the Utilities Top Menu, and the Publish diagrams. Option, could also be possible to store the diagram as a graphic file in the computer.

2nd step

Here, the developer chooses the Directory, the Format, and the File Name to store the diagram.

3rd step

Now, with the graphic file stored it is important to save the scripts of creation for all the objects in the diagram, because in this way the execution of "SchemaSpy" and, in fact, the "Publish Data Models" tool could work properly.
First of all, is important to select everything in the diagram, it is easy with the mouse or the "Ctrl" key. When everything is selected, just clicking the mouse right button (on one of the selected objects) and choosing the option Generate... the process will go ahead.

4th step
A new dialog window will appear, here, choose the *DDL Files Only as Target for Generation* and select the appropriate *Oracle* release.
Fill also in the *File prefix* and *Directory* fields, with the name of the patch and the name of the folder where the Oracle Designer scripts must be stored for a properly future use of "transform" tools to generate the ADP delivery.
Then clicking on *Start* button the generation goes ahead.
It is very important to choose *DDL Files Only* option inside the *Target for Generation*, because you need to have a target to compare with the Oracle Designer models, and generate scripts only for the changed objects.

The Oracle Designer application will be the main tool to develop patches and generate appropriate outputs, but in the framework (the subject of this project) there were only something like the Navigation flow as a Manual of Oracle Designer. The developers will be supposed to have this application installed and working in the computer.
Previous design considerations

After the project definition, it seems easy to see that the following three tools to develop will be in the same interface. Therefore, the design pattern applied in the Framework Design will be a FAÇADE PATTERN so as to put all the tools under the website interface.

Going forward with the project, the application of the Decorate pattern has also been necessary, in order to modify the outputs of Schema Spy tool (see the Developing of functionality two "Publish Data Models" section). A decoration structure will be created a decoration structure in order to display exactly the outputs requested by the users. Where the decorator will be the modifications of Schema Spy outputs, in order to display in one hand "complete diagrams of schemas" (represented by ConcreteComponent instance) and on the other hand "Small representations of the schemas".

Where: Component is the final output, ConcreteComponent is the output of the Schema Spy application, and Decorator is the structure created in this project.
Developing of functionality one "Patch Delivery"

This is the first functionality to develop, the target is to create a delivery of a patch from a SQL script. This delivery has to contain the same script, but for all the synonyms where is applicable.

When developing this tool, it is important to start with the framework development (as a website) also, and integrate the "Patch Delivery" tool and future tools when they will be created.

Previous tool

Currently, the developers, in order to generate the patch deliveries, do it in three steps:

- Creation of a SQL script.
- Store it in the local computer.
- Execute transform_sql Oracle Tool.

The creation of the SQL scripts is done with non graphical applications, then the developers just store the scripts in a linux box folder. At the end they execute the tool "transform_sql" (CLI with flags as follows):

```
/projects/obedelde/OracleTools/latest/bin/transform_sql.ksh -c configuration_dba.xml -i input_scripts -o output_scripts
```

After that if the patch implies the creation of some new tables, the developer has to modify the txt files from the distribution folder including the new tables (typing the names) in each schema file.
The idea for this tool is (the management of the creation of SQL scripts and storing them will be done by Oracle Designer) create an interface to manage the execution of transform_sql and new files addition in the appropriate schema file.

Functional Requirements

The functional requirements for this tool are the following:

- User friendly tool.
- The input has to be the output of Oracle Designer
- The output has to be the same as the current one.

The tool will be integrated in the website and the execution will be done with an easy form. To use the outputs of Oracle Designer, it will be mandatory to modify these outputs cause they are not in the proper format. Keeping the same tool as the one used currently (transform_sql.ksh from Oracle Tools) there will be the same kind of outputs used nowadays.

User friendly tool

Using an interface integrated in the website, and executing the tool just as a form, it should be enough to make that tool as "user friendly".
The input has to be the output of Oracle designer

Just transforming the output of OD in a proper input for transform_sql tool.

26-01-08_patch.tab

26-01-08_patch_tab.sql

The output has to be the same as the one used nowadays

There is no problem in satisfying this requirement since the same tool will be used.

