DECSA Costa Rica S.A.

Database Project

Prepared for: CS542 Databases
Prepared by: Franklin Angulo

December 13, 2007

DECSA Costa Rica S.A. Escazu San Jose, Costa Rica **T** (506) 228-6969 **F** (506) 228-2372 franklin.angulo@decsacr.com www.decsacr.com

Executive Summary

Objective

This project consists of extensions to an existing database application. A new and improved database design was introduced in order to organize the data in a more efficient and productive way. This database design covers three general areas: window descriptions, security and authentication and window navigation menus. These areas are described in the following project description:

A software development company has been in the process of transforming most of its old software. It translates its original software written in PowerBuilder to the relatively new Java programming language. The software it develops is targeted to companies that need to organize and manipulate data through a database. They achieve this through the use of various types of windows which present the data from the database to the user. Then the user can select several operations to perform on the data.

On its old systems, all the windows were described in separate files because that was the architecture used in PowerBuilder. Therefore, if there were 1000 windows in the whole system, then there would be exactly 1000 files describing each and every one of these windows. This posed a severe portability issue because 1000 files would occupy a great deal of disk space (in part because PowerBuilder executable files were gigantic). Java introduced a more efficient perspective. The windows could be described completely through database tables and fields. Then there would be a "window generator" that would take the description from the database and "paint" the window exactly as it was described. But most importantly, the window would never be physically stored in the disk. Instead, since Java just generated it, the window would be stored in memory during execution time until the user decided to close the window. The only file that would have to be stored in the disk would be the Java class for generating the windows (and the rows in the database). In summary, the 1000 files that existed originally would have to be converted into 1000 rows in a database table and a single Java class file which would be the window generator. Suddenly there was a great performance boost, even if the system had 5000 windows, the new Java software would only need a single file. The company is looking for a database design that will enable them to execute this idea.

Furthermore, the company is trying to implement an improved security and authentication system. Keeping in mind that all the windows will be described in a database table, the company now wants to assign access permissions to those windows also through a database table. Therefore, for every user of the system, there should be a row in a table specifying windows to which he has access.

The final addition to its system will be a type of window navigation description through menus. In the old system, users had to know the name of the window to open it. It was a command-like interface. With this new system, users can use a menu-driven interface to find the window they need and then open it. In addition, if users have some windows that they use more frequently than others, they may want to access these through some sort of "favorite window" mechanism.

Keywords

Oracle, PostgreSQL, Java, PowerBuilder

Overview

There are three main points that need to be addressed in this project: window descriptions, security and authentication and window navigation and menus.

Window Descriptions

For this point, we first need to figure out what data needs to be retrieved from the PowerBuilder files. After gathering this data, we may want to make some extensions to capture some functionality that the PowerBuilder files might not have captured. Finally, we would have to create the database design for the windows and translate from PowerBuilder files directly to the database.

Security and Authentication

When we have all of our windows described in the database, we need to be able to assign access rights to them. The database design for users and permissions has to be created. This will most likely involve entities such as users, companies and permissions.

Window Navigation and Menus

Another of the requirements for this project is to provide an easier way to navigate all the windows for which a user has access. In other to satisfy this requirement, we will need to define a way to group the windows into menus that can then be traversed. This will include both the design and the actual implementation in Java.

Requirements List:

- **Req 01:** Create a database design for windows and their respective fields and behavior
  - This consists of defining the tables and/or views that will make up the window descriptions.

- **Req 02:** Convert the PowerBuilder files into database rows
  - A converter that reads a PowerBuilder file needs to be created. It will output SQL insert statements to be executed against the database.

- **Req 03:** Implement a window generator that will present the window on screen
  - This window generator will access the data stored in the window description tables and paint the described window on screen. This generator will be written in Java.

- **Req 04:** Create a database design for granting access rights to the windows on a user or group basis
  - A design will be created that will allow easy assignment of access rights to specific windows. The assignments can be done for a specific user.

- **Req 05:** Create a database design for navigating the windows in a menu-like fashion
  - This requirement consists of creating tables and/or views that will display the windows in a menu-like fashion. This means that some of the windows will be grouped and further sub-grouped. In this way, a tree-like structure will be created that then can be navigated.

- **Req 06:** Implement the window navigation functionality
  - After creating the design for navigating windows, a menu system will be implemented that will allow the user to very easily go through all the windows and select one. After a window is selected, the window generator from Req 02 will be invoked and the appropriate window will be presented on screen.

- **Req 07:** Test the access rights on the menu navigation
  - With the window navigation implementation in place, we need to test that the access rights assignments actually work. In order to do this, we will create some windows and assign some access rights and then check to see which windows appear in the menu navigation. Only the windows that the user has access to should appear in the menu.

Project Goals

The goals of this project go hand-in-hand with the project requirements. At the end of this project, I aim to have developed a working implementation of window descriptions all the way from their definition, to access rights and a menu implementation that will allow the user to select the window they want and make it appear on screen.

Potential User Groups

There are two types of user groups that will benefit from this project. First and foremost are the developers. With the window descriptions and access rights design, developing the complete system will be much easier for them. At least there will be a standard for storing the window descriptions. The second user group that will benefit will be the end users. They will benefit from the more user-friendly way to navigate the windows of the system.

Background Material

My work on this project was based on several specific topics. First of all, I used the ER Model theory described in Chapter 2 of the book Database Management Systems. With this I was able to create a high-level model of all the entities and relationships between them. This design was part of the solution that was delivered to the software company. Apart from this, I drew upon aspects of the DDLs and DMLs of both the Oracle and PostgreSQL DBMS. There were several differences between these two that needed to be handled appropriately. For instance, there was a very specific difference between the DDL in Oracle and the DDL in PostgreSQL that did not have the same syntax for specifying an outer join when creating a view.

The topics listed previously deal with the DBMS and the data that had to be stored in them. The next step was to retrieve the data from data files and convert them into insert statements for the database. Usual file operations written in the Java programming language were used to scan files, retrieve and organize the data, and generate DML files that would then be executed against the DBMS tables. In order to interact with the data (modify, delete, insert), the JDBC drivers were used in the developed Java application in order to communicate with the DBMS. Finally, some knowledge of Java Swing was required to develop the user interface and components such as the window navigation tree and the window generator that displays windows and their appropriate fields.

DBMS Description

For my project I used two different DBMS: Oracle and PostgreSQL. The Oracle DBMS is a relational database management system software product released by Oracle Corporation and it has become a major player in the database market. PostgreSQL is a powerful, open source relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity and correctness.

The software that my company develops has two very distinct layers. In one layer we have all the presentation logic for describing how the user interface should look and function. The second layer deals with the business data that is stored by using the user interface. When a system is delivered to a client, the presentation layer contains all the logic of the software company that specifies how to handle insertions, deletions and modifications in a window. But there is no business data present at the beginning because the client has not started using the system. Without a doubt, the business data layer is the most important part and it should be protected. At the beginning we thought about putting both the presentation layer and the business data layer in a single Oracle DBMS. This was working perfectly until we thought about remote execution of the system. If a client had to travel and wanted to access the system using a slow internet connection, it would not be very pleasant to work. Apart from making a request to the remote database for the business data that was queried, the client would also have to request the presentation logic. Therefore, when a client would try to open a window, he would first have to wait a long time for the window to open. Then he would insert his query parameters and execute the query to request the business data which would also take a long time.

We decided to keep the business data in a centrally-located Oracle DBMS to ensure security of the information. But the presentation logic could be stored in each client. For example, we could create a database with the presentation logic in each of the laptops that would be used for travel. Then, when a client used the system, the presentation logic would be obtained from his local copy of the presentation database and the business data would be obtained from the centrally-located database. Our clients have very low IT budgets, so asking them to install an Oracle database in each of their laptops would be unreasonable. In addition, Oracle can be a very heavy database that would probably not function in a laptop with limited resources. Therefore, we decided to go with open-source PostgreSQL in the laptops. This added some complexity to the system because we had to handle several different databases, but it solved the great problem of remote execution of the system. PostgreSQL was also compared to mySQL, but we decided that the first was much more robust. At the time, mySQL did not support stored procedures nor views, so PostgreSQL was our clear choice.

Approach

Window Descriptions

There are three aspects that we need to address. This first aspect deals with the descriptions of the windows of the system. We plan to solve this problem using nine tables and two views. Our first table will be COM_VENTAN. Each row of the COM_VENTAN table will register a new window for the system. It will contain information such as the title, name and dimensions of the window. Related to the COM_VENTAN table will be the COM_AREDAT table. If we look at the layout of most of the windows of the system, we can notice that there are several different areas in a window. For example, in some of the accounting windows we may have a top area that selects the number of a receipt. Once the receipt number is selected, the middle area of the window is populated with all the entries for that receipt (e.g. what was bought). Finally, in the lowest area the sum of all the items appears. For this reason, each row of the COM_AREDAT table will describe the areas of the window. If there are three areas as in this case, there would be three different rows specifying three areas for the single window.

If we further subdivide the problem, we find that each area can have fields (e.g. textboxes, checkboxes, textareas) and labels (e.g. the titles corresponding to a field). Therefore, we will create the tables COM_CAMARE and COM_ETQARE for the fields and the labels for each area of a window. This allows for fields without a label for example. As a result of the division we made between fields and labels, we can apply different formatting to each of these. For example, we might want the label to be bold and blue and the field to have normal text of size 9pt and in Arial font. In order to do this, two more tables were created: COM_ATRCAM and COM_ATRETQ. These two tables correspond to the attributes or formatting details for fields and labels, respectively.

The final aspect that needed to be addressed in this window descriptions section is internationalization. For labels in an area of a specific window, we might want them to be flexible enough so that if the locale was changed, so could the text the was displayed in the label. COM_IDIOMA was created to keep track of all the available locales in the systems (e.g. English, Spanish, French, etc.). Then, in COM_INFCAM and COM_INFETQ we specify the translations for each of the localized objects. COM_INFETQ stores all the titles in each of the different locales for each of the available labels of the system. COM_INFCAM also stores the tooltip texts in each of the different locales for each of the available fields. Then, two views are created; COM_IDICAM and COM_IDIETQ. The COM_IDICAM view joins that data from COM_CAMARE and COM_INFCAM by locale. Basically, it takes the data stored for each of the fields for that area and sticks the correct tooltip in the current locale. COM_IDIETQ does the same but for titles; it joins COM_ETQARE and COM_INFETQ by locale. Therefore, when the system presents a window it retrieves the tooltips and label text in the current locale using the data from the views COM_IDICAM and COM_IDIETQ.

Security and Authentication

The second aspect we need to address with our database design is the security and authentication issue. All the windows of the system are registered in the COM_VENTAN table and described in the COM_AREDAT, COM_CAMARE, COM_ETQARE and all the other tables. Now the company wants to assign access permissions to those windows also through a database table. For every user of the system, there should be a row in a table specifying the windows to which he has access.

We will first have COM_COMPAN to keep track of all the companies registered in the system. Then we will create COM_USUARI to keep track of all the users of the system. With companies and users in place, now we create the permissions table called COM_PERMIS. This table will receive a company, a user, a window references and a permission code. In essence, a row in COM_PERMIS specifies that a user of a system under the specified company has the specified permissions (a combination of QIMDP) for the specified window that is references from COM_VENTAN.

QIMDP is an abbreviation that the software company has created to deal with operations that can be performed on database data through their windows.

Q	Perform (Q)ueries
I	(I)nsert a new row of data
M	(M)odify an existing row of data
D	(D)elete an existing row of data
P	(P)rint data from a window

I thought about creating a separate table for specifying the available permission codes. That is, all the possible combinations of QIMDP. But this makes the design more complex than necessary and we would have to join two tables to find out what the actual permission code would represent (e.g. if we assigned code 100 to permissions QMP, we would have to join both tables to find out what this meant). So, I kept it simple and allowed for direct insertion of a combination of QIMDP into the COM_PERMIS table.

This is the solution for granting access to users for a specific window. The next step was to find a way around having to input permissions repeatedly. Take this example: I have a group of developers called SwingDevelopers that all have the same exact permissions for several windows. They all have access to the same windows and for the same operations. Now, a new employee is hired into the SwingDevelopers group. How do I go about inserting permissions for the new employee knowing that all SwingDevelopers have the same permissions? Under the current design, I would have to go row by row duplicating the data while only changing the username. This would be a pain if we had, say, we had 5000 permission entries. I would be easier if we could say that the new employee inherits all the permissions of the group. To allow for this a new table was created: COM_USRDEP.

In the COM_USRDEP table we can specify that for a given company, user X is parent of user Y. As we will see later, this relationship will be used to determine permissions based on inheritance. This table not only allows for specifying direct inheritance from user to user, but we can also created groups of users. For example, we can define the group SwingDevelopers. We set this group as the parent and then for each member we create a new tuple setting the member as a child of SwingDevelopers. In order for our design to work correctly, we have to note something. For each user that we want to control using COM_USRDEP, we have to specify that it is a parent of itself. So, for example, SwingDevelopers will be a parent of itself and so will every member of SwingDevelopers. If we set Mary as a child of SwingDevelopers, Mary will also be a parent of herself.

As a final step, we create the view COM_PERUSR. This view will combine the data from COM_PERMIS and COM_USRDEP and will expose the inheritance functionality that we were looking for. This view is probably the most complex part of the database design. Apart from inheritance, this view will also allow for the concept of negative permissions. Take our example of SwingDevelopers. If the newly hired employee should have access to almost all windows except for one or two, it makes sense to apply the inheritance relationship to provide him access to all windows. Then we can specify negative permissions to “un-grant” him access to the one or two windows that he should not have access to. We do this by specifying an X in the permissions field of COM_PERMIS. For example, user Mary will have access to window W with permissions QIMDX. This allows Mary to query, insert, modify and delete, but NOT print using window W. The view COM_PERUSR notices that a child has been assigned a permission in COM_PERMIS and gives higher precedence to the child's permission that to the permission inherited from the parent. Therefore, even if the parent's permission for window W was QIDMP, the child will now have permission QIDMX. This will effectively execute the concept of negative permissions. In this case, we are taking away Mary's permission to print on window W.

Window Navigation Menus

Our last aspect is the creation of a window navigation interface for the users interacting with our system. First we will describe what we mean when we talk about a window navigation interface. The software company’s system has thousands of windows divided into subsystems. Each subsystem is divided further into groups or menus. For example, the Employees and Payroll subsystem may have a group called Managers and another called Employees. These groups are represented as menus. Inside the Managers menu there will be several windows with which we can work. The Employee menu also has some windows. There could exist a monthly salary window in the Employee menu that lets the employee query how much his salary for this month will be after taxes and other deductions. A window called Deductions may also exist under the Managers menu that lets the manager include deductions to the monthly salary of an employee. Now, this simple menu structure will look like this:

menu0	menu1	menu2	menu3	menu4
Start Menu	Employees & Payroll	Managers	Deductions (W)	
		Employees		Monthly Salary (W)

This is how the window navigation for our system should work. When a user enters the system, he starts in the "Start" menu. At this point he sees the Employees and Payroll subsystem menu. Inside this subsystem he sees both the Managers and Employees menus. He can now select the Managers menu and then open the Deductions window to perform some changes. After examining this structure, we find a very important problem. Should an employee be able to see and access the Managers menu? This could lead to some serious problems. Therefore, we need to grant access permissions and restrictions. If John is an employee, he should only see the Employees menu when he enters the system. On the other hand, if Jake is a manager he should only be able to see the Managers menu. We will provide this functionality using the COM_PERMIS table and a new table we will create named COM_NAVEGA.

