Reengineering Project: Database Optimization and Migration
Abstract

The purpose of this project is to help a growing company reform their static information system to a dynamic system compatible with growth. The solution consists of migrating the legacy system that they have in FileMaker to an open technology platform. To solve this specific problem “patterns” have been used and this project explains these general solution “patterns”. We understand patterns like a generic solution to persistent design problems. General solutions will not only be useful in this specific problem, but they will be useful in all kind of similar migration projects as well. This thesis gives a detailed explanation of how to apply these patterns into the AEMI specific problem and how they can be useful in the migration process.

The solution of the problem then, is following the advice of “patterns” to achieve our goals; these goals are the requirements that are given from the company supervisor. After the migration process a redesign process must be done in order to organize the information. This redesign consists of organizing the migrated information as well as adding the new information in the correct place.

This thesis focuses on the process of migrating from a legacy system to a MySQL system through the use of the generic solution called “patterns”. The final result is a MySQL database with all the old and new information together in a more adaptable platform for the company’s scalability.

Keywords: system reengineering, pattern, group of patterns, migration process, migration strategies.
## Content

1. Introduction.................................................................................................................. 1
  1.1 Problem and Motivation .......................................................................................... 1
  1.2 Goals and Criteria ................................................................................................. 1
  1.3 Outline .................................................................................................................... 2
2. Setting Direction........................................................................................................... 3
  2.1 Agree on Maxims .................................................................................................. 3
  2.2 Most Valuable First ............................................................................................. 3
  2.3 “If It Ain’t Broke, Don’t Fix it” ........................................................................... 4
  2.4 Keep it Simple ....................................................................................................... 4
  2.5 Organize a successful meeting ............................................................................. 4
3. First Contact .................................................................................................................. 6
  3.1 General Interview ............................................................................................... 6
  3.2 Mock installation ................................................................................................. 7
  3.3 Review the interview ......................................................................................... 7
  3.4 Project Plan .......................................................................................................... 10
    3.4.1 Description ..................................................................................................... 11
    3.4.2 AEMI Project Plan ...................................................................................... 11
4. Migration Strategies ..................................................................................................... 13
  4.1 Involve the Users ................................................................................................. 13
  4.2 Migrate Systems Incrementally ........................................................................ 13
5. Migration ....................................................................................................................... 15
  5.1 Why Migrate from FileMaker to MySQL? ......................................................... 15
  5.2 Technical Challenges with Migration ................................................................ 15
  5.3 Migration Process ............................................................................................... 17
  5.4 AEMI Migration Process ..................................................................................... 19
    5.4.1 Introduction .................................................................................................. 19
    5.4.2 Database tables design .............................................................................. 19
    5.4.3 Apply diagrams in MySQL ........................................................................ 20
    5.4.4 Export Information .................................................................................... 20
    5.4.5 Import Information ..................................................................................... 22
    5.4.6 Redistribute Information .......................................................................... 24
    5.4.7 Add new information ............................................................................... 30
6. Conclusions and Future Work .................................................................................... 35
  6.1 Conclusion ........................................................................................................... 35
  6.2 Future Work ......................................................................................................... 36
7. Bibliography ................................................................................................................ 37
Figure List

Figure 3.1 FileMaker Screenshoot1.................................................................8
Figure 3.2: FileMaker Screenshoot2..............................................................8
Figure 3.3: FileMaker Screenshoot3.............................................................9
Figure 3.4: FileMaker Screenshoot4.............................................................9
Figure 3.5: Relevant fields .........................................................................10
Figure 5.1: Migration Flowchart.................................................................17
Figure 5.2: First design...............................................................................19
Figure 5.3: Export Fields FileMaker Screenshot...........................................20
Figure 5.4: Export fields order FileMaker Screenshot...................................21
Figure 5.5: AEMI.csv screenshot.................................................................22
Figure 5.6: MySQL admin import option.....................................................23
Figure 5.7: MySQL database after migration process..................................23
Figure 5.8: Second design diagram............................................................24
Figure 5.9: Final diagram..........................................................................30
Figure 6.1: Database field’s names..............................................................35
Figure 6.2: “cursos_fcontinuada” table information.....................................36
1. Introduction

Many companies that are growing will have the same problem: their data system is not sufficient for their needs. Due to their insufficient system, they will need to do a reengineering process.

Why use patterns in the solution of a reengineering project?