Technologies used

The languages and technologies used to develop this tool are:

- HTML + CSS files, PHP and JAVASCRIPT to develop the interface.
- C programs to modify the scripts (outputs of Oracle Designer, txt files from /distribution folder...).
- Korn Shell to manage the execution of other C programs and scripts.

The election of these technologies are basically due to enviroment and background reasons. For example I had little knowledge about php, javascript and html before, and I think that these are common languages for web designing.

The C applications are motivated, because in the department of the company I work, the programming language used is C/C++.
The Korn shell as a scripting language is motivated for Oracle Tools already developed in my team, like the transform_sql tool that I will use in this tool.

When developing the website one Apache Server, with CGI files and SSH calls, the SSH calls will be used in order to access from the server to the developers local hosts and work with the scripts and outputs stored in developers sides. Further explanation of these technologies will be found in the section "Join tools together".

Navigation flow

This section is, using screenshots for an easy understanding, the explanation of the use cases of this tool.

The use cases for our developers are just the following two:

- Run the application.
- Watch the delivery scripts.

For this reason, in the website there are two buttons for this tool (aka Script Designer tool)

The button "Run" leads the user to the form to execute the tool, the other one "Scripts Delivery Folder" displays the folder where all the patches of the current user are stored and
permits to open and watch all the scripts in order to save the not empty, and create a proper delivery to ADP.

Run the application

1st step

It starts just clicking on the button "Run" in the "Script Designer tool" section of the Website as it is marked in the figure below.

2nd step

It appears the form to be filled in properly:

- "Source patch name" fill it in with the name of the Oracle Designer output.
- "Delivery patch name" the definitive name of the patch.
- "File distribution" choose the schema where all the new tables in the patch will be stored. It is only possible to choose one, because developers use to work in one schema each time.

The other fields are not alterable, they contain the user’s local machine and working directory.
A short Manual with the explanation to fill in the form properly it is also available.

3rd step

At this point, click on the Execute button and wait for the end of the execution, all changes and steps will be displayed in the screen.

When the execution has finished, just cancel the screen and the use case will be finished.

Watch the delivery scripts

1st step

As in the use case seen before, and like most of the use cases of the framework, it starts in the main page.

Just click on the “Script Delivery folder (outputs) button as it is marked in the picture below, and the use case starts.
2nd step

The developers show all their patches.
With all the synonyms applied properly.

As seen below, these two diagrams—one from the Server side and another one from the Client side—aim to show how these functionalities work.
Watch the patch scripts
Developing of functionality two “Publish Data Models”

This second functionality is a bit more complex than the first one. Basically, with the current program, it is only possible to have graphical representations of complete schemas, which are complicated to understand.

With this tool, it has to be allowed to display all the users working parts (not huge set of tables and relationships).

Previous tool

The current tool in charge to display graphical representations of the schemas is also an Oracle Tool (developed by OBE developers from Amadeus).

This Oracle Tool is linked directly with SchemaSpy application in charge to generate the output.

Then, the developers have just to execute the Oracle Tool “view_schema” and display the results as a website.

The main problems of this tool are:

- CLI to execute "view_schema".
- Not clear outputs, great amount of information in every diagram.
- Not possible to display only some sections of the schemas.
Functional Requirements

In order to solve the three main problems of the existing tool, this tool has to provide to developers, the following advantages:

- User friendly tool.
- Clear outputs.
- Display outputs for small parts of the schemas.

The first one, as all the other tools requirements of user friendly tools, will be solved with the integration of the tool in the same framework (website), the other two requirements are related themselves, and the original idea is to offer the same outputs than in the current tool (graphical display of all the schema) plus lots of user partitions of the schema.

User friendly tool

With an interface integrated in the website, and executing the tool just as a form, should be enough.

Now, the execution is just by command line:

```
/projects/obedelde/OracleTools/latest/bin/view_schema.ksh -c configuration_dba.xml -p fpm -o schema_spy_folder -s SCHEMA1,SCHEMA2 -u user -P password
```

With the new tool, the execution will be with a form, and the outputs will be integrated in the same website.

Clear outputs & Display outputs for small parts of the schemas
This requirement will be satisfied using the Oracle Designer outputs and executing directly the SchemaSpy application.