COM_NAVEGA will describe all the possibilities for navigation through the system. Referring to our example, it will say that from menu0 the user can see menu1. If the user then goes into menu1, then he can see menu2, and so on. Now, when the user enters menu3, he does not see any menus. He sees one window instead, Deductions. We need to distinguish between windows and menus. Menus are used to group windows and other menus. Windows are registered in the COM_VENTAN table. If you select a menu, our system will open the options available under that menu which could include windows or more submenus. On the other hand, if you select a window, the system will automatically generate that window using the descriptions from COM_VENTAN, COM_AREDAT, COM_CAMARE, COM_ETQARE, etc. and then present it to the display so the user can perform operations on the data. We will refer to our example again to illustrate this point.

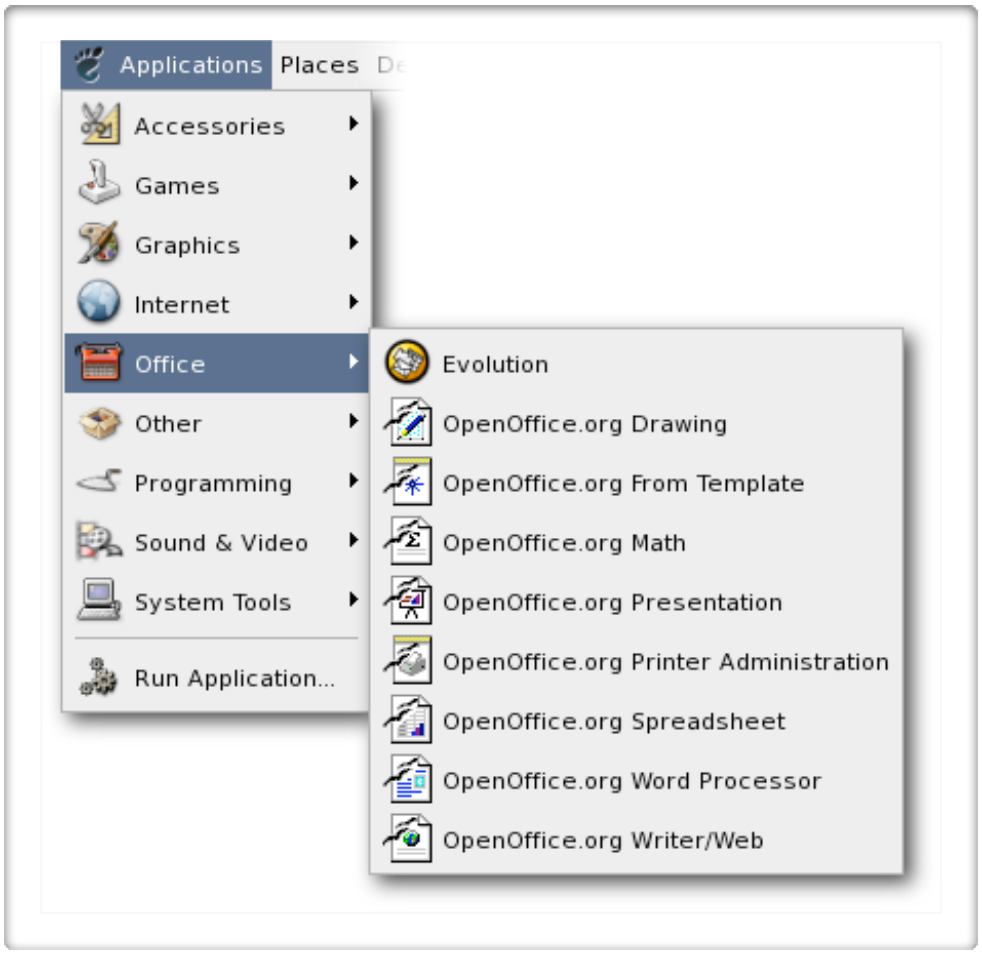
FROM	TO	TYPE
menu0	menu1	Menu
menu1	menu2	Menu
menu2	menu3	Menu
menu2	menu4	Menu
menu3	Deductions	Window
menu4	Monthly Salary	Window

This table above is a representation of how the COM_NAVEGA table would look. We define the rules of navigation in this table. We can say that the user may go from menu0 to another menu, menu1, using navigation of type "Menu". In such a case, the system would open the available options under menu1. We may also allow the user to go from a menu, menu3, to a window, Deductions, using navigation of type "Window". In this case, the system would automatically generate the Deductions window and present it to the display. We can find the definition of menu1 by querying the database and asking it what elements have FROM equal to menu1. If we print the distinct TITLE attributes for these elements we would get a list of elements that make up the menu1 definition.

This table will have a one-to-many relationship with COM_VENTAN. This is because the attribute TO_WHERE may contain the name of a window. For any window registered in the COM_WINDOW table there could be many navigation rules that include it.

Another more graphical example of the difference between menus and windows is the following. The first menu that the user selects is the one named Applications. When the user selects this menu, the system opens all the options under that menu which include Accessories, Games, Graphics, etc. Next the user goes down to the Office option and selects it. Office is also a menu as we can see from the arrow at the right of its name. Therefore, when the user selects it, all possible options will appear; Evolution, OpenOffice.org Drawing, OpenOffice.org from Template, etc. Now, say the

user selects the OpenOffice.org Word Processor option from within this menu. As we can see, this option does not have an arrow to the right of its name; thus it is a window or “runnable” option. So when the user selects this option, the Word Processor application will load so the user can work on a document. This is the same menu-driven functionality that our system will have.

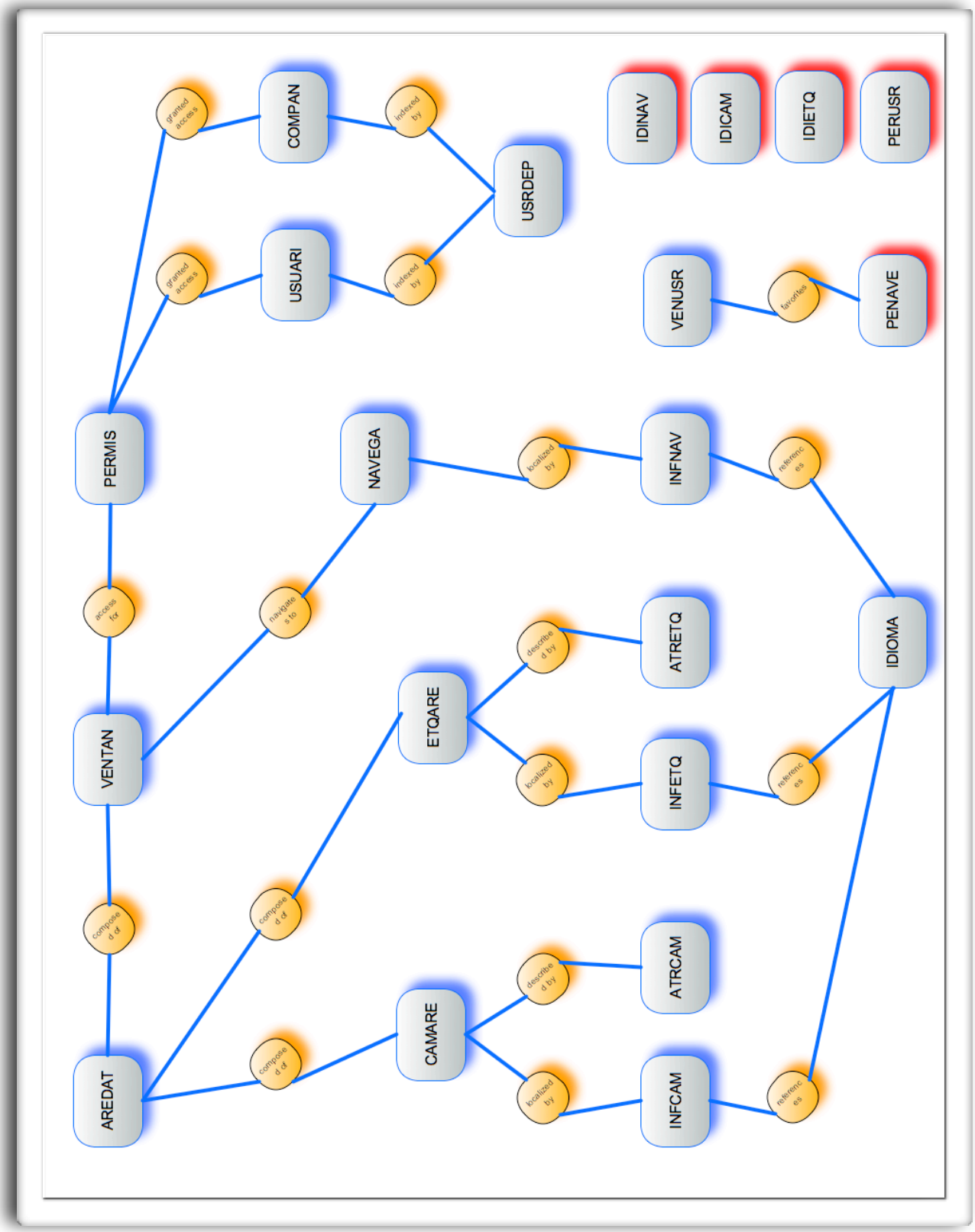


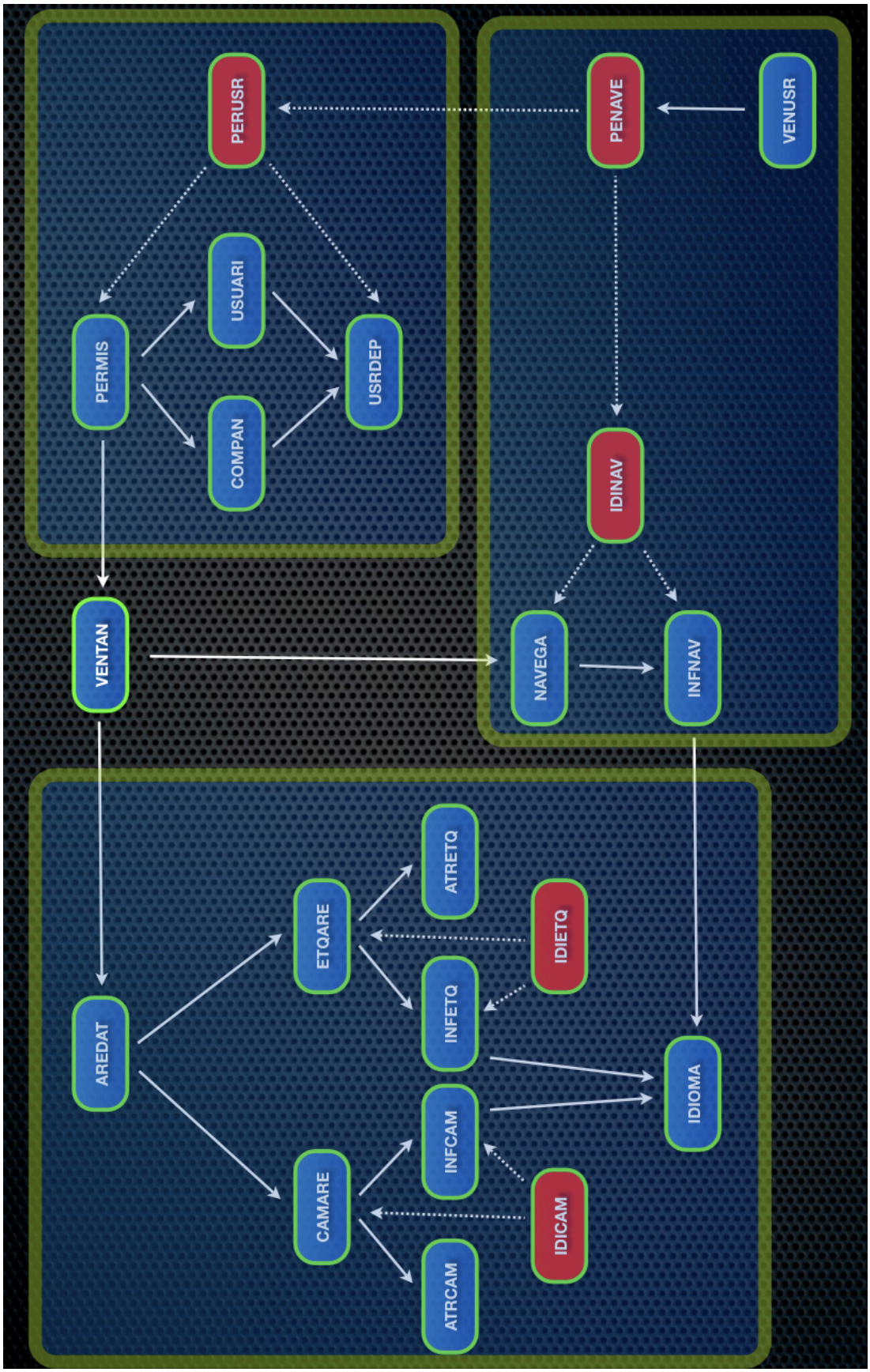
With COM_NAVEGA in place, we now create COM_INFNAV. This table is used to localize all the options of the menu navigation. So for each caller/callee combination we give the appropriate title translation in each locale. This is a very similar design to the one used to localize the label titles and field tooltips in the window descriptions section. Then the view COM_IDINAV combines COM_NAVEGA and COM_INFNAV using the required locale. During execution of the system, we pass in the current locale as parameter and get the correctly localized version of the menu navigation options.

In the previous section we defined how to specify access rights to specific windows. These permissions need to be portrayed in the menu navigation options. That is, only the windows for which the user has access to should appear. The others should be hidden. To do this, we create a new view: COM_PENAVE, navigation permissions. This view will join the data in COM_PERUSR and COM_IDINAV relations. It joins with COM_PERUSR because it defines all the permissions for all the users. It joins with COM_IDINAV because it specifies all the navigation options. In the end, we have a view that combines permissions with navigation options and this is our solution.

The software company also emphasized that many users could complain about using their window navigation structure for doing day-to-day work. For example, John checks the Monthly Salary Query window every day. He would like to avoid the process of navigation through all the menus. Instead he would like to have a “favorite window” utility in which he could include the Monthly Salary Query window. The software company expressed their desire for this functionality to be included in the database design since more and more users are asking for this. We will create a COM_VENUSR table.

