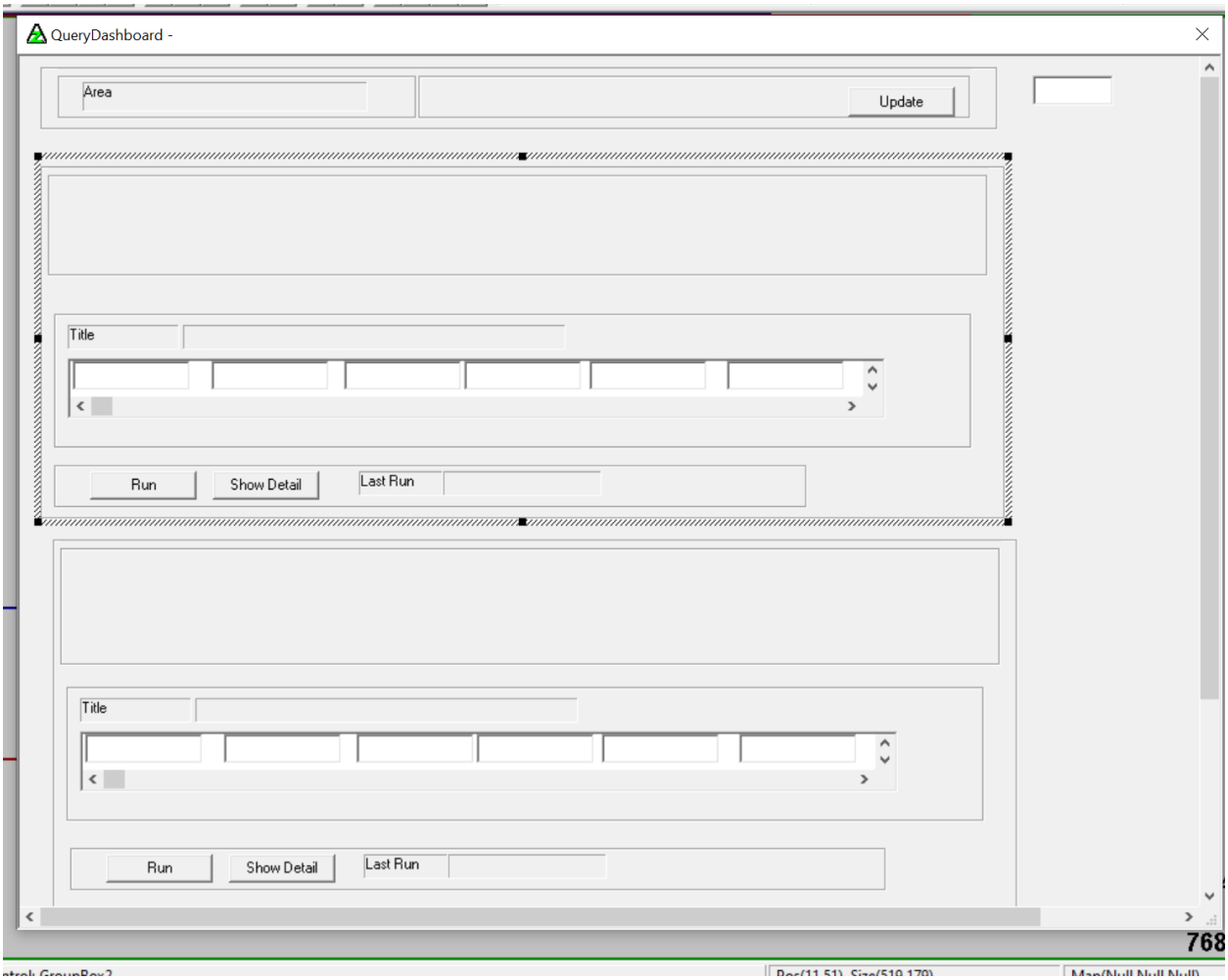
Knowledge. Faith. Service.