“Design patterns are generic solutions to recurring design problems” [1]

All the people can do a reengineering project following their own rules, but it would be a good idea to follow the same patterns because we can learn from the people that did similar projects in the past.

We can think that a pattern is only related to an Object – Orientated language but this is a mistake. A pattern is a generic solution and for this reason can be adapted or modified to achieve our goals. And this is exactly what the project is about. Adapt and modify the patterns to solve the AEMI Company problem of the legacy system.

For example in this project we will work from the software migration view but also how to properly handle company relations and how the work must be done.

1.1 Problem and Motivation

Why the company needs to reengineer the system?

All the information in a legacy system is valuable and has to be kept.

In the case of the AEMI Company they have more than 2000 members that drop out of the association but they still need to keep this information, because it is always important to know the market, and the profile of future customers.

This company needs to reengineer the software for the following reasons:

- Improve performance; the company is growing and they need to add information and delete unnecessary fields.
- To earn money; now the database is in FileMaker and they have to pay fees for the program. They would like to migrate the system to open technology software like MySQL, and in this case they would not have to pay fees anymore.
- Improve mobility; for the moment the database is with FileMaker and only can be used in one office. They are going to open new offices, and for this reason they need to migrate the software from static software (FileMaker) to dynamic software (MySQL). The new software will be used in all the offices at the same time.
- Security reasons; nowadays this database is inside the computers in the headquarters. They are doing security copies, but it is still not enough. It is more secure to have the database in a server outside the headquarters.
- Reduce human dependencies; because the database will be linked with the website there will be no need for interpersonal communication.

1.2 Goals and Criteria

The project is to adapt and modify patterns to solve the AEMI Company problem of the legacy system. With the help of these adapted patterns we will complete do the migration process.

The main goal of the thesis is to assess the applicability and usefulness of the "modified patterns" used in the migration project for the AEMI Company. We can measure the success of the thesis if all the adapted patterns will help to solve the AEMI company problem and some similar migration problems.
1.3 Outline

Chapter 2 contains the project priorities and describes the line that we should follow to do a success work. In Chapter 3 a first project contact and the project scope is presented through patterns. Chapter 4 contains the strategies that we will use in the migration process. Chapter 5 contains the migration from the legacy system to an open technology platform following the appropriate patterns. Finally, in Chapter 6 we will summarize the conclusions that we have taken while developing this project.
2. Setting Direction

In this chapter, the reengineering project can be started. This chapter contains the important project priorities and describes the line that it has to follow.

For example, the management direction will want to be more competitive in the market and reduce maintenance costs as well. Users and workers want easy functionality, and developers want an easy job.

Setting Direction is a group of patterns that are going to help to find out: Where to start the project? Which problems focus on first? Who is our target group?
In the following patterns these questions will be solved.

2.1 Agree on Maxims

Pattern purpose:
“Establish the key priorities for the project and identify guiding principles that will help the team to stay on track.” [2]

Pattern application in the project:
In this point the key priorities of the project have to be defined.
- Analyze the legacy system.
- Discard the unnecessary fields.
- Migrate the system to open technology.
- Add new classes.
- Optimize the searches.

The purpose of this pattern in the project is to have a general idea of all the important principles that have to be followed correctly in order for the project to be successful.

2.2 Most Valuable First

Pattern purpose:
The title of this pattern is the answer to the question of which problems focus on first. The most valuable problems have to be focus on first.

The difference between what the customer thinks is important and what we think is valuable have to be separate. We should start with the main problems that affect the customers.

The difficulty in this pattern is defined by answering who is my customer and what is valuable.

Pattern application in the project:
The first meeting with the company was focused on this point. In this meeting I spoke with the company workers directly.

They only want an easy platform, easy to use. After that I spoke with the company project supervisor and the most valuable point for him was not lose anything kept in the database. The company supervisor stated that all the fields are important regardless whether we think that some data is irrelevant for the new system or not. They did not put any restriction on how to do the migration only that the old information must appear in the new system.

Pattern evaluation:
This pattern is valuable because helps understand the main problems from the company's point of view and this will help us have a better final product. The company may not always be an expert in this field and for this reason sometimes this pattern does not have as much importance. Then we should advise the company in order to help them to realize what the main problems are.
2.3 “If It Ain’t Broke, Don’t Fix it”

Pattern purpose:
This pattern talks about which parts of the old system have to be reengineered and which we did not have to touch.