We have four attributes for the COM_VENUSR table. We need to record the company and the user. These will reference the COM_COMPAN and COM_USUARI tables for integrity. Then we need to record the caller and callee. With these two attributes we index into the COM_NAVEGA table to find the specific window that was flagged as a favorite window. Now, a single window can be a favorite for many users and of course, many users may have many favorite windows.


IDINAV
IDICAM
IDIETQ
PERUSR
COMPAN
USRDEP
USUARI
VENUSR
PENAVE
PERMIS
NAVEGA
INFNAV
IDIOMA
VENTAN
ATRETQ
ETQARE
INFETQ
AREDAT
ATRCAM
CAMARE
INFCAM
granted access
indexed by
favorites
access for
navigates to
localized by
referenc es
describe d by
compose d of


PERUSR
PERMIS
USUARI
USRDEP
COMPAN
PENAVE
VENUSR
IDINAV
VENTAN
NAVEGA
INFNAV
AREDAT
ETQARE
ATRETQ
IDIETQ
INFETQ
IDIOMA
CAMARE
INFCAM
IDICAM
ATRCAM

Field Requirements

underline: primary key
italic: foreign key

Window Descriptions

COM_VENTAN(NOM_VENTAN, DES_VENTAN)

COM_AREDAT(NOM_VENTAN, NOM_ARE_DA, NOM_EST_BA, NOM_EST_AC, TIP_PRESEN, VAL_ALTURA, VAL_LARGO)

COM_CAMARE(NOM_VENTAN, NOM_ARE_DA, NOM_CAMPO, DES_CAMPO, SEC_ORDENA, TIP_DATO, LON_PRESEN, FOR_PRESEN, LON_CAMPO, TIP_PRESEN, UBICA_X, UBICA_Y, ALT_CAMPO, NUM_ORDENA, TIP_ORDENA, IND_ACTUAL, IND_MODIFI, IND_CONSUL, IND_OCULTO, IND_OBLIGA, IND_LLAVE, IND_ATR_ES, SEC_LLAVE)

COM_ETQARE(NOM_VENTAN, NOM_ARE_DA, NUM_ETIQUE, ALT_ETIQUE, LON_ETIQUE, UBICA_X, UBICA_Y)

COM_ATRCAM(NOM_VENTAN, NOM_ARE_DA, NOM_CAMPO, TIP_FUENTE, TAM_FUENTE, COL_FUENTE, COL_FONDO, IND_NEGRIT, IND_CURSIV, IND_SUBRAY, TIP_JUSTIF, TIP_BORDE)

COM_ATRETQ(NOM_VENTAN, NOM_ARE_DA, NUM_ETIQUE, TIP_FUENTE, TAM_FUENTE, COL_FUENTE, COL_FONDO, IND_NEGRIT, IND_CURSIV, IND_SUBRAY, TIP_JUSTIF, TIP_BORDE)

COM_INFCAM(NOM_VENTAN, NOM_ARE_DA, NOM_CAMPO, COD_IDIOMA, DES_CAMPO)

COM_INFETQ(NOM_VENTAN, NOM_ARE_DA, NUM_ETIQUE, COD_IDIOMA, TIT_ETIQUE)

COM_IDIOMA(COD_IDIOMA, DES_IDIOMA)

COM_IDICAM(NOM_VENTAN, NOM_ARE_DA, NOM_CAMPO, SEC_ORDENA, TIP_DATO, LON_PRESEN, FOR_PRESEN, LON_CAMPO, TIP_PRESEN, UBICA_X, UBICA_Y, ALT_CAMPO, NUM_ORDENA, TIP_ORDENA, IND_ACTUAL, IND_MODIFI, IND_CONSUL, IND_OCULTO, IND_OBLIGA, IND_LLAVE, IND_ATR_ES, SEC_LLAVE, COD_IDIOMA, DES_CAMPO)

COM_IDIETQ(NOM_VENTAN, NOM_ARE_DA, NUM_ETIQUE, ALT_ETIQUE, LON_ETIQUE, UBICA_X, UBICA_Y, IND_ATR_ES, COD_IDIOMA, TIT_ETIQUE)

Security and Authentication

COM_COMPAN(COMPANIA, DES_COMPAN)

COM_USUARI(NOM_USUARI, DES_USUARI)

COM_PERMIS(COMPANIA, ABR_FORMAT, NOM_AUTORI, PERMISOS)

COM_USRDEP(COMPANIA, USR_PADRE, USR_HIJO, NOM_USUAR)

COM_PERUSR(COMPANIA, ABR_FORMAT, NOM_AUTORI, PERMISOS, USR_PADRE, USR_HIJO)

Window Navigation Menus

COM_NAVEGA(LLAMANTE, LLAMADO, NOM_CAMPO, SECUENCIA, DES_LLAMAD, TIP_LLAMAD)

COM_VENUSR(COMPANIA, NOM_USUAR, LLAMADO, LLAMANTE, SEC_ORDENA)

COM_INFNAV(LLAMANTE, LLAMADO, COD_IDIOMA, TITULO)

COM_IDINAV(LLAMANTE, LLAMADO, NOM_CAMPO, SECUENCIA, DES_LLAMAD, TIP_LLAMAD, OPE_BITACO, ENCADENADO, AUX_LLAMAD, COD_ID_NAV, COD_IDIOMA, TITULO)

COM_PENAVE(COMPANIA, NOM_AUTORI, PERMISOS, LLAMANTE, LLAMADO, TITULO, NOM_CAMPO, SECUENCIA, DES_LLAMAD, TIP_LLAMAD, OPE_BITACO, ENCADENADO, COD_ID_NAV, AUX_LLAMAD, COD_IDIOMA, NOM_VEN_BA, TIP_PRESEN, VAL_ALTURA, VAL_LARGO, MAX_REG_CO)

SQL Statements

In this section, I list the SQL statements for a PostgreSQL database which is what I've used for implementation. But our system will potentially support any database, so the statements will translated to fit the specific DBMS if needed.

Window Descriptions

COM_VENTAN

```
create table com_ventan
(
        nom_ventan      varchar(15)     not null,
        des_ventan      varchar(240)    ,
        constraint com_ventan_llave primary key (nom_ventan)
)
without oids;
alter table com_ventan owner to dec;
grant all on table com_ventan to dec;
grant all on table com_ventan to public;
```

COM_AREDAT

```
create table com_aredat
(
        nom_ventan      varchar(15)     not null,
        nom_are_da      varchar(15)     not null,
        nom_est_ba      varchar(10)     ,
        nom_est_ac      varchar(10)     ,
        tip_presen      varchar(3)      not null,
        val_altura      numeric(7)      not null,
        val_largo       numeric(7)      not null,
        constraint com_aredat_llave primary key (nom_ventan, nom_are_da),
        constraint com_aredat_fllave foreign key (nom_ventan) references com_ventan
)
without oids;
alter table com_aredat owner to dec;
grant all on table com_aredat to dec;
grant all on table com_aredat to public;
```

COM_CAMARE

```
create table com_camare
(
        nom_ventan      varchar(15)     not null,
        nom_are_da      varchar(15)     not null,
        nom_campo       varchar(10)     not null,
        des_campo       varchar(240)    ,
        sec_ordena      numeric(7)      not null,
        tip_dato        varchar(3)      not null,
        lon_presen      varchar(6)      not null,
        for_presen      varchar(20)     not null,
        lon_campo       varchar(6)      not null,
        tip_presen      varchar(1)      not null,
        ubica_x         numeric(7)      not null,
        ubica_y         numeric(7)      not null,
        alt_campo       numeric(7)      not null,
        num_ordena      numeric(7)      ,
        tip_ordena      varchar(1)      ,
        ind_actual      varchar(1)      not null,
        ind_modifi      varchar(1)      not null,
        ind_consul      varchar(1)      not null,
        ind_oculto      varchar(1)      not null,
        ind_obliga      varchar(1)      not null,
        ind_llave       varchar(1)      not null,
        ind_atr_es      varchar(1)      not null,
        sec_llave       numeric(7)    ,
        constraint com_camare_llave primary key (nom_ventan, nom_are_da, nom_campo),
        constraint com_camare fllave1 foreign key (nom_ventan, nom_are_da) references com_aredat
)
without oids;
alter table com_camare owner to dec;
grant all on table com_camare to dec;
grant all on table com_camare to public;
```

COM_ETQARE

```
create table com_etqare
(
        nom_ventan      varchar(15)     not null,
        nom_are_da      varchar(15)     not null,
        num_etique      numeric(7)      not null,
        alt_etique      numeric(7)      not null,
        lon_etique      numeric(7)      not null,
        ubica_x         numeric(7)      not null,
        ubica_y         numeric(7)      not null,
        constraint com_etqare_llave primary key (nom_ventan, nom_are_da, num_etique),
        constraint com_etqare fllave1 foreign key (nom_ventan, nom_are_da) references com_aredat
)
without oids;
alter table com_etqare owner to dec;
grant all on table com_etqare to dec;
grant all on table com_etqare to public;
```

COM_ATRCAM

```
create table com_atrcam
(
        nom_ventan      varchar(15)     not null,
        nom_are_da      varchar(15)     not null,
        nom_campo       varchar(10)     not null,
        tip_fuente      varchar(40)     not null,
        tam_fuente      numeric(7)      not null,
        col_fuente      varchar(15)     not null,
        col_fondo       varchar(15)     not null,
        ind_negrit      varchar(1)      not null,
        ind_cursiv      varchar(1)      not null,
        ind_subray      varchar(1)      not null,
        tip_justif      varchar(1)      not null,
        tip_borde       varchar(1)      not null,
        constraint com_atrcam_llave primary key (nom_ventan, nom_are_da, nom_campo),
        constraint com_atrcam_fllave1 foreign key (nom_ventan, nom_are_da, nom_campo) references
com_camare
)
without oids;
alter table com_camare owner to dec;
grant all on table com_atrcam to dec;
grant all on table com_atrcam to public;
```

COM_ATRETQ

```
create table com_atretq
(
        nom_ventan      varchar(15)     not null,
        nom_are_da      varchar(15)     not null,
        num_etique      varchar(10)     not null,
        tip_fuente      varchar(40)     not null,
        tam_fuente      numeric(7)      not null,
        col_fuente      varchar(15)     not null,
        col_fondo       varchar(15)     not null,
        ind_negrit      varchar(1)      not null,
        ind_cursiv      varchar(1)      not null,
        ind_subray      varchar(1)      not null,
        tip_justif      varchar(1)      not null,
        tip_borde       varchar(1)      not null,
        constraint com_atretq_llave primary key (nom_ventan, nom_are_da, num_etique),
        constraint com_atretq_fllave2 foreign key (nom_ventan, nom_are_da, num_etique) references
com_etqare
)
without oids;
alter table com_camare owner to dec;
grant all on table com_atretq to dec;
grant all on table com_atretq to public;
```

COM_INFCAM

```
create table com_infcam
(
        nom_ventan      varchar(15)     not null,
        nom_are_da      varchar(15)     not null,
        nom_campo       varchar(10)     not null,
        cod_idioma      varchar(6)      not null,
        des_campo       varchar(240)    not null,
        constraint com_infcam_llave primary key (nom_ventan, nom_are_da, nom_campo, cod_idioma),
        constraint com_infcam_fllave3 foreign key (nom_ventan, nom_are_da, nom_campo) references
                                                                                com_camare,
        constraint com_infcam_fllave4 foreign key (cod_idioma) references com_idioma
)
without oids;
alter table com_infcam owner to dec;
grant all on table com_infcam to dec;
grant all on table com_infcam to public;
```

COM_INFETQ

```
create table com_infetq
(
        nom_ventan      varchar(15)     not null,
        nom_are_da      varchar(15)     not null,
        num_etique      numeric(7)      not null,
        cod_idioma      varchar(6)      not null,
        tit_etique      varchar(240)    not null,
        constraint com_infetq_llave primary key (nom_ventan, nom_are_da, num_etique, cod_idioma),
        constraint com_infetq_fllave3 foreign key (nom_ventan, nom_are_da, num_etique) references
                                                                                com_etqare,
        constraint com_infetq_fllave4 foreign key (cod_idioma) references com_idioma
)
without oids;
alter table com_infetq owner to dec;
grant all on table com_infetq to dec;
grant all on table com_infetq to public;
```

COM_IDIOMA

```
create table com_idioma
(
        cod_idioma      varchar(6)      not null,
        des_idioma      varchar(240)    not null,
        constraint com_idioma_llave primary key (cod_idioma)
)
without oids;
alter table com_idioma owner to dec;
grant all on table com_idioma to dec;
grant all on table com_idioma to public;
```

COM_IDICAM

```
create view com_idicam
as select
   com_camare.nom_ventan          as nom_ventan  ,
   com_camare.nom_are_da          as nom_are_da  ,
   com_camare.nom_campo           as nom_campo   ,
   com_camare.sec_ordena          as sec_ordena  ,
   com_camare.tip_dato            as tip_dato    ,
   com_camare.lon_presen          as lon_presen  ,
   com_camare.for_presen          as for_presen  ,
   com_camare.lon_campo           as lon_campo   ,
   com_camare.tip_presen          as tip_presen  ,
   com_camare.ubica_x             as ubica_x     ,
   com_camare.ubica_y             as ubica_y     ,
   com_camare.alt_campo           as alt_campo   ,
   com_camare.num_ordena          as num_ordena  ,
   com_camare.tip_ordena          as tip_ordena  ,
   com_camare.ind_actual          as ind_actual  ,
   com_camare.ind_modifi          as ind_modifi  ,
   com_camare.ind_consul          as ind_consul  ,
   com_camare.ind_oculto          as ind_oculto  ,
   com_camare.ind_obliga          as ind_obliga  ,
   com_camare.ind_llave           as ind_llave   ,
   com_camare.ind_atr_es          as ind_atr_es  ,
   com_camare.sec_llave           as sec_llave   ,
   com_infcam.cod_idioma          as cod_idioma  ,
   com_infcam.des_campo           as des_campo
from  com_camare, com_infcam
where  com_camare.nom_ventan   =   com_infcam.nom_ventan
and    com_camare.nom_are_da   =   com_infcam.nom_are_da
and    com_camare.nom_campo    =   com_infcam.nom_campo
;
```

COM_IDIETQ

```
create view com_idietq
as select
    com_etqare.nom_ventan          as nom_ventan   ,
    com_etqare.nom_are_da          as nom_are_da   ,
    com_etqare.num_etique          as num_etique   ,
    com_etqare.alt_etique          as alt_etique   ,
    com_etqare.lon_etique          as lon_etique   ,
    com_etqare.ubica_x             as ubica_x      ,
    com_etqare.ubica_y             as ubica_y      ,
    com_infetq.cod_idioma          as cod_idioma   ,
    com_infetq.tit_etique          as tit_etique
from  com_etqare, com_infetq
where  com_etqare.nom_ventan   =   com_infetq.nom_ventan
and    com_etqare.nom_are_da   =   com_infetq.nom_are_da
and    com_etqare.num_etique   =   com_infetq.num_etique
;
```