Display Settin

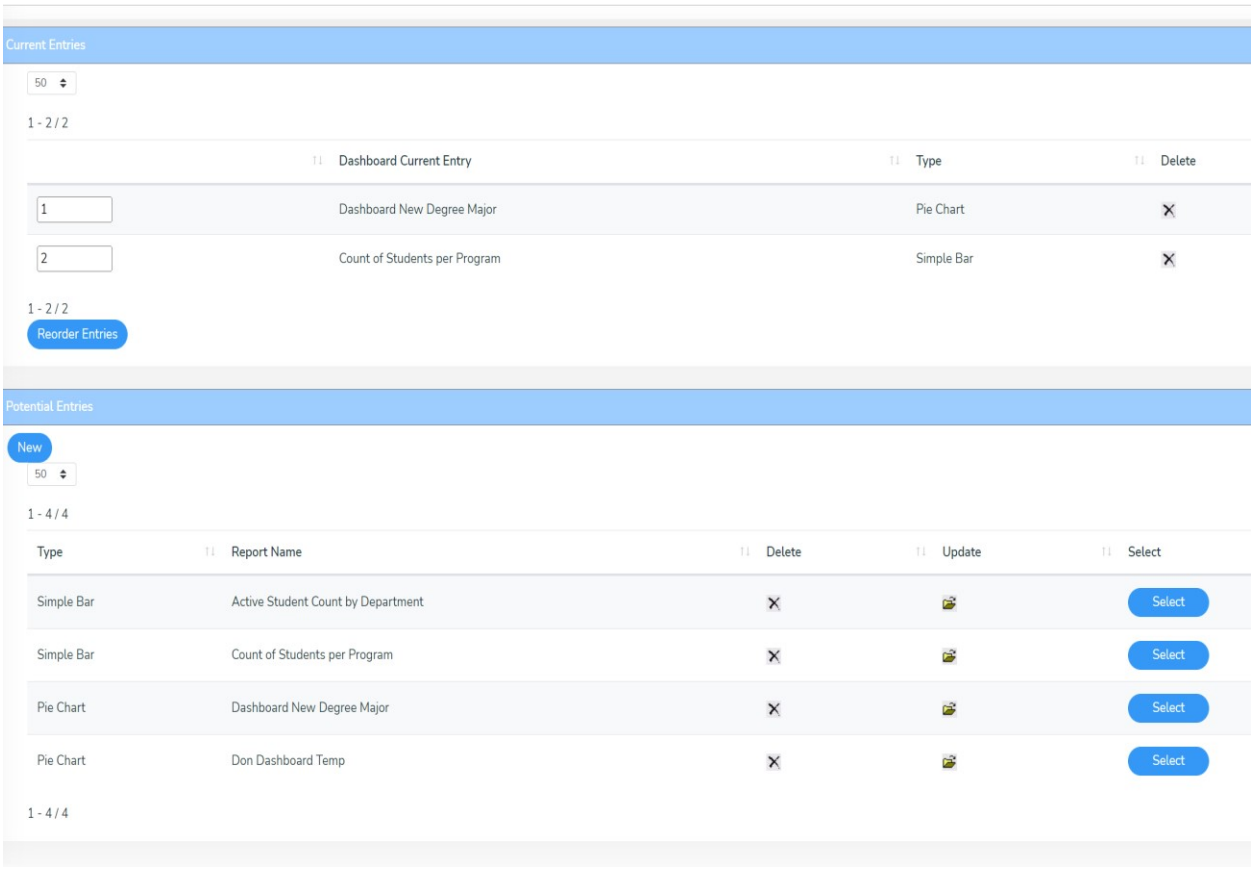
Integrated Dashboard Images

The power of the CRM Query and its ease in managing a wide variety of summary information in query objects is also available for integrating Dashboard Images into the combined target application. One or more standard Dashboard Group definitions are added to a Dialog page definition as shown in the first example below. At run time, a user can simply tie the Group to a Summary Query as shown in the second example below, producing the runtime images as show in the third example below.