“We should fix problems not symptoms and if it is not broke, do not fix it” [2]

Pattern evaluation:
I disagree in the part of we should fix problems not symptoms, of course that the problems have to be solved but the symptoms as well because one symptom in the future can become a problem or a big problem.

I agree in the sentence If it not broke, do not fix it. But I would like to add: if the company has a bad design or structure please fix that first. This is the case of AEMI Company they have a bad database design, which we can see in the following chapters.

2.4 Keep it Simple

Pattern purpose:
“Do the simplest thing that will work” [3]

This is a short sentence but explains everything. The best way is to use simple solutions, which will produce an excellent product.

Pattern application in the project:
This pattern will be used in the migration chapter and in the meetings with the company supervisor when we propose designs for the new database.

Pattern evaluation:
If we take simple solutions to solve a big problem this solution can be analyzed for the customers faster and easier. On the other hand, if we take complex solutions it will be harder to fix if something is wrong.

2.5 Organize a successful meeting

Pattern purpose:
The reason for this pattern is to know how to organize an excellent meeting with the company.

There are seven steps to hold a successful meeting:

Organization: Never suddenly call a meeting. Notice should be given at least one day in advance, unless it is an urgent meeting that could not be avoided.
The meeting should last about thirty minutes and should focus on treating a single topic if there must be two topics, thirty minutes should be added. It is not good practice to hold meetings with a lot of topics to work with.
Be on time. The meeting has to start on time and end on time.

Participation: Include only the people that are affected or interested in the topic of the meeting and never let a person send a replacement if he/she is important for the meeting.

Preparation: Never do a meeting without having prepared. The preliminary arrangements can determine the success of a meeting.
The meeting room should be well lit and ventilated and away from distractions. In addition try to avoid interruptions like cell phones or other types of communication devices.

Presentation: Those who will attend the meeting have to feel comfortable. And we should be honest. Begin the discussion topic in a practical way to engage all the participants. Start by introducing the topic and say in simple words what the meeting is about.
Technical discussions: Encourage the group to exchange different point of views. Ask questions that cannot be answered by “yes” or “no”. Encourage group members to think individually or collectively, and then let them talk and ask questions.

According to the group: It is important for the group to agree with the improving tasks assigned and finally write down all the suggestions.

Monitoring: Summarize what was discussed to the attendees, do a brief review on what has been working and make a decision and then apply the decision.

Pattern application in the project:
These ideas have been used every time that we held a meeting with the company.

Pattern evaluation:
We did not use all the steps from this pattern, for this reason the applicability of this pattern was low but we considered it good practice to apply all of them.

After we know how to start the project and we have a clear idea of the project priorities, is time to continue with the overview of the legacy system. In the following chapters we will explain the guideline that the reengineering project will have.
3. First Contact

Time is valuable for this reason we do not have to waste it. Trying to understand the legacy system, we can waste a lot of time. But following the next group of patterns in the next flowchart we will do the work faster and structured. We should apply these patterns in a period of a week and use them to take all the information that we can from the legacy system.

3.1 General Interview

Pattern purpose:
This first meeting provides a general idea of where I should focus first and provided a lot of new data to process.

Pattern application in the project:
In the first place I had the interview through Skype with the end-users. They commented on how the system looks from their point of view and the general usage that they have been doing. On the other hand the manager informed me how the system should be and how it should fit in the general system. At the end of this meeting they provided me the FileMaker database.

After this meeting I have to say that I was satisfied that I asked everything and felt that I had everything under control. In a few days though, I quickly realized that I had a lot of new questions and a lot of missing data.

Pattern evaluation:
In Addition to be clear this pattern is only a general interview of the legacy system, not about the design of the new system. This patterns helps to understand the usages of the old system and all the information that is relevant.
This pattern is useful because it provides us the tools to ask good questions. For example, we can ask all that we do not understand of the legacy system. In my opinion we should take advantage of this pattern because is faster ask that search.

3.2 Mock installation
Pattern purpose:
This pattern consists of a first contact with the old system in order to understand the general usages.
Pattern application in the project:
The legacy system had been installed. The version was File Maker Pro10.
I did not have any previous knowledge about this program so I needed to gain a general knowledge quickly. I tried to read all the documentation about FileMaker that was important, to provide me with a general knowledge, you did not need to be an expert on this program.