Security and Authentication

COM_COMPAN

```
create table com_compan (
        compania                varchar(6)          not null,
        des_compan              varchar(100)        ,
        constraint com_compan_llave primary key (compania)
);
alter table com_compan owner to dec;
grant all on table com_compan to dec;
grant all on table com_compan to public;
```

COM_USUARI

```
create table com_usuari (
        nom_usuari              varchar(8)          not null,
        des_compan              varchar(100)        ,
        constraint com_usuari_llave primary key (nom_usuari)
);
alter table com_usuari owner to dec;
grant all on table com_usuari to dec;
grant all on table com_usuari to public;
```

COM_PERMIS

```
create table com_permis (
        compania                varchar(6)          not null,
        abr_format              varchar(15)         not null,
        nom_autori              varchar(8)          not null,
        permisos                varchar(6)          not null,
        constraint com_permis_llave primary key (compania, abr_format, nom_autori),
        constraint com_permis_fllave1 foreign key (compania) references com_compan,
        constraint com_permis_fllave2 foreign key (abr_format) references com_ventan,
        constraint com_permis_fllave3 foreign key (nom_autori) references com_usuari
);
alter table com_permis owner to dec;
grant all on table com_permis to dec;
grant all on table com_permis to public;
```

COM_USRDEP

```
create table com_usrdep (
        compania                varchar(6)          not null,
        usr_padre               varchar(8)          not null,
        usr_hijo                varchar(8)          not null,
        nom_usuar               varchar(8)          ,
        constraint com_usrdep_llave primary key (compania, usr_padre, usr_hijo),
        constraint com_usrdep_fllave1 foreign key (compania) references com_compan,
        constraint com_usrdep_fllave2 foreign key (usr_padre) references com_usuari,
        constraint com_usrdep_fllave3 foreign key (usr_hijo) references com_usuari
);
alter table com_usrdep owner to dec;
grant all on table com_usrdep to dec;
grant all on table com_usrdep to public;
```

COM_PERUSR

```
create view com_perusr
as select
        com_permis.compania         as      compania,
        com_permis.abr_format       as      abr_format,
        com_permis.nom_autori       as      nom_autori,
        com_permis.permisos         as      permisos,
        com_usrdep.usr_padre        as      usr_padre,
        com_usrdep.usr_hijo         as      usr_hijo
from  com_permis,  com_usrdep
where  com_permis.compania = com_usrdep.compania
and    com_permis.permisos not in ('X')
and    ((com_permis.nom_autori = com_usrdep.usr_hijo
and      com_usrdep.usr_padre  = com_usrdep.usr_hijo)
or     (com_permis.nom_autori = com_usrdep.usr_padre
and      com_usrdep.usr_hijo not in (com_usrdep.usr_padre)
and not  com_permis.abr_format = any (select abr_format
                                from   com_permis
                                where com_permis.nom_autori = com_usrdep.usr_hijo
                               )
));
```

Window Navigation Menus

COM_NAVEGA

```
create table com_navega (
      llamante              varchar(10)       not null,
      llamado               varchar(10)       not null,
      nom_campo             varchar(10)       not null,
      secuencia             numeric(7)        not null,
      des_llamad            varchar(240)      ,
      tip_llamad            varchar(1)        ,
      constraint com_navega_llave primary key (llamante, llamado)
);
alter table com_navega owner to dec;
grant all on table com_navega to dec;
grant all on table com_navega to public;
```

COM_VENUSR

```
create table com_venusr (
      compania              varchar(6)        not null,
      nom_usuar             varchar(8)        not null,
      llamado               varchar(15)       not null,
      llamante              varchar(15)       not null,
      constraint com_venusr_llave primary key (compania, nom_usuar, llamado, llamante),
      constraint com_venusr_fllave1 foreign key (compania) references com_compan,
      constraint com_venusr_fllave2 foreign key (nom_usuar) references com_usuari,
      constraint com_venusr_fllave3 foreign key (llamante, llamado) references com_navega
);
alter table com_venusr owner to dec;
grant all on table com_venusr to dec;
grant all on table com_venusr to public;
```

COM_INFNAV

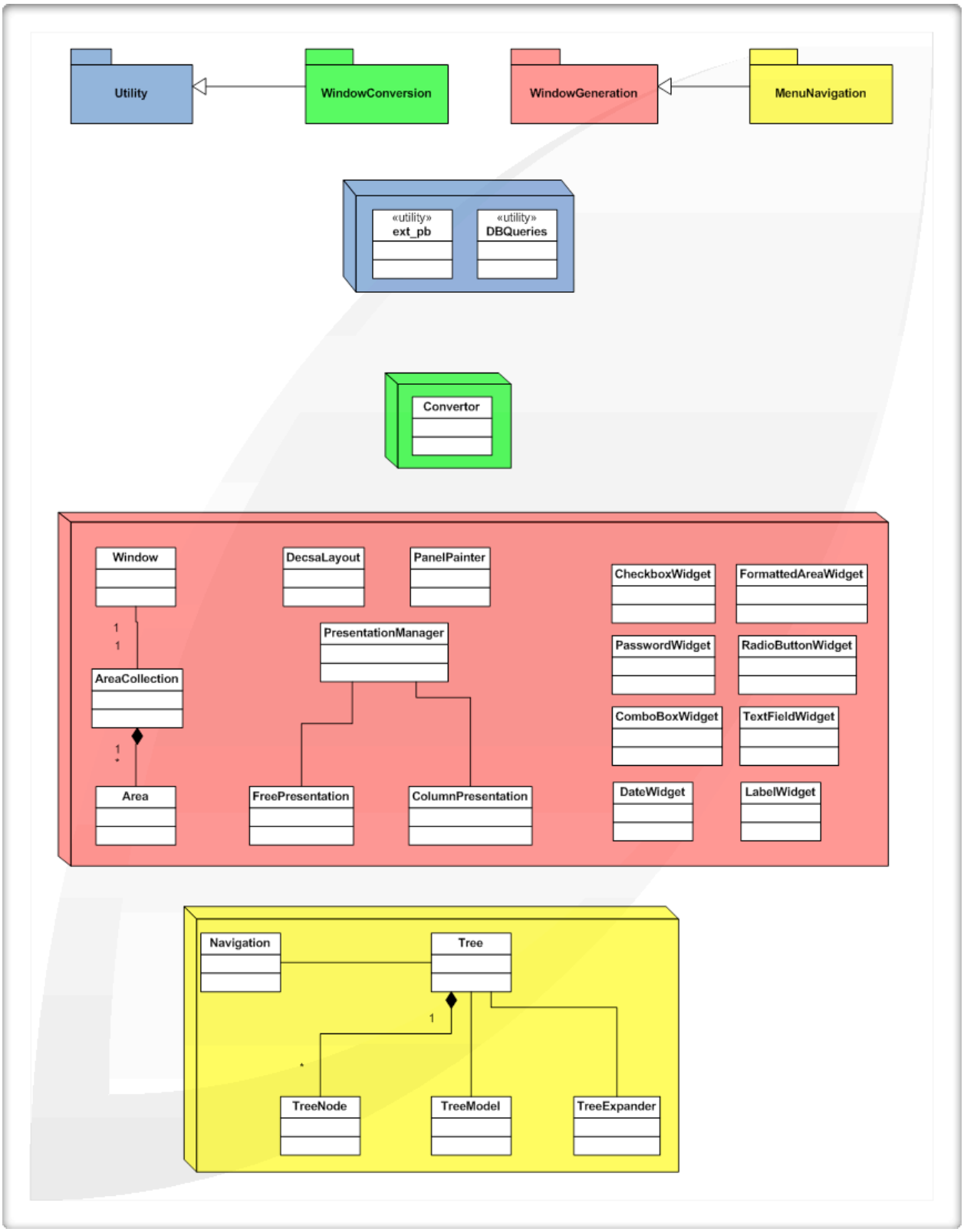
```
create table com_infnav (
        llamante                varchar(10)         not null,
        llamado                 varchar(10)         not null,
        cod_idioma              varchar(6)          not null,
        titulo                  varchar(40)         not null,
        constraint com_infnav_llave primary key (llamante, llamado, cod_idioma),
        constraint com_infnav_fllave1 foreign key (llamante, llamado) references com_navega,
        constraint com_infnav_fllave2 foreign key (cod_idioma) references com_idioma
);
alter table com_infnav owner to dec;
grant all on table com_infnav to dec;
grant all on table com_infnav to public;
```

COM_IDINAV

```
create view com_idinav
as select
    com_navega.llamante             as llamante     ,
    com_navega.llamado              as llamado      ,
    com_navega.nom_campo            as nom_campo    ,
    com_navega.secuencia            as secuencia    ,
    com_navega.des_llamad           as des_llamad   ,
    com_navega.tip_llamad           as tip_llamad   ,
    com_infnav.cod_idioma           as cod_idioma   ,
    com_infnav.titulo               as titulo
from  com_navega, com_infnav
where  com_navega.llamante  =  com_infnav.llamante
and    com_navega.llamado   =  com_infnav.llamado
;
```

COM_PENAVE

```
create view com_penave
as select
        distinct com_perusr.compania        as      compania,
        com_perusr.usr_hijo                 as      nom_autori      ,
        com_perusr.permisos                 as      permisos        ,
        com_idinav.llamante                 as      llamante        ,
        com_idinav.llamado                  as      llamado         ,
        com_idinav.titulo                   as      titulo          ,
        com_idinav.nom_campo                as      nom_campo       ,
        com_idinav.secuencia                as      secuencia       ,
        com_idinav.des_llamad               as      des_llamad      ,
        com_idinav.tip_llamad               as      tip_llamad      ,
        com_idinav.cod_idioma               as      cod_idioma
from  com_perusr, com_idinav
where  com_perusr.abr_format = com_idinav.llamado
and    tip_llamad in ('N', 'V', 'H', 'A', 'S')
; -- (+)
```


Utility
WindowConversion
WindowGeneration
MenuNavigation
«utility»
ext_pb
«utility»
DBQueries
Convertor
Window
DecsaLayout
PanelPainter
CheckboxWidget
FormattedAreaWidget
1
1
PresentationManager
PasswordWidget
RadioButtonWidget
AreaCollection
ComboBoxWidget
TextFieldWidget
1
*
Area
FreePresentation
ColumnPresentation
DateWidget
LabelWidget
Navigation
Tree
1
*
TreeNode
TreeModel
TreeExpander