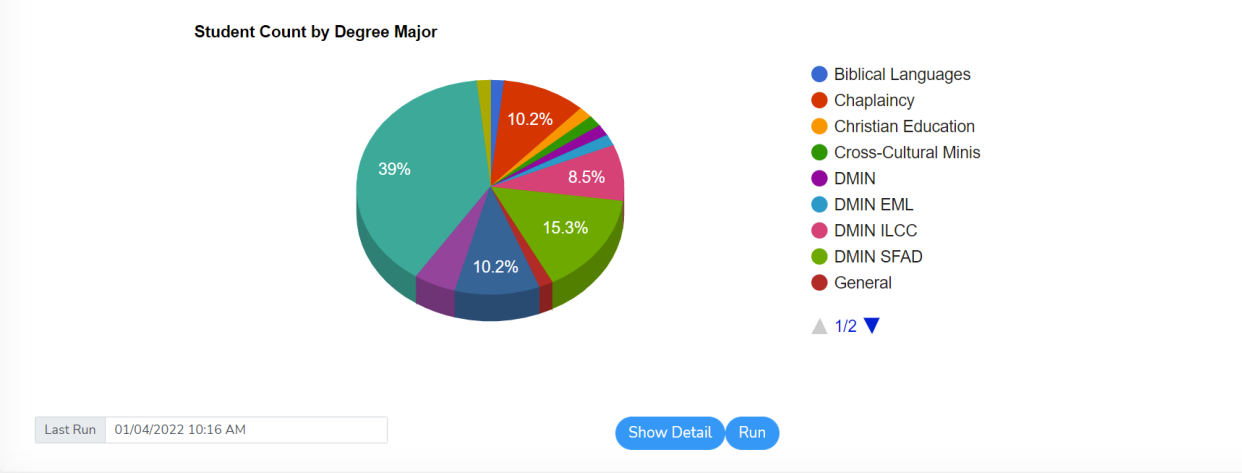
Sample Dashboard Dialog Page Definition

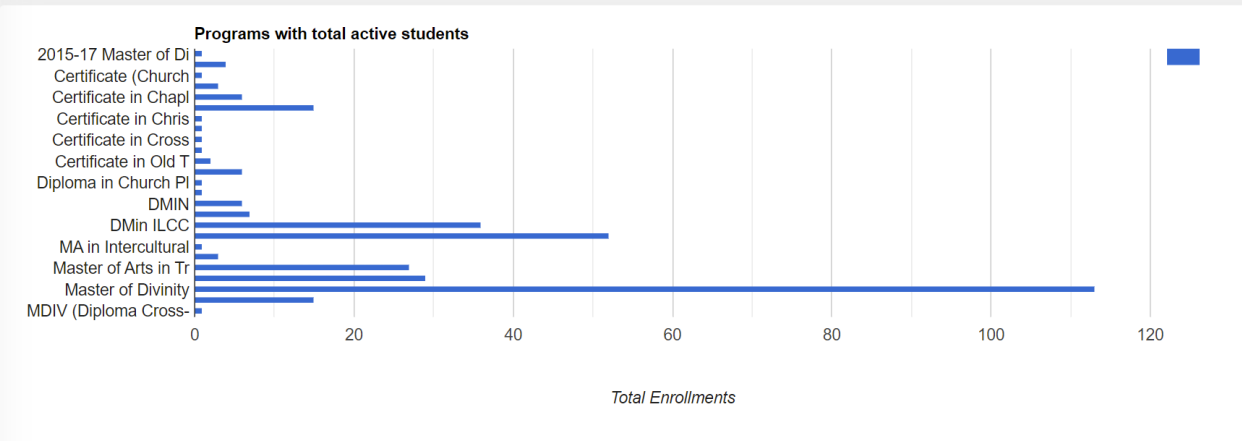


Sample Dashboard Query Entry Selection



Sample Integrated Dashboard Generated Graphics





Business Processing Rules

As has been discovered in billing interfaces used by at least one of the leading health care systems, a data tree structure lends itself to the definition of high-level business rules. That characteristic is effectively used with the information tree structures of ICDP to specify rules for a number of purposes, as shown in this section and the section that follows. Those business rules are defined in ICDP using two different techniques. The first technique specifies rules through nonprocedural definitions defined against an IC Object, as shown in the three "Nonprocedural…" examples below. The second technique utilizes procedural rules, also defined against an IC Object, as described in the "Procedural Rules" section that follows.