3.2 Interviews during a Demo
Pattern purpose:
This pattern consists of seeing when the workers are doing their normal job and ask some appropriate questions to understand how the system works and start understanding what they want from the new system.
Pattern evaluation:
For this pattern it is important to do this because it helps to increase our credibility. If we do a good interview properly asking questions, it shows the interviewee that we have a genuine interest in their opinions.
I remember one of the first meetings one of the workers told me: “We do not need anything, this software works perfectly” and I was a bit afraid, but after an interview during a demo both realized that this was not true, as we found some errors. After a while she told me that she was scared about the new system. She was concerned about whether she would need a lot of training to know how it works, because she does not have a lot of computer knowledge and thought that her job was in danger. I assured her that with the new system her job would be simpler rather than more complex.
Pattern application in the project:
I did four meetings in which they had two presentations with all the fields. I asked which information is important to keep it for the new system and these are the main results:
The Figure 3.1 shows the general view of all the tables in FileMaker from the legacy system, on the other hand the Figure 3.2 shows the fields that the supervisor did not want to keep for the new system with a red X.
Figure 3.3 and Figure 3.4 are exactly the same. Figure 3.3 shows the bank tables and Figure 3.4 the fields that the supervisor thought that were not necessary to keep for the new system.
3.3 Review the interview.

Pattern purpose:
This pattern consists of putting all the information from the meetings together.

Pattern application in the project:
A class diagram has been used to see all the fields from the previous figures. This Figure 3.5 shows all the relevant fields that the supervisor wants to keep for the new system. Apart from that we can appreciate that is a poor design because all the information is kept in one table. But we are not here to speak about that we are here to migrate and to improve the design later. We can see this work in the next chapters.

I added a number at the beginning of every field to make it easier during the migrating and redesign processes.

In conclusion we have seventy-six fields that we will have to migrate.

Pattern evaluation:
The usefulness of this pattern consists of having a summary of all the interviews that we had been doing to provide us a dead line for the project.
3.4. Project Plan

3.4.1. Description

Once all the necessary information is gathered for the project it is time to create the project plan.

This pattern consist of:

- Project Scope: it is a short paragraph, usually half a page describing the project including the goals and the criteria that we will use to verify that we have met the set goals. To write about that is important in the pattern “Agree on Maxims” which establish the key priorities for the project.

- Opportunities: Is a list of items that we had discovered during the “First Contact” pattern. Basically this is a list of factors that we will expect to contribute to achieve the project goals.

- Risks: list of items that may cause problems during the project development. Like missing code libraries, test suites, etc… Apart from that we should evaluate the impact of every risk (high, moderate, low) and pay more attention to the critical risks.

- Go / No-go decision: In this point we will decide if the project should be continued or cancelled. A good strategy is to use the risk list to argue about the final decision.

- Activities: Only in case of go decision. Explain how we would try to reach the project goals. The list of activities should give the opportunities to reduce the critical risks and attain the project goals.

Pattern application in the project is explained in the chapter 3.4.2.

3.4.2 AEMI Project Plan

Project Scope

The AEMI Company have a members database in FileMaker, this company will grow quickly because they have a new contract with the government. For this reason, they want to migrate all the legacy system where not all the information is useful to a new open platform system. After doing all the migration process a redesign has to be done to add more information.

The project consists of two parts: the migration and the redesign.

Company Project Goal Criteria

- Satisfy the company requirements.
- Satisfy the company workers that will use the new system.
- New system has to be secure, reliable and fast.
- New system has to be simple.

Company Project Measure Success

- To measure the workers satisfaction with the new system, statistics and questionnaires between them will be done.

Opportunities

- Readability of the legacy system.
- Non-skilled maintainers. It is considered an opportunity because they will need our future help to maintain the new system.
Risks
- Is difficult to export photos in FileMaker. (Low)
- Bad legacy system design. (High)
- We do not have any primary key in the FileMakers tables, no unique field. (Moderate)
- All the information in the legacy system in not introduced. (Low)
- Information in some fields has to be distributed in to multiple sorted fields. (Moderate)
- Missing two tables of the FileMaker database. (Low)

Go/No-go decision
After evaluating the risks and the opportunities the decision is “Go”.