Now:

```
View_schema -> SchemaSpy
```

*Not clear outputs*

With the new tool:

```
View_schema -> SchemaSpy
```

```
SchemaSpy.
```

*Clear outputs and old kind of outputs*
Technologies used

The languages and technologies used to develop this tool are:

- HTML + CSS files, PHP and JAVASCRIPT to develop the interface.
- C programs to modify the scripts (outputs of Oracle Designer, the outputs of SchemaSpy and the new output website).
- Korn Shell to manage the execution of other C programs and scripts.

The election of these technologies are basically for environment and background reasons. For exemple I had little knowledge about php, javascript and html before, and I think that they are common languages for web designing.

The C applications are motivated because in the department of the company were I work, the programming language used is C/C++.

The Korn shell as a scripting language is motivated for Oracle Tools already developed in my team. For instance the transform_sql tool that I will use in this tool.

Navigation flow

This section is, using screenshots for an easy understanding, the explanation of the use cases of this tool.

The use cases for our developers are just the following two:

- Run the application.
- Watch the outputs.

For this reason, in the website there are two buttons for this tool (aka Publish Designer tool).

The button "Run" leads the user to the form to execute the tool, the other one "Schema Spy outputs" displays on the browser the output website of the tool, with two main sections, the "Public schemas" as in the current tool, and the "User schemas" with all schemas stored by the user, using Oracle Designer.
Run the application

1st step

It starts when clicking on the button “Run” in the “Publish Designer tool” section of the Website.

2nd step

It leads the developer to the following form.

To fill the form properly, in required to:

- “Username” and “password” of the database.
- “Configuration file” as filled by default, all developers have this file in their local machine.
- “Phase” to connect to the database.
- “Schema filters” to display these specific schemas.
- “Schema owners” the name of the owner of the schemas.
- “Database” the name of the database.

The other fields are not alterable, they contain the user’s local machine and working directory.
A short Manual with the explanation to fill in the form properly it is also available

**3rd step**

At this point, it is enough to click on the Execute button and wait for the end of the execution, all changes and steps will be displayed in the screen.

When the execution has finished, just switch off the screen and the use case will be finished.

**Watch the outputs**

**1st step**

As in the use case seen before, and like most of the use cases of the framework, it starts in the main page.

Just clicking in the “Schema Spy outputs” button as it is marked in the picture below the use case starts.
This button is linked to the main page of outputs stored in the server side. In this new page, there are two sections, the first one, "PUBLIC SCHEMAS" links the user to the original outputs of SchemaSpy, with the complete schemas’ display, the second one "USER SCHEMAS" is linked with the outputs generated by the users using Oracle Designer.

2nd step
3rd step: (Some screenshots of PUBLIC SCHEMAS)

SchemaSpy Analysis of REH9.DBANGIPFM_P

Generated by SchemaSpy on Wed Mar 19 16:54 MET 2008
Database Type: Oracle - Oracle9i Enterprise Edition Release 9.2.0.8.0 - 64bit Production With the Partitioning, OLAP and Oracle Data Mining options JServer Release 9.2.0.8.0 - Production

104 Tables:

<table>
<thead>
<tr>
<th>Table</th>
<th>Children</th>
<th>Parents</th>
<th>Rows</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>APE_CONFIG_VERSION</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>APE_PARAMETERS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>APE_PARAMETERS_EXT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CHELINE_PARAMETERS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COM/Publication_Target</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COM/Publication_Target_Address</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>COM_SCHEMA_Mapping</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>COM_SEQENCED_DELIVERY_LOGS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CRJ_AIRLINE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CRJ_AIRPORT_TERMINAL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CRJ_APPLICATION</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CRJ_APT_TABLE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CRJ_CARRIERN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CRJ_COUNTRY</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

SchemaSpy Analysis of REH9.DBANGIPFM_P - Relationships Graph

Generated by SchemaSpy on Wed Mar 19 16:54 MET 2008

Include implied relationships
Compact

Legend:
Primary key columns
Columns with indexes
Excluded column relations
n > number of related table
Please export this profile
4th step: (Some screenshots of USER SCHEMAS)
As seen below, two diagrams—one from the Server side and another one from the Client side—aim to show how these functionalities work.
Database Model Publishing

Copy of Schema Spy Output

Watch the output

Server Side

Client Side
Developing of functionality three “Developing tuning support tool”

This third tool is only an optimization of the existing tool.