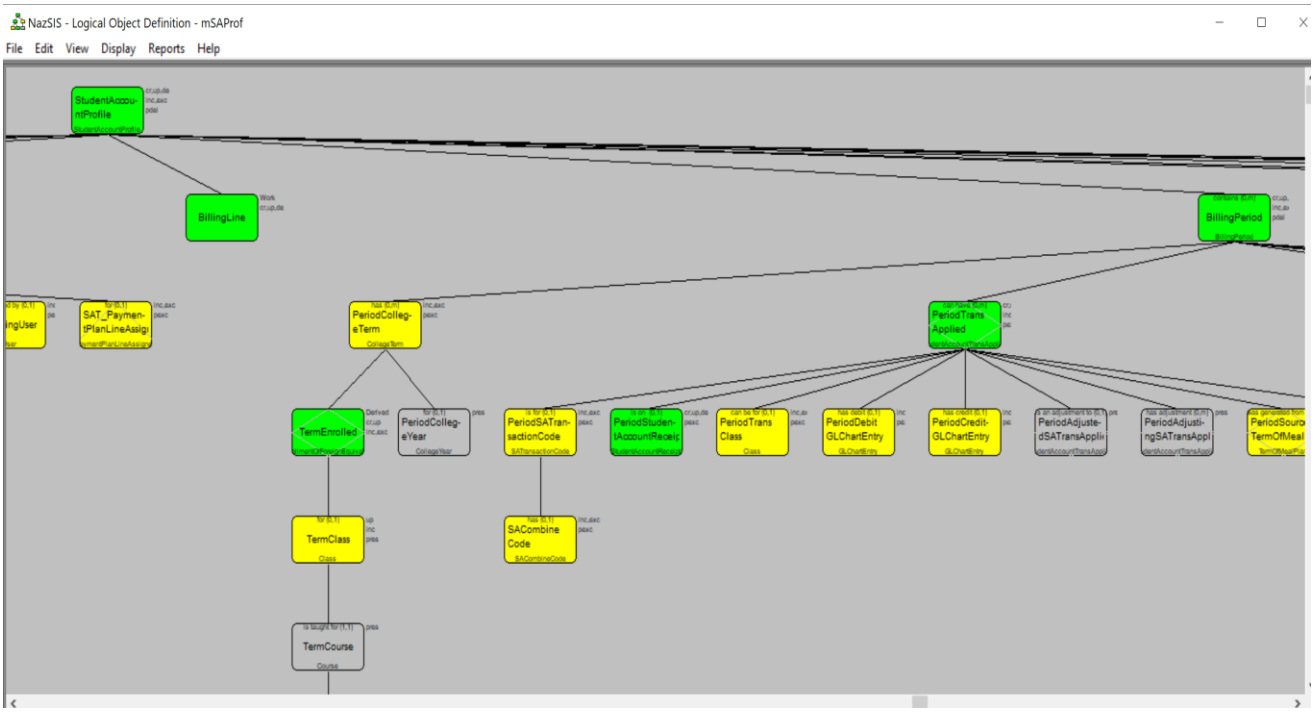
The first nonprocedural example is for the specification of a billing rule defined against the IC Object shown below under "Billing Sample IC Object." That nonprocedural example is shown on two sample pages: The sample first page, shown below under "Nonprocedural Rule Billing Example Interface," defines a Boolean rule and each component of the Boolean statement. The second sample page, shown below under "Nonprocedural Rule Query Example Interface," presents a selectable version of a billing IC object containing the information that can be used within a billing rule, along with the entity/attribute rule criteria created from that object. Entries from the "Entity/Attribute" data group are selected when adding a Boolean statement to the "Criteria" data group.