Activities
To solve the project the reengineering lifecycle will be used. This information it is taken from [2]

Reverse Engineering:
- Setting Direction: find the best strategy; find the main problems that exist and focus the effort on these obstacles.
- First Contact:
  - Learning about the context of the project through discussions with the workers that are maintaining the previous system.
  - Gain a first impression of the code.
  - Obtain a first impression of the functionality of the software.
  - Installing the system and recompiling the code.
  - Initial Understanding.
  - Analyze the data.

Reengineering:
- Migration Strategies:
  - Decide what is the best migration strategy and what is the best platform to make it. Make a plan to do the migration correctly.
- Optimize the new system:
  - Choosing optimal data types.
  - Indexing basis and strategies.
  - Good design.

Pattern evaluation:
This pattern is valuable because it is a document that the company approved that works as a guideline for the project. Also this document helps to evaluate the risks and the opportunities of the project.
4. Migration Strategies

The project is underway and the preview patterns give us a good understanding of the legacy system and now it is time to migrate.

The main forces that we should follow in this chapter are:
- Big-bang migration carries a high risk of failure.
- Introducing too many changes at once may alienate users.
- Constant feedback helps you stay on track.
- Users have to get their work done; they do not want to be distracted by incomplete solution.
- Legacy data must survive while the system is being used.” [2]

This chapter covers the patterns that will help us to stay on track in the migration process, and will provide some clues of how to migrate the system properly.

4.1 Involve the Users

Pattern purpose:
“Maximize acceptance of changes by involving the users at every step.” [5]

Sometimes this is difficult because the workers know how the old system works and how to manage the problems. Normally the workers do not like to learn something that will be new, they like a simple job.

To try to get the workers/users involved in the development we should try to follow these steps:
- Get the users to tell you where their priorities lie.
- Start with Most Valuable First pattern.
- Break the priorities down into small steps that can be delivered in regular increments.
- Create an environment that will encourage contact between users and developers.
- Establish simple procedures to obtain an easy feedback.“ [2]

Pattern application in the project:
Firstly we should hold a meeting only to listen the users/workers, they will tell us the key priorities for the project, then with this information start with the pattern “Most Valuable First”, dividing the information into small steps and trying to get all the feedback that we can from the users/workers.

Pattern evaluation:
A good method is to show to the users results constantly, for example every two weeks; it will increase confidence and create a good atmosphere that can help us to do the work.

4.2 Migrate Systems Incrementally

Pattern purpose:
We should try to avoid the complexity in the reengineering project by developing the new system in small increments.

Pattern application in the project:
The application of this pattern in the project is explained in chapter 5 in the migration process.

Pattern evaluation:
This is considered one of the most important patterns in this project. To evaluate the usefulness of this pattern we should follow these steps:
- Decompose the legacy system into parts.
- Choose one part to tackle at a time.
- Put tests in place for that part and the parts that depends on it.
- Take appropriate steps to wrap.
- Deploy the updated component and obtain feedback.
- Iterate.” [2]
5. Migration.

FileMaker allows the people to create a quick data base application and they do not need a lot of knowledge. Usually FileMaker is used for people that work independently from a corporation. However, when the number of users/workers grows it may become necessary to do a migration process to an open platform, to MySQL for example. MySQL offers a big number of hosting options and better adaptability. This chapter cover the procedures involved with migrating from FileMaker10 database running on Windows, to a MySQL database running on any platform. This chapter is based on the text from [9].

5.1 Why Migrate from FileMaker to MySQL?

MySQL is the most popular open technology database that is currently available. This popularity gives us a wide availability of MySQL servers. Almost all the ISPs offer MySQL database hosting inside the web hosting plans with an economical price. On the other hand with FileMaker database you pay more and it is installed in a limited number of hosting providers.

MySQL offers a better adaptability and features. For example FileMaker allows around 250 simultaneous users and MySQL allow thousands of simultaneous users. Currently the AEMI company has around 150 simultaneous users but it is expected that will be doubled in a few years.

Database replication is the most important failover capability that is crucial for AEMI Company and FileMaker does not provide this service.

MySQL offers different types of connectivity options for all platforms, for example Perl, PHP and Java, also there are no limits to the number of hosts or users that will use the database.

5.2 Technical Challenges with Migration

We will use a manual migrating FileMaker database, some of the technical challenges are changing column names, work with large numbers, migrate more than 255 characters and the most difficult: migrate images from FileFilmaker.