System Design

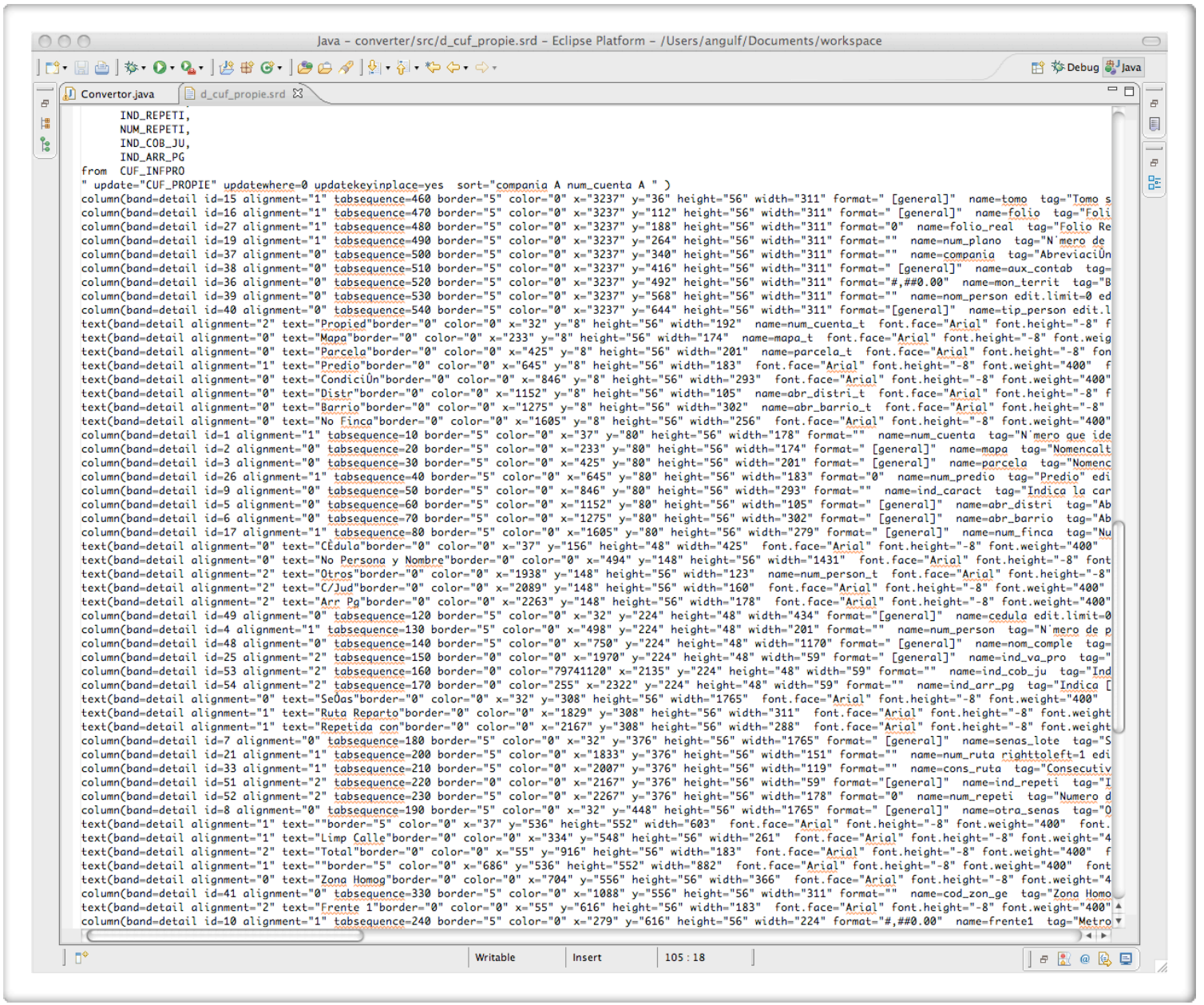
Window Converter

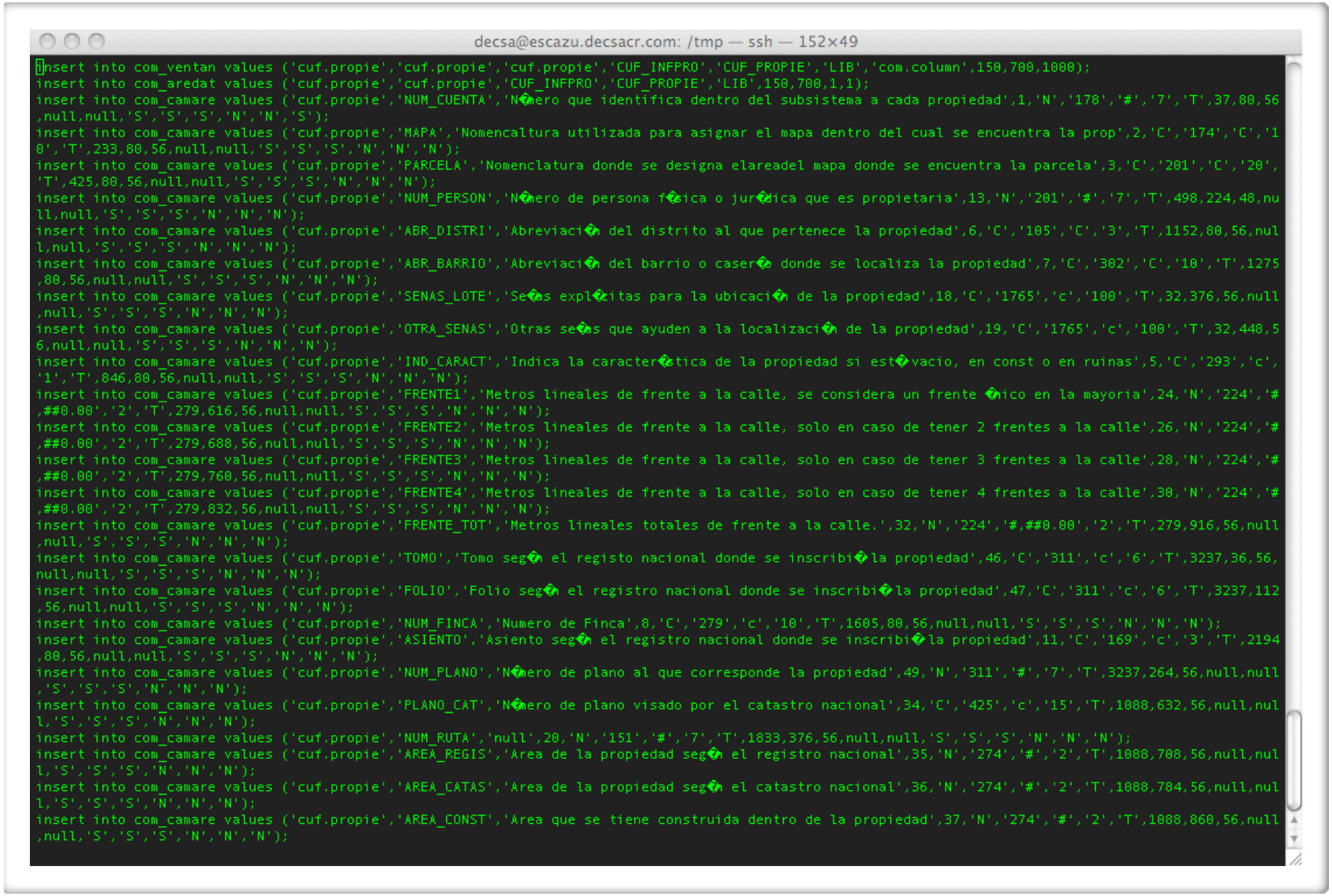
After finishing up the database design, the next step was to convert the PowerBuilder window definitions into database rows that could be inserted into the new database design. I did not know enough about PowerBuilder in order to accomplish this part of the project on my own. For this reason, I got help from several employees of the company. Due to the time limitations, I just presented the new database design to the PowerBuilder experts and they determined what could be extracted from the PowerBuilder window definitions. I attach a screenshot of what a portion of the window definition looks like.

A utility program was created called ext_pb, or extract from PowerBuilder. This utility program was created by the PowerBuilder experts and it received the path for a window definition file. It would then go through the file and output four different files. The first file contains the mappings of the separate fields to the database. That is, what information from the database was displayed in that field. The second file shows the select statement executed to get the data from the database in order to populate the window fields with data. The third file describes each of the window labels and specifies formatting such as height, width, border, title and x and y coordinates. The fourth file describes each of the window fields and also specifies formatting.

Out of the four files, we only needed the third and fourth files to import data into the new window definition tables that were created. The other two files deal with issues that are out of the scope of this project. When I got these files, I created a convertor utility program that would read these label and field descriptions and created SQL statements for each of the windows. This is a very straight-forward, but lengthy (approximately 1000 lines), file manipulation Java program. The program converts other information that is not covered in this project. For example, it converts information for presentation values. If the database returns an 'S', the window field will be present true. If the database returns an 'N', the window field will present false. This information is needed for the system but not for this project. In total, the convertor generates three files. We only are interested in the first one which creates the SQL statements for inserting into COM_VENTAN, COM_AREDAT, COM_CAMARE and COM_ETQARE. A screenshot of this file is attached.

It must be noted that this convertor is not perfect. Even after conversion, the window fields and labels have to be tweaked for the window to look correct. But with the new database design, assigning formatting to the labels and fields is particularly easy. Unfortunately, all the formatting and the issues with field and label positioning in the window have to be solved by hand. That is, actual database tuples have to be modified. One of the future enhancements of this project will be to create a tool for modifying windows easily. The idea is that a developer can move fields around in a drag and drop manner until he is satisfied with the look and layout of the window. The same goes for formatting. The developer should be able to select a label and apply a font, size and other formatting in an easy way. Unfortunately, there is no time to finish this before December.


Java - converter/src/d_cuf_propie.srd - Eclipse Platform - /Users/angulf/Documents/workspace
Debug
Java
Convertor.java
d_cuf_propie.srd
IND_REPETI,
NUM_REPETI,
IND_COB_JU,
IND_ARR_PG
from CUF_INFPRO
" update="CUF_PROPIE" updatewhere=0 updatekeyinplace=yes sort="compania A num_cuenta A ")
column(band=detail id=15 alignment="1" tabsequence=460 border="5" color="0" x="3237" y="36" height="56" width="311" format=" [general]" name=tomo tag="Tomo s
column(band=detail id=16 alignment="1" tabsequence=470 border="5" color="0" x="3237" y="112" height="56" width="311" format=" [general]" name=folio tag="Foli
column(band=detail id=27 alignment="1" tabsequence=480 border="5" color="0" x="3237" y="188" height="56" width="311" format="0" name=folio_real tag="Folio Re
column(band=detail id=19 alignment="1" tabsequence=490 border="5" color="0" x="3237" y="264" height="56" width="311" format="" name=num_plano tag="N'mero de
column(band=detail id=37 alignment="0" tabsequence=500 border="5" color="0" x="3237" y="340" height="56" width="311" format="" name=compania tag="AbreviaciÛn
column(band=detail id=38 alignment="0" tabsequence=510 border="5" color="0" x="3237" y="416" height="56" width="311" format=" [general]" name=aux_contab tag=
column(band=detail id=36 alignment="0" tabsequence=520 border="5" color="0" x="3237" y="492" height="56" width="311" format="#,##0.00" name=mon_territ tag="B
column(band=detail id=39 alignment="0" tabsequence=530 border="5" color="0" x="3237" y="568" height="56" width="311" format="" name=nom_person edit.limit=0 ed
column(band=detail id=40 alignment="0" tabsequence=540 border="5" color="0" x="3237" y="644" height="56" width="311" format="[general]" name=tip_person edit.l
text(band=detail alignment="2" text="Propied"border="0" color="0" x="32" y="8" height="56" width="192" name=num_cuenta_t font.face="Arial" font.height="-8" f
text(band=detail alignment="0" text="Mapa"border="0" color="0" x="233" y="8" height="56" width="174" name=mapa_t font.face="Arial" font.height="-8" font.weig
text(band=detail alignment="0" text="Parcela"border="0" color="0" x="425" y="8" height="56" width="201" name=parcela_t font.face="Arial" font.height="-8" fon
text(band=detail alignment="1" text="Predio"border="0" color="0" x="645" y="8" height="56" width="183" font.face="Arial" font.height="-8" font.weight="400" f
text(band=detail alignment="0" text="CondiciÛn"border="0" color="0" x="846" y="8" height="56" width="293" font.face="Arial" font.height="-8" font.weight="400"
text(band=detail alignment="0" text="Distr"border="0" color="0" x="1152" y="8" height="56" width="105" name=abr_distri_t font.face="Arial" font.height="-8" f
text(band=detail alignment="0" text="Barrio"border="0" color="0" x="1275" y="8" height="56" width="302" name=abr_barrio_t font.face="Arial" font.height="-8"
text(band=detail alignment="0" text="No Finca"border="0" color="0" x="1605" y="8" height="56" width="256" font.face="Arial" font.height="-8" font.weight="400"
column(band=detail id=1 alignment="1" tabsequence=10 border="5" color="0" x="37" y="80" height="56" width="178" format="" name=num_cuenta tag="N'mero que ide
column(band=detail id=2 alignment="0" tabsequence=20 border="5" color="0" x="233" y="80" height="56" width="174" format=" [general]" name=mapa tag="Nomencalt
column(band=detail id=3 alignment="0" tabsequence=30 border="5" color="0" x="425" y="80" height="56" width="201" format=" [general]" name=parcela tag="Nomenc
column(band=detail id=26 alignment="1" tabsequence=40 border="5" color="0" x="645" y="80" height="56" width="183" format="0" name=num_predio tag="Predio" edi
column(band=detail id=9 alignment="0" tabsequence=50 border="5" color="0" x="846" y="80" height="56" width="293" format="" name=ind_caract tag="Indica la car
column(band=detail id=5 alignment="0" tabsequence=60 border="5" color="0" x="1152" y="80" height="56" width="105" format=" [general]" name=abr_distri tag="Ab
column(band=detail id=6 alignment="0" tabsequence=70 border="5" color="0" x="1275" y="80" height="56" width="302" format=" [general]" name=abr_barrio tag="Ab
column(band=detail id=17 alignment="1" tabsequence=80 border="5" color="0" x="1605" y="80" height="56" width="279" format=" [general]" name=num_finca tag="Nu
text(band=detail alignment="0" text="CÉdula"border="0" color="0" x="37" y="156" height="48" width="425" font.face="Arial" font.height="-8" font.weight="400"
text(band=detail alignment="0" text="No Persona y Nombre"border="0" color="0" x="494" y="148" height="56" width="1431" font.face="Arial" font.height="-8" font
text(band=detail alignment="2" text="Otros"border="0" color="0" x="1938" y="148" height="56" width="123" name=num_person_t font.face="Arial" font.height="-8"
text(band=detail alignment="2" text="C/Jud"border="0" color="0" x="2089" y="148" height="56" width="160" font.face="Arial" font.height="-8" font.weight="400"
text(band=detail alignment="2" text="Arr Pg"border="0" color="0" x="2263" y="148" height="56" width="178" font.face="Arial" font.height="-8" font.weight="400"
column(band=detail id=49 alignment="0" tabsequence=120 border="5" color="0" x="32" y="224" height="48" width="434" format="[general]" name=cedula edit.limit=0
column(band=detail id=4 alignment="1" tabsequence=130 border="5" color="0" x="498" y="224" height="48" width="201" format="" name=num_person tag="N'mero de p
column(band=detail id=48 alignment="0" tabsequence=140 border="5" color="0" x="750" y="224" height="48" width="1170" format=" [general]" name=nom_comple tag=
column(band=detail id=25 alignment="2" tabsequence=150 border="0" color="0" x="1970" y="224" height="48" width="59" format=" [general]" name=ind_va_pro tag="
column(band=detail id=53 alignment="2" tabsequence=160 border="0" color="79741120" x="2135" y="224" height="48" width="59" format="" name=ind_cob_ju tag="Ind
column(band=detail id=54 alignment="2" tabsequence=170 border="0" color="255" x="2322" y="224" height="48" width="59" format="" name=ind_arr_pg tag="Indica [
text(band=detail alignment="0" text="SeÒas"border="0" color="0" x="32" y="308" height="56" width="1765" font.face="Arial" font.height="-8" font.weight="400"
text(band=detail alignment="1" text="Ruta Reparto"border="0" color="0" x="1829" y="308" height="56" width="311" font.face="Arial" font.height="-8" font.weight
text(band=detail alignment="1" text="Repetida con"border="0" color="0" x="2167" y="308" height="56" width="288" font.face="Arial" font.height="-8" font.weight
column(band=detail id=7 alignment="0" tabsequence=180 border="5" color="0" x="32" y="376" height="56" width="1765" format=" [general]" name=senas_lote tag="S
column(band=detail id=21 alignment="1" tabsequence=200 border="5" color="0" x="1833" y="376" height="56" width="151" format="" name=num_ruta righttoleft=1 edi
column(band=detail id=33 alignment="1" tabsequence=210 border="5" color="0" x="2007" y="376" height="56" width="119" format="" name=cons_ruta tag="Consecutiv
column(band=detail id=51 alignment="2" tabsequence=220 border="0" color="0" x="2167" y="376" height="56" width="59" format="[general]" name=ind_repeti tag="I
column(band=detail id=52 alignment="2" tabsequence=230 border="5" color="0" x="2267" y="376" height="56" width="178" format="0" name=num_repeti tag="Numero d
column(band=detail id=8 alignment="0" tabsequence=190 border="5" color="0" x="32" y="448" height="56" width="1765" format=" [general]" name=otra_senas tag="O
text(band=detail alignment="1" text=""border="5" color="0" x="37" y="536" height="552" width="603" font.face="Arial" font.height="-8" font.weight="400" font.
text(band=detail alignment="1" text="Limp Calle"border="0" color="0" x="334" y="548" height="56" width="261" font.face="Arial" font.height="-8" font.weight="4
text(band=detail alignment="2" text="Total"border="0" color="0" x="55" y="916" height="56" width="183" font.face="Arial" font.height="-8" font.weight="400" f
text(band=detail alignment="1" text=""border="5" color="0" x="686" y="536" height="552" width="882" font.face="Arial" font.height="-8" font.weight="400" font
text(band=detail alignment="0" text="Zona Homog"border="0" color="0" x="704" y="556" height="56" width="366" font.face="Arial" font.height="-8" font.weight="4
column(band=detail id=41 alignment="0" tabsequence=330 border="5" color="0" x="1088" y="556" height="56" width="311" format="" name=cod_zon_ge tag="Zona Homo
text(band=detail alignment="2" text="Frente 1"border="0" color="0" x="55" y="616" height="56" width="183" font.face="Arial" font.height="-8" font.weight="400"
column(band=detail id=10 alignment="1" tabsequence=240 border="5" color="0" x="279" y="616" height="56" width="224" format="#,##0.00" name=frente1 tag="Metro
Writable
Insert
105 : 18