The second nonprocedural example, shown below under "Nonprocedural Rule Query Example Interface," is for a Query definition that is a part of the CRM reusable library. The interface is similar to that of billing, except that the top data group defines the data to be returned from the query and the second data group defines the Boolean rule (query criteria) which qualifies the data being returned. The third data group defines the selectable structure and data of the IC query object.

The third nonprocedural example is from the ICDP Tool Set itself and defines the rule generating a "derived attribute" for an object. Because the Tool Set runs as an MS Windows system, the layout of the window is quite different. Also, the rule structure itself is more flexible as it is comprised of a calculation set containing both a condition Boolean statement, "Rule Criteria", and a calculation statement, "Rule Calculation." The condition statement provides qualification for the calculation statement. Note that, though the use is different, the information tree structure and rule definition content are much the same.

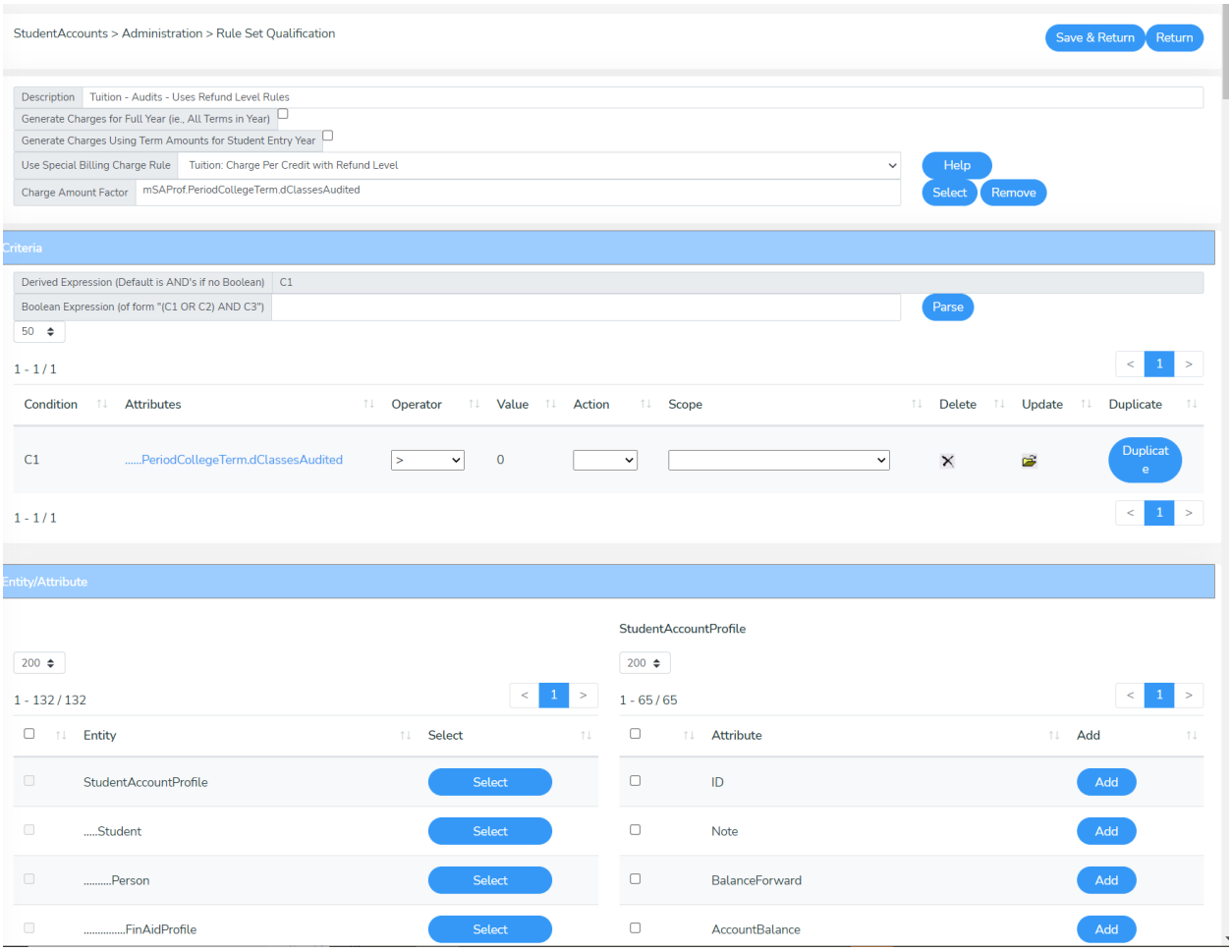
Billing Sample IC Object – SIS Student Accounts Billing