- Field Name problems:
  One of the first tasks was to change the database names to remove all the special characters. FileMaker field names can contain characters that are illegal for MySQL column name, when we create the MySQL table the special characters should be deleted from the field names, for example the spaces have to be changed to underscore. One important issue is that the column names in MySQL cannot be longer than 64 characters.

- FileMaker Fields:
  To do this project I worked with FileMaker Pro 10 but the original AEMI database was with FileMaker 7 this was a problem because FileMaker7 contains global fields used to store information types and the container fields cannot be exported.
  In FileMaker the software handles the calculation features, during the record insertion this software we can find it in the interface to the database. Instead MySQL offer his own triggers and procedures to do the same functions.
  To try to fix the problem of having too many fields and the possibility of exceeding the maximum number of columns that can be created with MySQL, we should try to remove all global fields, summary fields and calculation fields from the FileMaker database.
- Large Text Fields:
FileMaker7 can store around 2GB of text inside each text field. In contrast MySQL the longest type that we can use is VARCHAR that can only store up to 255 characters. Then for example the fields like “Observacions” and “Data_Baixa” have to be migrated to MySQL TEXT columns, which can store 64,000 characters for every record. If we can see that 64,000 characters is not enough we can changed to LONGTEXT then we will not have any problem.

- Repeating Fields:
Another problem is that in MySQL we do not have any data type equivalent for repeating FileMaker fields. Thus we should try to migrate these fields like text fields, for example, the numeric fields that contain repeating values we have to convert these fields into text fields, otherwise MySQL only will recognize the first character from every numeric field.

- Container Fields:
This is in my opinion one of the biggest technical challenges. Starting from the beginning, container fields are the way that FileMaker store pictures, sounds and video. The big problem is that we are working in FileMaker 7 and in this version it is impossible to migrate the container fields. I tried to migrate the fields into a LONGBLOB type, a Binary type but it is not working. Instead, if we are working with the last version of FileMaker pro 10 then is it possible but only with the new information that we introduced not with the old information that came from FileMaker.

A possible solution is to export the images, in a semi-manual way, by combining FileMaker scripting, AppleScript and shell script.

   The idea is: try to keep all the images in the clipboard, following these steps:
   Iterate all the records.
   Copy every image container into the clipboard.
   Open a file using AppleScript.
   Write all the information from the clipboard to the file.
   Close the file.
   Now we should move this file into a directory with a unique filename.

This is a semi-manual way to do the difficult work to keep all the old images.

- FileMaker Calculations fields and FileMaker Scripts:
We do not have these kinds of fields but in case we do have it, we would have to re-engineer this fields with a specific code application or a remote trigger in case of the script.
5.3 Migration Process

The flowchart in Figure 5.1 shows the migration process that has been followed.

1.- Define MySql Table

2.- Define MySql Column Names from FileMaker Fields

3.- Determine Maximum Field sizes

4.- Create MySQL Table

5.- Transfer Data via csv file from FileMaker to MySQL

Figure 5.1: Migration Flowchart
Define MySQL Table Name: In our case we have one table in the first migration step, which is not essential to be the same name with the FileMaker database. In this first step we should be aware if the Filemaker database includes spaces or special characters, which are not compatible with MySQL table name. But it is good practice to have the same table name when we transfer the information in a csv file.

Define MySQL Column Name from FileMaker Fields: We need to rename the fields that have spaces and special characters because cannot be used as MySQL column name. We also have to take care if the column name is too long because FileMaker allows that and after migration we can find problems.

Determine Maximum Field sizes: Firstly we have to determine the maximum amount of data stored inside each FileMaker field in all the records from the database, after we can determine the MySQL column sizes. The problem is there are not any options in FileMaker to find this information for this reason we should do a manual statistic. This part is important because if we choose a small size when we migrate the program will truncate the information.

Create MySQL Table: When the previous steps are complete the MySQL table can be created. We also have to make the decision about the column types. For example in Filemaker7 the container field can store around 2Gb of information, but now the text field contains much less information.

If the text field contains under 255 characters we can use Varchar columns type. Otherwise if the text field contains between 255 and 64000 characters we can use Text columns fields, the same case if Filemaker has LongText fields.

There is also another problem with the Date field. In the migration process to MySQL we should change Date field to Varchar if we do not do that the date will not arrive to the new system.