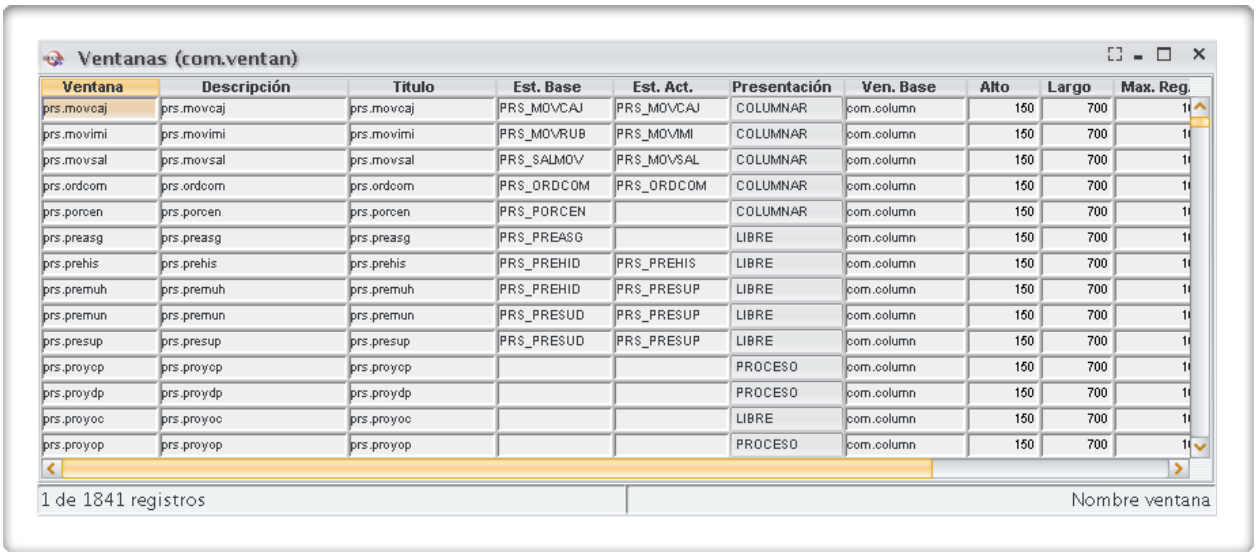

decsa@escazu.decsacr.com: /tmp — ssh — 152×49
insert into com_ventan values ('cuf.propie','cuf.propie','cuf.propie','CUF_INFPRO','CUF_PROPIE','LIB','com.column',150,700,1000);
insert into com_aredat values ('cuf.propie','cuf.propie','CUF_INFPRO','CUF_PROPIE','LIB',150,700,1,1);
insert into com_camare values ('cuf.propie','NUM_CUENTA','N□mero que identifica dentro del subsistema a cada propiedad',1,'N','178','#','7','T',37,80,56,null,null,'S','S','S','N','N','S');
insert into com_camare values ('cuf.propie','MAPA','Nomencaltura utilizada para asignar el mapa dentro del cual se encuentra la prop',2,'C','174','C','10','T',233,80,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','PARCELA','Nomenclatura donde se designa elareadel mapa donde se encuentra la parcela',3,'C','201','C','20','T',425,80,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','NUM_PERSON','N□mero de persona f□sica o jur□dica que es propietaria',13,'N','201','#','7','T',498,224,48,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','ABR_DISTRI','Abreviaci□ del distrito al que pertenece la propiedad',6,'C','105','C','3','T',1152,80,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','ABR_BARRIO','Abreviaci□ del barrio o caser□ donde se localiza la propiedad',7,'C','302','C','10','T',1275,80,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','SENAS_LOTE','Se□as expl□citas para la ubicaci□ de la propiedad',18,'C','1765','c','100','T',32,376,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','OTRA_SENAS','Otras se□as que ayuden a la localizaci□ de la propiedad',19,'C','1765','c','100','T',32,448,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','IND_CARACT','Indica la caracter□stica de la propiedad si est□vacio, en const o en ruinas',5,'C','293','c','1','T',846,80,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','FRENTE1','Metros lineales de frente a la calle, se considera un frente □nico en la mayoria',24,'N','224','#,##0.00','2','T',279,616,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','FRENTE2','Metros lineales de frente a la calle, solo en caso de tener 2 frentes a la calle',26,'N','224','#,##0.00','2','T',279,688,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','FRENTE3','Metros lineales de frente a la calle, solo en caso de tener 3 frentes a la calle',28,'N','224','#,##0.00','2','T',279,760,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','FRENTE4','Metros lineales de frente a la calle, solo en caso de tener 4 frentes a la calle',30,'N','224','#,##0.00','2','T',279,832,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','FRENTE_TOT','Metros lineales totales de frente a la calle.',32,'N','224','#,##0.00','2','T',279,916,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','TOMO','Tomo seg□ el registo nacional donde se inscribi□la propiedad',46,'C','311','c','6','T',3237,36,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','FOLIO','Folio seg□ el registro nacional donde se inscribi□la propiedad',47,'C','311','c','6','T',3237,112,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','NUM_FINCA','Numero de Finca',8,'C','279','c','10','T',1605,80,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','ASIENTO','Asiento seg□ el registro nacional donde se inscribi□la propiedad',11,'C','169','c','3','T',2194,80,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','NUM_PLANO','N□mero de plano al que corresponde la propiedad',49,'N','311','#','7','T',3237,264,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','PLANO_CAT','N□mero de plano visado por el catastro nacional',34,'C','425','c','15','T',1088,632,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','NUM_RUTA','null',20,'N','151','#','7','T',1833,376,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','AREA_REGIS','Area de la propiedad seg□ el registro nacional',35,'N','274','#','2','T',1088,708,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','AREA_CATAS','Area de la propiedad seg□ el catastro nacional',36,'N','274','#','2','T',1088,784,56,null,null,'S','S','S','N','N','N');
insert into com_camare values ('cuf.propie','AREA_CONST','Area que se tiene construida dentro de la propiedad',37,'N','274','#','2','T',1088,860,56,null,null,'S','S','S','N','N','N');

Window Generator

The next big step was to create a window generator that would query the new database tables at runtime and display a Java Swing frame in the system with all the specified fields and labels. First things first. As many people know, layouts in Java Swing are not very pleasant to work with. In this case, our window definitions stored field and label positions in pixels. No layout that I knew of could solve our positioning problem in pixels easily. For this reason, a completely new layout was created called DecsaLayout. It is very simple in reality. The developer passes the object that needs to be displayed along with a vector that contains four entries: width, height, x-coordinate and y-coordinate. It is just as easy as it sounds. I was very surprised at how simple it was to create a layout in Java from scratch.

With this new layout, things became a little less difficult. The generator consisted of a class called Window which represented a window in the system. This would be the interface that would be called once we got the menu navigation working. Once the user selected a window in the menu system, this class would be invoked. The Window class would first of all call the AreaCollection class to create the different areas that make up the window. This class queries the database to find the information in COM_AREDAT for each of the areas. For each of the areas defined in the database table, AreaCollection creates an Area class instance which represents a specific area of a window.

Within the Area instance we make a call to the PresentationManager class. This class manages the presentation of the fields and labels for each area. In the system there are two types of windows. There are windows with columnar presentation and others with free presentation.

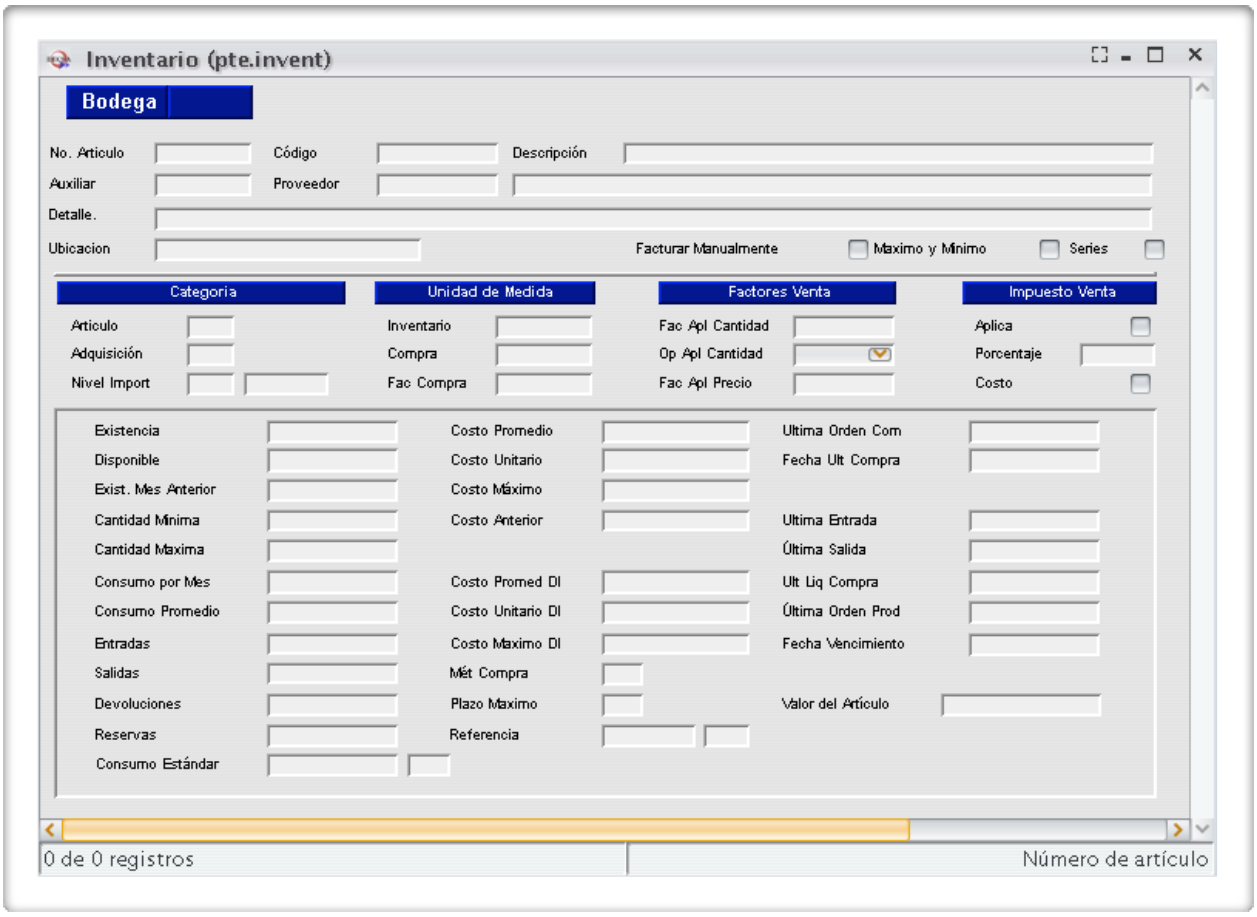


Columnar Presentation Window

PresentationManager determines which type of presentation is needed by querying the database and calls either FreePresentation or ColumnarPresentation. FreePresentation creates the free presentation and it goes through the window definition looking for each of the window fields and labels, positioning them correctly on the JPanel depending on their coordinates, and finally applying any formatting that is necessary such as bolding, italics, underlining, colors, size, font, borders, etc. ColumnPresentation is much simpler because it just needs to create a table much like a spreadsheet with columns corresponding to each of the fields. Columnar presentations do not have labels, they consist only of field descriptions. We just check each of the fields, the field type (checkbox, combobox, normal textbox), and the field width.

This concludes the description of the central piece of the window generator. I also created specific classes to handle the functionality of checkboxes, comboboxes, textboxes and all the other presentation objects that could appear in a window. These classes extend the normal Java Swing classes such as JComboBox, JTextField, JCheckbox, etc. These are especially useful when applying the formatting to each of the fields and labels. I exposed methods that would make formatting a breeze. For example, I could call the combo box method for changing the font. I would specify the desired font and this would make any necessary changes to the JComboBox object in the window. These methods will

also be useful if, later on, we implement runtime changes to the font or text size for example. The screenshots presented above and below are actual windows generated by the window generator. Clearly, the one up top does not lend itself to much formatting. It is just a simple spreadsheet with values. But the one in the following page has colors, different text sizes and a more complex layout. This is the true power of the window generator combined with the database design for window descriptions. But again, as you can imagine, in order to move one of these fields a little to the left, we need to go into a database row and modify the coordinate values. This can be a very time-consuming task, especially when you are trying to get 5,000 windows perfectly displayed.



Free Presentation Window

Menu Navigation

The last part of the project that was completed before this report was due, is the menu navigation subsystem. With the menu navigation database design in place, we decided to create a way to navigate through all the options in a straight-forward manner. The first option that came to mind was “cascading-windows”. That is, for each menu defined in the COM_NAVEGA table we would create a list in a window specifying all the options (windows or more menus). If the user selected a “runnable” window from the list, then the window generator would be invoked to display the selected window. On the other hand, if the user selected another menu within the menu, another list would appear next to the current list. In effect, we would be creating a sort of waterfall of lists. To go back one level, the user could close the lowest list in the waterfall and go back to the previous list.

This wasn't a bad idea, but then I came across the JTree component in Java. Implementing the menu with this component would have several advantages over the cascading windows options. First, there would be a single window to display all the menu navigation options because the only component on screen would be the tree. Furthermore, the user could have several menus expanded at the same time. With the cascading windows option, expanding different menus would be very confusing to the user. The tree, on the other hand, would keep the options organized and there would be no “mess” of windows all over the screen. I decided to go with the tree implementation, even though I had never used these components before.

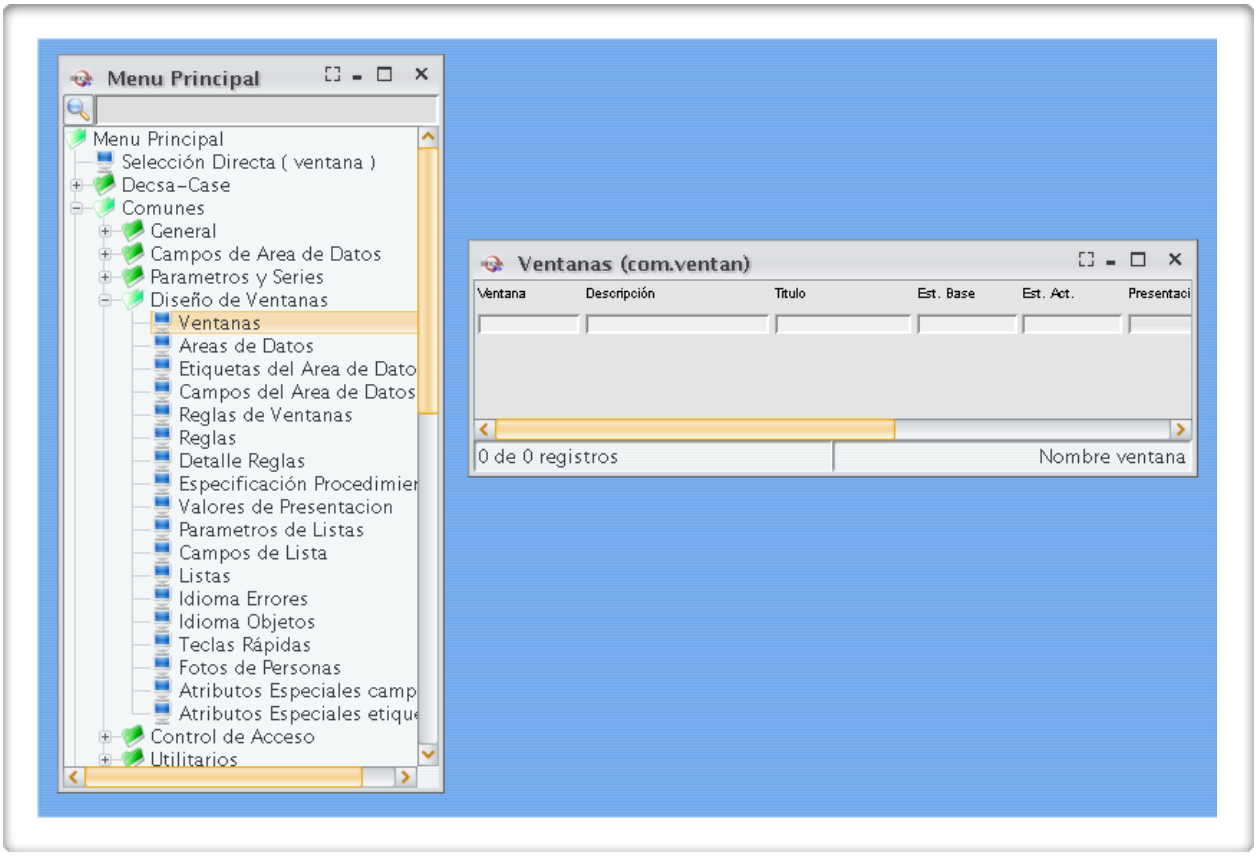
In order to create a JTree, I first needed to define a TreeModel that defined the behavior of the tree. A class called TreeModel was created and it extended the Java Swing TreeModel (same name) class. In this class, I defined methods such as counting how many children the tree had, and how to expand and collapse nodes. The class TreeNode extended DefaultMutableTreeNode and implemented the Java interface TreeNode (same name). This class represents a node, but it can be either a node or a leaf of the tree. This was all that was necessary to create a tree.