The following is part of the diagram for a rigorous IC Object for billing in a large Student Information System. The object definition (which contains more than 100 entity components) supports a visual understanding of the necessary data structures and processing rules to manage and automate a very complex billing problem. The data relationships behind such a complex problem are almost impossible to understand and automate without an effective visual image of those relationships as implemented in ICDP using the information tree structure.



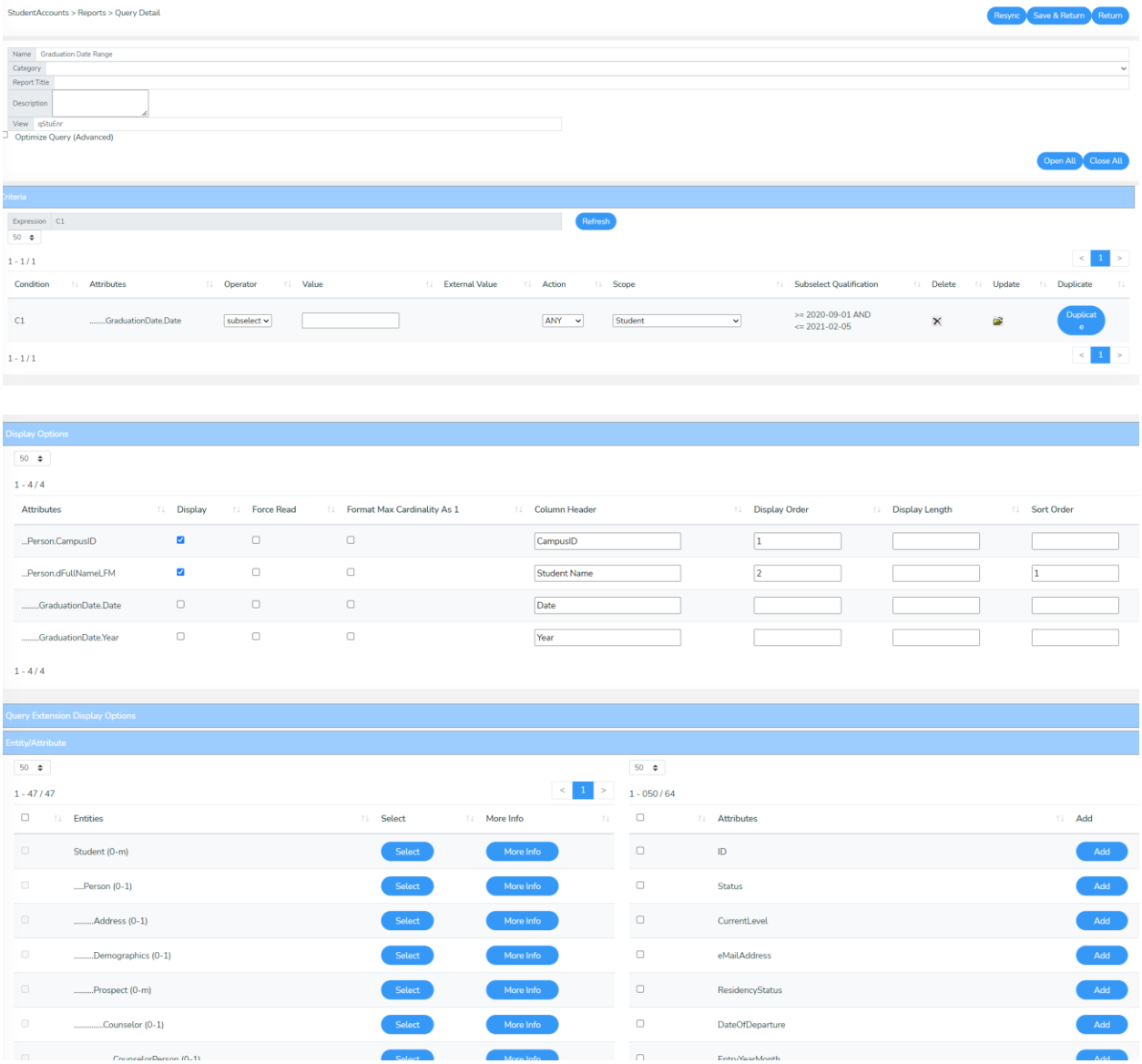
Nonprocedural Rule Billing Example Interface