Transfer Data via csv file from FileMaker to MySQL:
1.- If we use phpMyAdmin it will be easiest.
2.- Export the FileMaker database with the option "<< .csv >>" file.
   • Inside FileMaker follow this steps:
     - Exports separating the fields by ‘,’ (comma).
   • From phpMyAdmin we have to follow these steps:
     - The table and the database have to be created
     - The table must have the same fields as FileMaker, otherwise it will not work.
     - Beware with the Date and Time format fields. If we do not change to Varchar usually it will not working.
3.- Change the exported file format from Unicode UTF-16 to UTF-8.
4.- Go to phpMyAdmin import option in our database.
5.- Select file format. CVS with the option to Load Data.
6. Follow these guidelines:
   • Replace table data with file
   • Fields finished with “,” (comma)
   • Fields enclosed by ““ ”
   • Escape character “\”
   • Lines ending in “\r” -this is very important because is the way that FileMaker export when change the line.
7.- We will have our tables in mySQL database in a few milliseconds.
5.4 AEMI Migration Process

5.4.1 Introduction

Now is the moment to apply all the previous patterns in a practical case. We divided this chapter in seven tasks. First, an accurate UML-design should be done because this work is essential for the future interface developing task. Secondly these diagrams have to be applied in MySQL database, when this part is done the information has to be exported from FileMaker and imported to MySQL. Once this is completed we should redistribute this information following our previous UML design. The last task is to add the new UML tables for the final design.

5.4.2 Database tables design

Following the patterns in the chapter 3 “First Contact”, around five meetings were needed to find a solution that the company approved. In the last meeting I presented three diagrams. Why three diagrams? Because following the pattern “Migrate Systems Incrementally” states that it is better migrate the system in a small increments.

The first diagram (Figure 5.2) shows how the information is kept in FileMaker, in this system they only have one big table and they always work with that. I have to say that it is not a big company and they have medium computer knowledge.

<table>
<thead>
<tr>
<th>Table</th>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socio1</td>
<td>name</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>apellidos</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>cedula</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>cedula_postal</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>poblacion</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>provincia</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>altura</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>edad</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>nacionalidad</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>email</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>fono</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>phone</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>fax</td>
<td>varchar(50)</td>
</tr>
<tr>
<td></td>
<td>postal_code</td>
<td>varchar(50)</td>
</tr>
</tbody>
</table>

Figure 5.2: First design

I wrote a number before every field because will be easier follow the migration process when we distribute this information in different tables.

All of the information that will be migrated is the information that the company wanted to keep. In the next sub-chapters the other two diagrams will be showed.
5.4.3 Apply diagrams in MySQL

Following the pattern “Migrate Systems Incrementally”, we will migrate the big table first, and in the chapter 5.5.6 we will apply the diagram two and in the chapter 5.5.7 the diagram three.

Following the first diagram (Figure 5.2) we will create seventy-six columns (fields) in MySQL with the same names that have in FileMaker. Is not necessary that they have the same name, it will only be mandatory in the Export process. After this task, the following table will show the FileMaker data types and the correct column type in MySQL:

<table>
<thead>
<tr>
<th>FileMaker Field Type</th>
<th>MySQL Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Int / Double</td>
</tr>
<tr>
<td>Text</td>
<td>Text or LongText (&gt;255 bytes)</td>
</tr>
<tr>
<td>Text</td>
<td>Varchar(255)</td>
</tr>
<tr>
<td>Date/Time</td>
<td>Varchar(20)</td>
</tr>
<tr>
<td>Container</td>
<td>LongBlob</td>
</tr>
</tbody>
</table>

Table 1: Data types migration

After introduce the Name and the Type we will chose the codification UTF-8, in the next chapter we will explain why we chose this codification.

5.4.4 Export Information

The first task in the export process is to launch FileMaker10 pro, open the AEMI.fp7 database file. We will select Export Fields from the File menu then will bring up a dialog as displayed in Figure 5.3.

![Figure 5.3: Export Fields FileMaker Screenshot](image)

We have to write the name of the file and after select the extract file type that we want. In our case the best is .csv (comma separated values) because it is easy to import in MySQL later on. CSV files are a type of single open document to represent data in a table, in which the columns are separated by commas or semicolon where the comma is the decimal like in Spain or France, and the rows by newlines. The fields that will contain a comma, new line or double quote must be enclosed in double quotes. (Figure 5.5)
After this process we accept and that will bring up the Define Fields dialog. All the fields inside the FileMaker database will be listed in this dialog. We will use this information to do a list of MySQL column names.