Next, I defined the Navigation and Tree classes. The Navigation class would create a frame with a Tree inside it. The Tree was initialized with an instance of the TreeModel class and a “dummy” TreeNode was inserted as the root of the tree. Now, I had to populate the tree with the values from COM_PENAVE. To start, the Tree class queries COM_PENAVE with the attribute LLAMANTE = 'glb.menupr'. This corresponds to the first level of menus of the system. The results of this query would be inserted into the tree. The query could return two types of results: actual windows or submenus. For this reason, the TreeNode class had two separate constructors. One would create a node as a leaf and the other would create the node as a submenu.

At this point, I had to make a decision about how the tree would be populated. The first option was to populate the complete tree during initialization of the system. The second was to only populate the first level of the tree. Then, when the user wanted to expand a node, a separate query would be executed that would populate the next level. I tried the first option while running the database locally. It took approximately 10 seconds to populate the entire tree for a user with access to approximately 3000 windows. Once I executed the query using a remote database, I decided to go with the second option. For the same user, the remote query took almost five minutes. It was preferable for the user to wait 1-2 seconds to expand each subsequent level of the tree, than to wait five minutes to load the entire tree at the beginning. In addition, the user might not need the entire tree to begin with.

To implement the second option, I created a class called TreeExpander. Once a user selected a node in the tree to expand, the Tree class would pass a reference to the selected node as parameter to the TreeExpander class. This class would execute the query to COM_PENAVE to retrieve the options under that node and populate it. Therefore, the Tree class would listen for expansion events and call the TreeExpander class. This concluded the implementation of the menu navigation system.

The screenshot below shows a navigation tree with several levels open and a window that was generated and displayed after the user selected the ‘Ventanas’ runnable option in the tree.



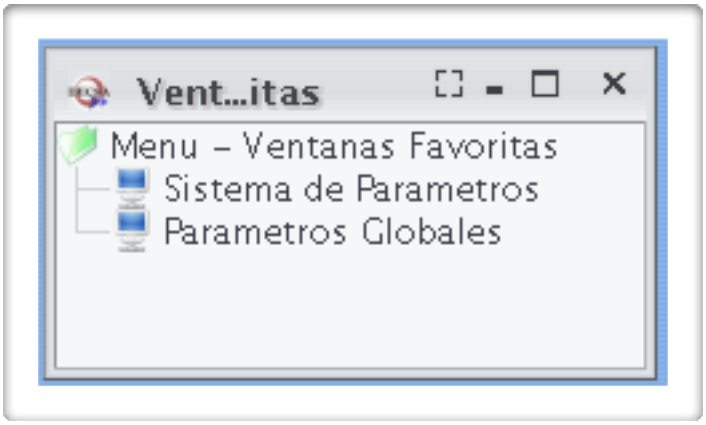
Menu Navigation and Window Generation Combined

Favorite Windows

The favorite windows implementation was very much simplified after the menu navigation was all complete. We could extend from the functionality provided to the main tree. In our favorite windows tree, we would only have windows, no submenus. Therefore, our tree would be much simpler. First, we provided a way to add favorite windows to the system. In order to do this, the user would right-click on any of the window options of the main tree navigation window. To add the selected window, the user would need to select the 'Add to Favorite Windows' option.

Add a Favorite Window

This action would trigger a new creation of a tuple in the COM_FAVWIN table. The information stored in the COM_FAVWIN table is taken from the COM_PENAVE view. Since the nodes of the tree store their unique identification, we can index into the appropriate tuple in COM_PENAVE, to retrieve the window information necessary to create the new tuple in COM_FAVWIN. After the insertion into the table, if the user brings up the favorite window tree, he will find the newly selected option. The favorite window tree only populates the data from the COM_FAVWIN table. There is no reference to the COM_PENAVE view with has all the security features for access control. It could be argued that there is a hole in the security of the system. But since the only way to add a favorite window is through the main navigation menu, then I believe the 'security hole' is non-existent. That is, also assuming that the user cannot hack into the database and inserts tuples himself. Since the favorite windows tree inherits the functionality from the main navigation tree, it will have the same way of operation. If the user selects a node, the window generator will be invoked to present the window on the desktop.

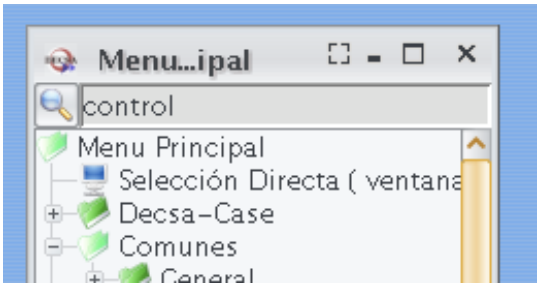


Favorite Windows Tree

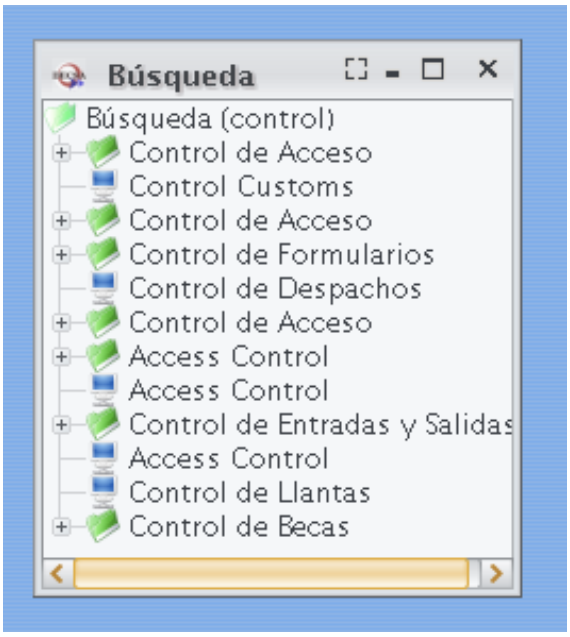
Window Search

Above the main menu navigation tree, there is a text box for searching for a window. This is used for rapid access to windows. Instead of traversing the tree, the user can just search for a window by title or codename. When the user wants to execute a search, the COM_PENAVE view is queried for titles and codenames of windows. The windows returned are presented in another tree similar to the favorite window tree. The only difference is that a search can return submenus, unlike the favorite window tree.

The favorite windows implementation was very much simplified after the menu navigation was all complete. We could extend from the functionality provided to the main tree. In our favorite windows tree, we would only have windows, no submenus. Therefore, our tree would be much simpler. First, we provided a way to add favorite windows to the system. In order to do this, the user would right-click on any of the window options of the main tree navigation window. To add the selected window, the user would need to select the 'Add to Favorite Windows' option.



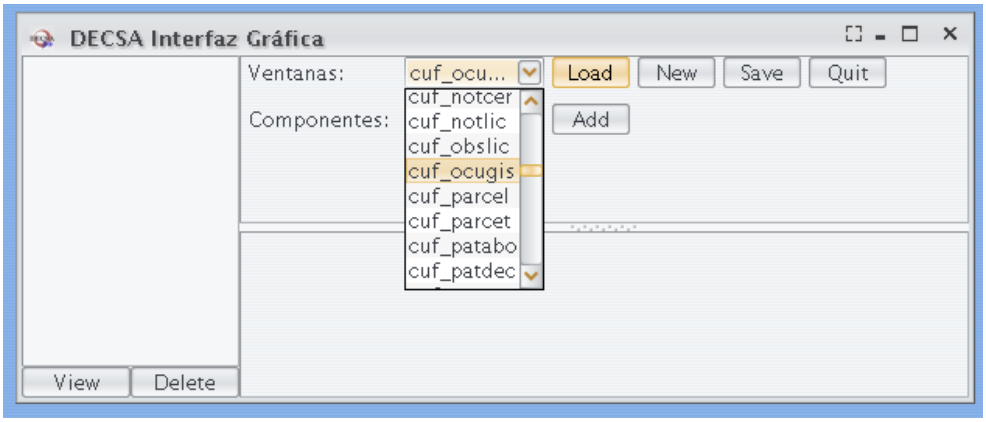
Search for 'control'



Search Results for 'control'

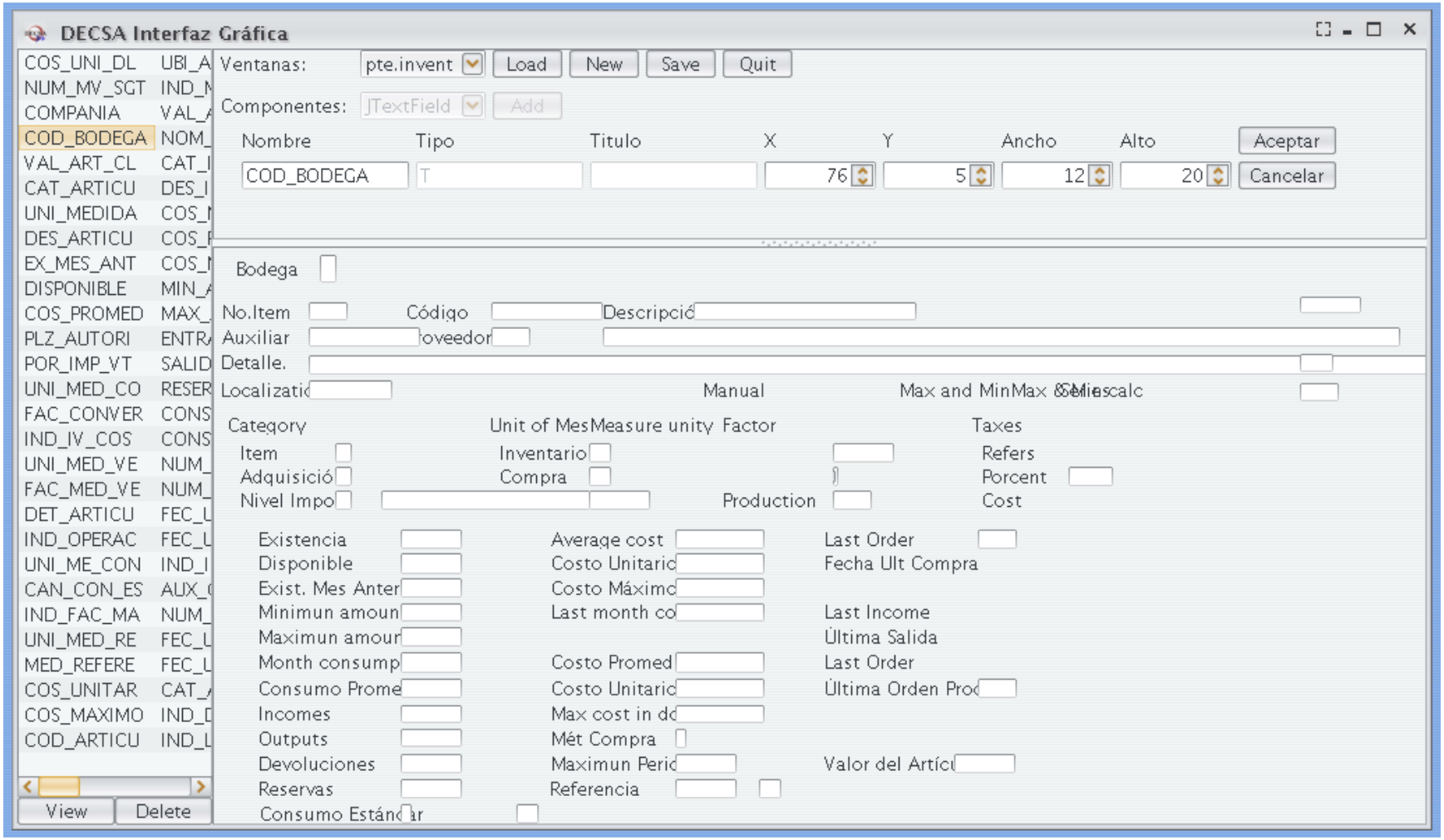
Window Designer (Prototype)

The window generator feature is very convenient, but not for when a window needs to be modified. In order to modify the layout, the user would manually have to change the tuples in the table. To make life easier on the developers, I started to develop a prototype for a window designer. Ultimately, this designer would have drag-and-drop functionality and formatting capabilities. But due to the limited time remaining, I kept this as a prototype.



Load the 'cuf_ocugis' window

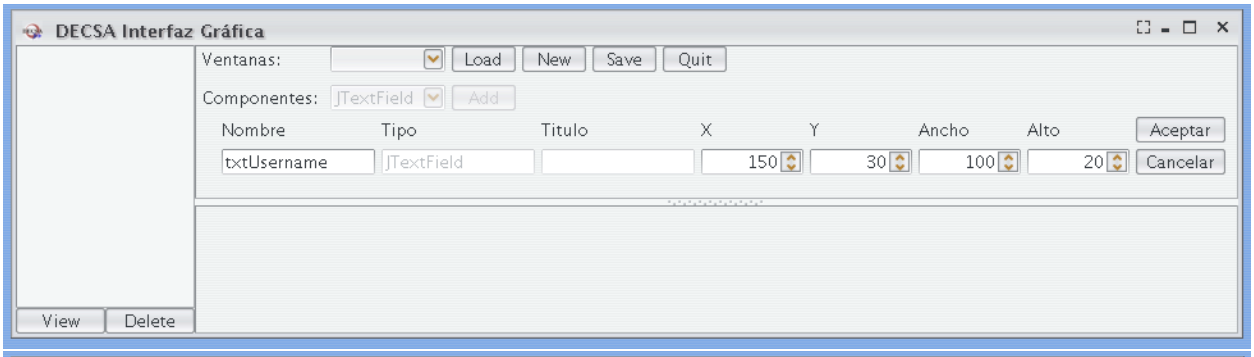
The developer can either select an existing window and load it, or create a new window from scratch. In the figure above, the window 'cuf_ocugis' is loaded into the window designer. The window fields and labels will appear in the list on the left. Then the developer can select a field or label to display its properties. Currently, the window designer will display the name, title, component type, x-coordinate, y-coordinate, width and height. Then, the user can modify any of these values. The developer can also delete or add new fields if necessary. In the screenshot on the next page, the developer is modifying the 'COD_BODEGA' field. Notice that in the bottom panel, the actual window is presented, just like it would look using the window generator. Once the developers modifies a value, the corresponding changes are displayed in the bottom panel preview. Clearly, this specific window needs some modifications in order to be ready for use.