As noted above, this example has two sections. The top section defines a Boolean rule and each component of the Boolean statement. The second section defines the structure and data of the IC object containing the information that can be used within a billing rule. Entries from the second section are selected to be added to the first.



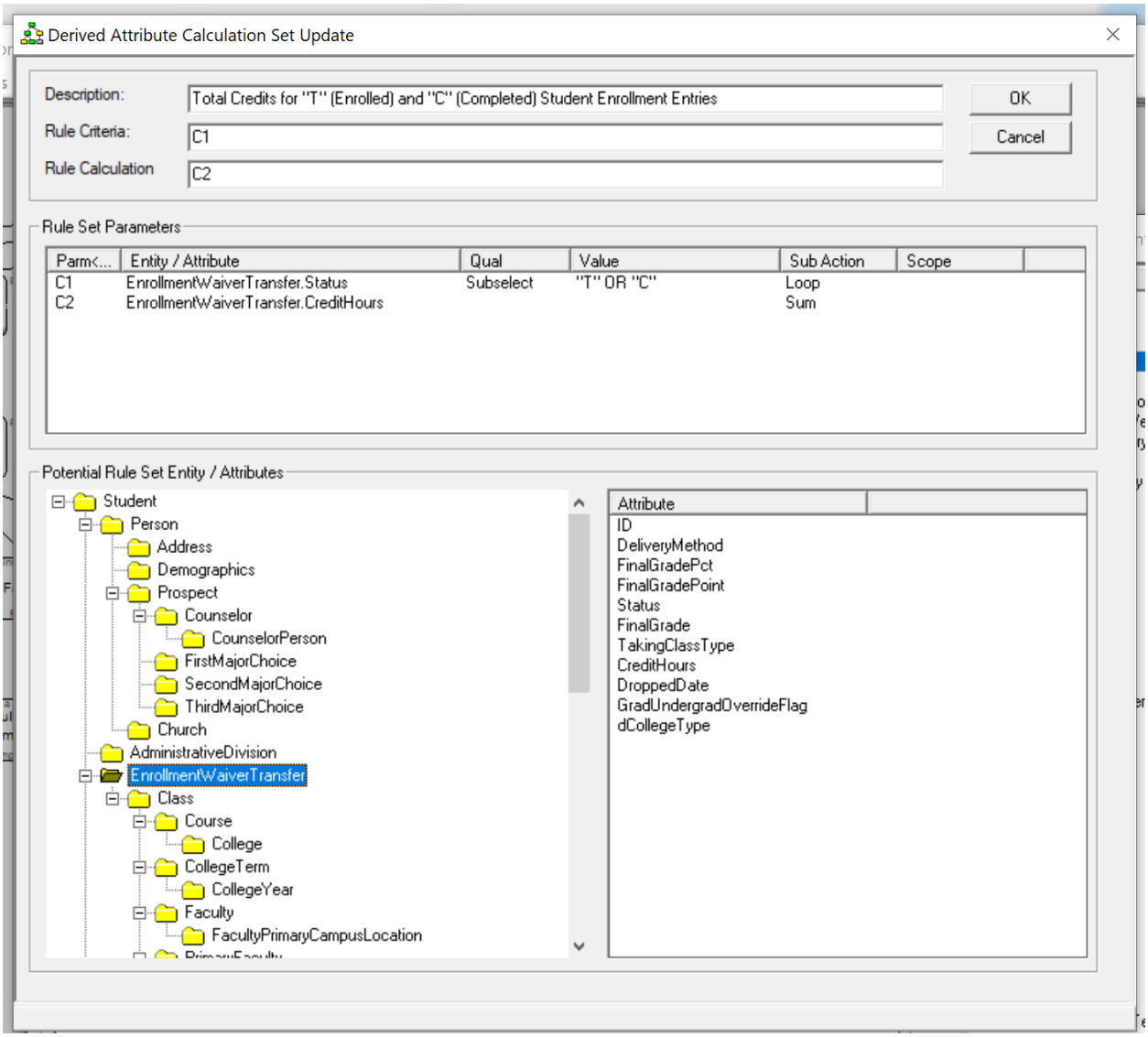
Nonprocedural Rule Query Example Interface