Now, in Amadeus OBE, the Statspack tool is used to get information about the performances of the databases, and with its help, tune the queries and structures in the database. These “snapshots” (process that displays a report with all performances issues of Amadeus databases) are executed every 30 minutes, and the reports are stored and available on the intranet of global core.

With the new tool, nothing changes, but integrates on the framework the display of this reports. Moreover, only specific reports with the most useful information (for Database developers) that have been in a local repository will be directly and exclusively selected from this framework.

Previous tool

The previous tool will be the same, the “Statspack”, a free tool that offers executions against the databases, and provides the users a large number of reports with the performances issues of the database.

The problem, or modification, is in changing “how to display the reports” and “what kind of reports” display.

Then, the current problems are:

- Difficult to access.
- Difficult to get the proper reports.

As we can see in the following diagram flow, it is really uncomfortable to access to the information.

(Screenshots from the intranet of Global Core):
After nine steps the user can access to the report. And, what is more, it is difficult to be sure about if this report will contain the information the user needs.

**Functional Requirements**

Then, requirements to satisfy are:

- Easy interface to access.
- Specific, concrete reports.

To satisfy the first one, the new functionality will only have one screen where select and display at the same time the reports.

In order to satisfy the second requirement, now, the specific and useful reports are being executed every hour, and saved in a local repository.

Then this tool will be just a link to this repository, and there will be also available a link to the intranet (last steps) in order to allow the users select other kind of reports, more information and so on.
Technologies used

The languages and technologies used to develop this tool are:

- HTML + CSS files, PHP and JAVASCRIPT to develop the interface.
- In order to get the information of Applications and Databases xml parser inside the html code, in php (this information is in an Amadeus file called db_topology.xml) is also used.

Navigation flow

It seems easy to see that there are 2 use cases:

- Access to specific reports.
- Access to intranet reports.

Both start when clicking on the "Statspack tool" button on the website main page.

Then, the user is in one webpage where he/she can manage all the reports.
Here, we can see how execute/display the 2 use cases.

Displaying Snapshots

Zoom of the outputs:
Join tools together

As it has been said along the project, all the tools developed are in the same framework (website), this section only aims to explain further details about the web designing.

Firstly, an Apache webserver with PHP is installed in a specific linux box where I had root permissions.

All the tools, scripts and outputs are stored in the local machines of the developers, then the server needs to connect to the client side, execute everything and copy the outputs to the server side. To implement that, "public keys" and CGI's with SSH calls are used.

The CSS files are the same, with some modifications than the CSS of the Global Core Intranet.

Before the main webpage shown before in the project:

![Image of data model publishing tools]

There is now, another main page before that, with a guide for Oracle Designer, the downloads of the scripts, C programs and the script to allow the server connect to the client side.

![Image of data model publishing tools]
To go ahead just fill in the fields with the "username@machine" related with the linux box where the developers work, and the directory of work.

Here, the complete Diagram Flow of the application.
In the website there are a lot of Guides, Helps and Manuals too, that let surfers know what to do in every single moment.

Help buttons in order to explain, how to install the environment, use the tools, fill the forms...
Real plan of work done

At the end of the project, we could see some changes in the previous plan of work:

Gantt diagram for the already done project
<table>
<thead>
<tr>
<th>Task definition</th>
<th>Start</th>
<th>End</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge of company environment / overview of the</td>
<td>1-10-07</td>
<td>20-10-07</td>
<td>Get some background of the company department and project.</td>
</tr>
<tr>
<td>project</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slides with the knowledge</td>
<td>22-10-07</td>
<td>24-10-07</td>
<td>Prepare a presentation to my project director</td>
</tr>
<tr>
<td>Installation/configuration of Oracle Designer</td>
<td>24-10-07</td>
<td>31-10-07</td>
<td>Install tool.</td>
</tr>
<tr>
<td>Oracle Designer testing</td>
<td>31-10-07</td>
<td>14-11-07</td>
<td>Working with the tool.</td>
</tr>
<tr>
<td>Writing a manual of OD</td>
<td>31-10-07</td>
<td>19-11-07</td>
<td>Slides and document, how to use Oracle Designer.</td>
</tr>
<tr>
<td>Script Designer tool developing</td>
<td>19-11-07</td>
<td>3-12-07</td>
<td>Start with the first tool.</td>
</tr>
<tr>
<td>Test</td>
<td>3-12-07</td>
<td>6-12-07</td>
<td>Test.</td>
</tr>
<tr>
<td>Delivery-Document</td>
<td>3-12-07</td>
<td>7-12-07</td>
<td>Prepare slides and documentation to my project director.</td>
</tr>
<tr>
<td>Publish Designer tool developing</td>
<td>7-12-07</td>
<td>24-12-07</td>
<td>Start with the second tool.</td>
</tr>
<tr>
<td>Test</td>
<td>3-1-08</td>
<td>8-1-08</td>
<td>Test.</td>
</tr>
<tr>
<td>Delivery-Document</td>
<td>7-1-08</td>
<td>12-1-08</td>
<td>Prepare slides and documentation to my project director.</td>
</tr>
<tr>
<td>Snapshot tool Developing</td>
<td>14-1-08</td>
<td>26-1-08</td>
<td>Start with the last tool.</td>
</tr>
<tr>
<td>Test</td>
<td>28-1-08</td>
<td>2-2-08</td>
<td>Test.</td>
</tr>
<tr>
<td>Delivery-Document</td>
<td>28-1-08</td>
<td>5-2-08</td>
<td>Prepare slides and documentation to my project director.</td>
</tr>
<tr>
<td>Website Developing</td>
<td>3-12-07</td>
<td>18-2-08</td>
<td>Develop the interface integrating the already finished tools</td>
</tr>
<tr>
<td>Join test</td>
<td>5-2-08</td>
<td>1-3-08</td>
<td>Test everything toghether.</td>
</tr>
<tr>
<td>Demo</td>
<td>3-3-08</td>
<td>15-3-08</td>
<td>Prepare a demo to my project director and manager of the department.</td>
</tr>
<tr>
<td>Final Report</td>
<td>3-3-08</td>
<td>28-3-08</td>
<td>Join information and write the current report.</td>
</tr>
</tbody>
</table>

I took some unpaied days of holidays from 24 of December till 2 of January, both included.

After the first weeks of knowledge, I did some slides to make a feed-back with the team leader about what I understood about the company.

Another task included was the creation of an Oracle Designer Manual.
Future and updates

After delivery, present and make a demo to my team leader and manager, there are some items fixed to be changed in the near future of this application:

- After contacting with Oracle, Oracle Designer's price is high, and another free tool from Oracle (JDesigner) has to be tested and validated. In the first overview of the tool it is noticed that it could manage either DDL and SQL/PL.

- In the future, it has to be possible to store (in the database) all the releases of a Data Model, and allow the developers to work with the desired release.

- The framework has to allow the developers to execute snapshots when they want.
Project cost

This project has been developed in 26 weeks. I have taken 5 unpaid days, and there were 3 more free days (1st November, 25th December, 1st January)

Then, I have been working in it just 24 weeks.

24 weeks x 5 working days/week = 120 days.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Concept</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Travel expenses from Barcelona to Nice</td>
<td>Flight ticket</td>
<td>230€</td>
</tr>
<tr>
<td>Hotel, placement on arrival</td>
<td>14 nights x 50€</td>
<td>700€</td>
</tr>
<tr>
<td>Travel expenses for every day</td>
<td>6 months x 30€</td>
<td>180€</td>
</tr>
<tr>
<td>Canteen subvention</td>
<td>120 days x 4€</td>
<td>480€</td>
</tr>
<tr>
<td>Hand work</td>
<td>6 months x 1800€</td>
<td>10080€</td>
</tr>
<tr>
<td><strong>TOTAL COST</strong></td>
<td></td>
<td>12390€</td>
</tr>
</tbody>
</table>

If the company had paid the project to another company, it would have cost:

24 weeks x 37 hours/week = 888 hours of work.

Around 30€/hour.

**TOTAL COST: 888 hours x 30€/hour** 26640€