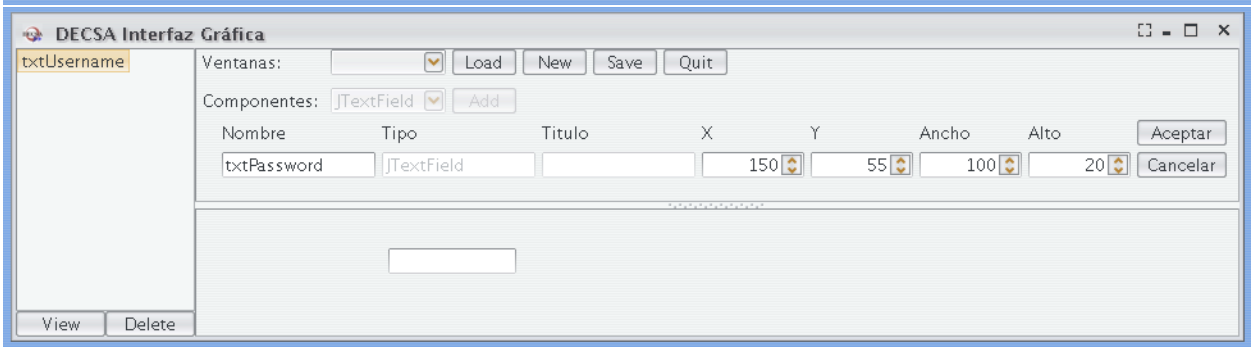

DECSA Interfaz Gráfica
COS_UNI_DL
NUM_MV_SGT
COMPANIA
COD_BODEGA
VAL_ART_CL
CAT_ARTICU
UNI_MEDIDA
DES_ARTICU
EX_MES_ANT
DISPONIBLE
COS_PROMED
PLZ_AUTORI
POR_IMP_VT
UNI_MED_CO
FAC_CONVER
IND_IV_COS
UNI_MED_VE
FAC_MED_VE
DET_ARTICU
IND_OPERAC
UNI_ME_CON
CAN_CON_ES
IND_FAC_MA
UNI_MED_RE
MED_REFERE
COS_UNITAR
COS_MAXIMO
COD_ARTICU
View
Delete
Ventanas:
pte.invent
Load
New
Save
Quit
Componentes:
JTextField
Add
Nombre
Tipo
Titulo
X
Y
Ancho
Alto
Aceptar
COD_BODEGA
T
76
5
12
20
Cancelar
Bodega
No.Item
Código
Descripció
Auxiliar
oveedor
Detalle.
Localizati
Manual
Max and Min
Category
Unit of Mes
Measure unity
Factor
Taxes
Item
Inventario
Refers
Adquisició
Compra
Porcent
Nivel Impo
Production
Cost
Existencia
Disponible
Exist. Mes Anter
Minimun amoun
Maximun amour
Month consump
Consumo Prome
Incomes
Outputs
Devoluciones
Reservas
Consumo Estánc
Average cost
Costo Unitaric
Costo Máximc
Last month co
Costo Promed
Costo Unitaric
Max cost in dc
Mét Compra
Maximun Perid
Referencia
Last Order
Fecha Ult Compra
Last Income
Última Salida
Last Order
Última Orden Proc
Valor del Artícu

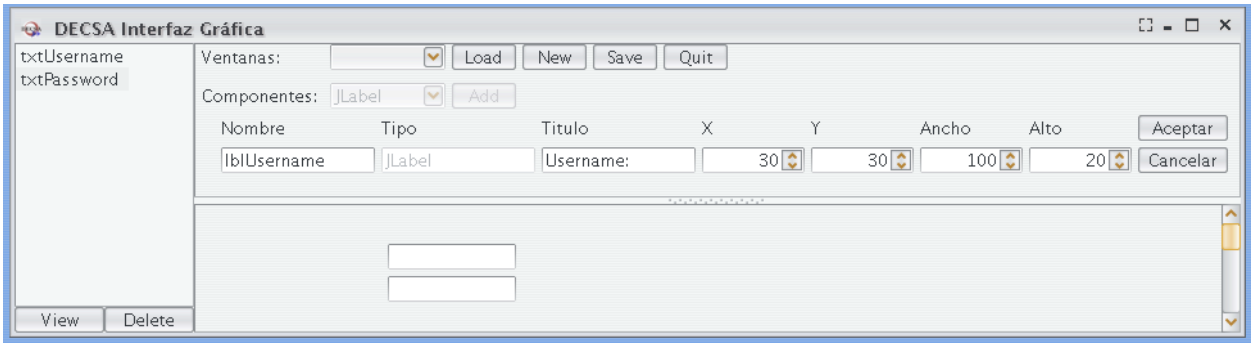
Evaluations

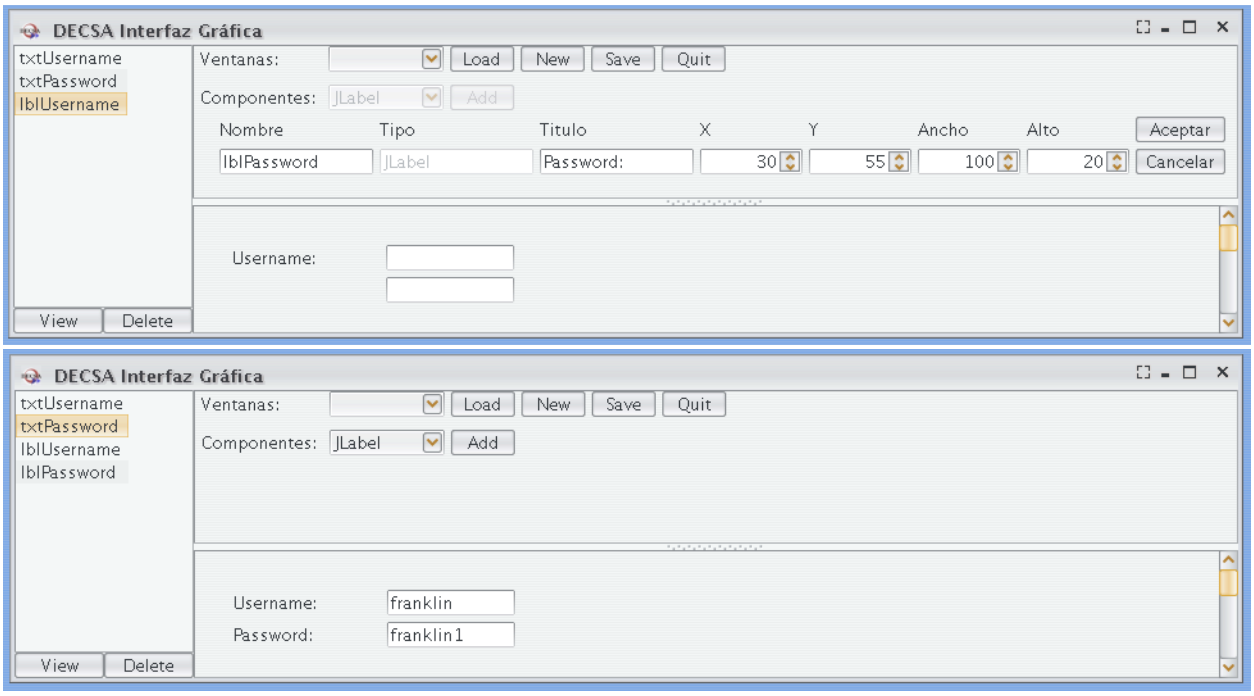
Window Generator Evaluation

The Window Designer prototype not only served as a window modification tool, but also as an evaluator for the window generator. Since the generator is embedded into the window designer, I could test it by inserting new fields and noticing the changes in the visual window output. In the following screenshots, I go through the steps of adding several components and labels and the changes in the generated window can be seen.



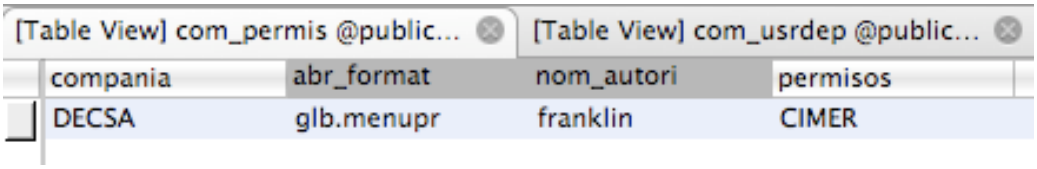




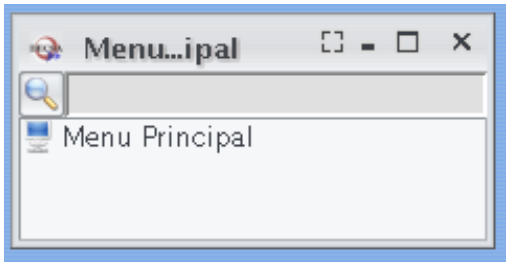


Access Rights Evaluation

The requirements specify that we need to verify that the access rights for a user are enforced in the menu navigation tree. I will start by assigning access to the user named 'franklin' in the COM_PERMIS table. This user will only have access to the 'glb.menupr' menu which is the top level object of the tree. Therefore, the menu navigation tree will be empty. Only the top-level node should be present.



Assign permissions to 'franklin'



Corresponding tree for 'franklin'

Next, I defined permissions to the ‘asesores’ user. This user can be defined conceptually as a group. All developers are defined as children of ‘asesores’. For this example I gave ‘asesores’ access to all the windows in the system.

[Table View] com_permis @public... | [Table View] com_usrdep @public...

compania	abr_format	nom_autori	permisos
DECSA	pep.conint	asesores	CIMER
DECSA	pep.explab	asesores	CIMER
DECSA	coi.sercia	asesores	CIMER
DECSA	coi.series	asesores	CIMER
DECSA	coi.paraco	asesores	CIMER
DECSA	coi.parame	asesores	CIMER
DECSA	coi.usuari	asesores	CIMER
DECSA	coi.usrdep	asesores	CIMER
DECSA	coi.permis	asesores	CIMER
DECSA	coi.navega	asesores	CIMER
DECSA	grd.menuex	asesores	CIMER

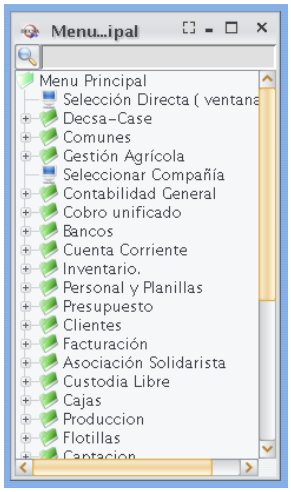
Some of the permissions for ‘asesores’

In the COM_USRDEP table I defined ‘franklin’ to be a child of ‘asesores’. Note that ‘franklin’ also has to be defined as a parent of himself. With this change, now ‘franklin’ should have access to all the windows for which ‘asesores’ has access. The screenshot shows that now the tree has been populated with all the options.

[Table View] com_permis @public... | [Table View] com_usrdep @public...

compania	usr_padre	usr_hijo	nom_usuar
DECSA	asesores	franklin	franklin
DECSA	franklin	franklin	franklin

‘asesores’ is parent of ‘franklin’



Corresponding tree for ‘franklin’

Validation of Approach

Have you built the system?

Yes I have. The database design, window convertor, window generator, menu navigation, favorite window navigation, window search and window designer have been built. All of the aspects are complete except the window designer which was kept as a prototype. In later projects, developers can build upon what was developed for the window designer. Evaluations were also carried out on the window generator and the access rights on the menu navigation tree.

Which part of the system is not working?

Through testing and actual use of the system, it has been verified that all the aspects of these project are working as expected. Again, the only part that is not ready for production is the window designer.

What experiments have you run?

Please refer to the *Evaluations* section. The two major tests were to verify the enforcement of access rights in the menu navigation tree and to verify the output of the window generator.

How will you know that the system works?

The menu navigation system was tested for access rights enforcement. Integration with the window generator was also tested. Basically, if the user selects a window, the corresponding window should be displayed on the screen by the window generator. Favorite windows and window searches are also returning the correct results.

Conclusions

Lessons Learned

Describe your experience and what you have learned thus far?

I have enjoyed this project. It has been a very-very-time-consuming project but I believe that it was well worth it. The benefits that the software company will receive from the deliverables of this project will be quite noticeable. In fact, they have asked me to give them alpha releases so they can start playing around with the windows and adjust them to their own business needs. During the presentation at the end of this course, I will present the window generator and menu navigation tree in a “real” environment with “real” data provided by the software company. Due to the ease of use of this generator, integration is also very straight-forward. And it is all backed up with a very strong database design, in my own opinion.

From the database aspect of the project, I was amazed with what could be done with views. The COM_PERUSR view combined with the COM_USRDEP table is probably the accomplishment that I am most proud of. It solved all the requirements and even more. Ideas such as negative permissions were like an impossible, and I achieved that functionality with the use of a view, a very complex view.

I believe I spent considerable more time on the database design than on the application implementation. The COM_PERUSR and COM_PENAVE views took a couple of weeks to come up with, finalize and test. But due to the time spent designing, the GUI implementation was very easy and straight-forward. The queries that needed to be executed were also very simple and straight-forward. So I guess that the moral of the story is that if you spend sufficient time designing, life will be easier later on. I was very lucky this time because I did not have to go back and change my design after I had started implementing the GUI. But for example, the database design changed considerably from what I had presented on October 11 for the project proposal.

What skills you are practicing or new tools and techniques you are working with, that you did not know before?

I worked with Eclipse for the GUI implementation and also to modify the SQL statements. I also used Navicat to look through the PostgreSQL tables and their data and I used PostgreSQL 8.3. My environment was a Mac with OS X, but I have tried my deliverables in both Linux and Windows and they work perfectly. To insert tables and data I used the command-line programs that come with PostgreSQL.

Member Contribution

Not applicable. I am the only member of this group.
But I have to note that I did receive help by PowerBuilder experts to create the PowerBuilder data exporter called ext_pb.

Future Work

The next step would be to create a fully-functional window designer.The software company could really benefit from this component. Especially when the developers have to modify window layouts.