As noted above, this second nonprocedural example is for a Query definition that is a part of the CRM reusable library. The interface is similar to that of billing, except that the top section ("Criteria") defines the Boolean rule which qualifies the data being returned and the second section ("Display Options") defines the data to be returned from the query. The third section defines the structure and data of the IC query object.



Nonprocedural Rule Derived Attribute Interface

In the example below, the Rule Criteria, C1, identifies the repeated EnrollmentWaiverTransfer entries (“Loop”) where the Rule Calculation “Sum” is applied for Status values “T” or “C”.



Procedural Business Rules

There are limitations of the nonprocedural definitions in handling very complex rules. So, ICDP utilizes the normal structure of procedural rules (ie., program structure) to define very complex algorithms. However, it is important to note that those procedural statements reference IC Objects through that same consistent information tree structure as used in the high-level rules.

Thus an “IF” statement can reference an attribute as an object/entity/attribute combination:

```
IF mPerson.Person.LastName = “Smith”

IF mPerson.Person.LastName = lContactList.ContactItem.PersonLastName
```

A loop statement references an entity and/or entity/attribute combination:

```
FOR EACH mPerson.Address

FOR EACH mPerson.FamilyRole WHERE mPerson.FamilyRole.LivesWith = “Y”
```

Position can be changed within an object instance:

```
SET CURSOR FIRST mPerson.FamilyRole WHERE mPerson.FamilyRole.LivesWith = “Y”
```

All the data for an object instance can be accessed with a single read statement, which not only generates all the SQL necessary to access the data, but sets up a work-in-progress object area in memory to hold the data while it is being processed:

```
ACTIVATE mPerson WHERE mPerson.ID = lContactList.Person.ID
```

Also, a single statement is all that is necessary to update the database with all the modified, added and deleted data that has been accumulated in a work-in-progress object.

```
COMMIT mPerson
```

Again, although the structure of those statements is procedural, they have all the power of being defined against the information tree structure of the IC Object, producing dramatic improvements in productivity and in the understanding of complex business rules.

Conclusion

Using *Information-Centric Architecture* and the *Information-Centric Development Platform* to assemble the business applications of the future in days from high-level reusable components (instead of the months and years required by the current fractured and & fragmented techniques) will usher in a new generation of cloud business applications. The results will be superior applications, flexible in both function and technology, all created and maintained at a small fraction of the cost and time currently required to build such systems in the world within which we now operate.