In the left side of Figure 5.4 we can see all the fields of the FileMaker database and in the right side all the fields that we want to export, in this point the order is important because we will export in a csv file. We will follow the order that we have in the first diagram (Figure 5.2).

At the bottom of the Figure 5.4 in “options” we can see that the codification method is “Unicode (UTF - 16)” we should select this because if we select for example “Unicode (UTF - 8)” we will lose a lot of characters like “‘” or “ç” or “ñ”.

After we complete all this work we will have an AEMI.csv file like this: (Figure 5.5). In this Figure we can appreciate that every fields is closed by “’” and the fields separates between them by “,” we have seventy six fields for every member.
5.4.5 Import Information

Before importing the information we have to solve a problem, when we did the export process we exported the information using UTF-16 because if we exported using UTF-8 we would lose some characters. Now we will have a problem because MySQL only accepts UTF-8. To solve this problem we can use for example this free web: “http://www.fileformat.info/convert/text/utf2utf.htm” which converts our file to UTF-8 without losing any information. Also we can use an Apple program like TextEdit, which is faster and more secure. It is more secure because in the other case we have to upload our file to the internet and these files contain private information from our company, which might cause problems in the future.

Then using TextEdit we only have to save the file with the option UTF-8 and then the problem is solved and we have our field ready to import to MySQL.

Now we can start with the import process. The first task is open phpMyAdmin in our server and the import option.

In the Location of the text file select our modified file “AEMI.csv” and in the Character set of the file select UTF8. After doing that, inside the table “Format of
imported file” select CSV with the option to Load Data. And the last step is: (Figure 5.6)
- Replace table data with file
- Fields terminated by “,” (comma)
- Fields enclosed by “ “
- Fields escaped by “\ ”
- Lines terminated by “\r ”

Then in a few milliseconds we will have our information migrated to MySQL (Figure 5.7) in the same order that was in the csv file.
5.4.6 Redistribute Information

Trying to follow the pattern “Involve the Users” a second design should be done (Figure 5.8). Working together with some employees we tried to improve the organization of the information, because before the migration process all the information was only in one table. Now the idea is to do this redesign in MySQL without losing any data. In conclusion we have to change our first design (Figure 5.2) and transform into (Figure 5.8). Because the employees do not have any computer knowledge we did this design between the class diagram and the Entity-relationship model but it shows clearly what the company needs are.

Figure 5.8: Second design diagram
In this point an important problem is found. In all the classes there is not a primary key. For example, if we work with "ID" as a primary key to make a connection with the other classes we find that not all the members have an "ID". If everyone does not have an "ID" this field cannot be used as a primary key and we realize that we do not have any other specific field that can do this job.

To solve this problem I created a script that creates a new field that will be an auto increment number, then we will have a primary key to work with.

We will start explaining the classes that are involved and creating the corresponding tables with the fields.

**Class “soci2” = member.**

```sql
CREATE TABLE `socis2` (  
`id_soci` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
`n_soci` INT(11) NOT NULL,  
'nif' VARCHAR(50) NOT NULL,  
'nom' VARCHAR(50) NOT NULL,  
'cognoms' VARCHAR(50) NOT NULL,  
'adreca' VARCHAR(50) NOT NULL,  
'codi_postal' VARCHAR(50) NOT NULL,  
'poblacio' VARCHAR(50) NOT NULL,  
'provincia' VARCHAR(50) NOT NULL,  
'pais' VARCHAR(50) NOT NULL,  
'tel1' VARCHAR(50) NOT NULL,  
'tel2' VARCHAR(50) NOT NULL,  
'tel3' VARCHAR(50) NOT NULL,  
'fax' VARCHAR(50) NOT NULL,  
'email' VARCHAR(50) NOT NULL,  
'adreca2' VARCHAR(50) NOT NULL,  
'professio' VARCHAR(50) NOT NULL,  
'estat_laboral' VARCHAR(50) NOT NULL,  
'foto_any' VARCHAR(50) NOT NULL,  
'autonoma' VARCHAR(50) NOT NULL,  
'data_alta' VARCHAR(50) NOT NULL,  
'data_baixa' VARCHAR(50) NOT NULL,  
'motiu_baixa' VARCHAR(50) NOT NULL,  
'm_pagament' VARCHAR(50) NOT NULL,  
'banca' VARCHAR(50) NOT NULL,  
'entitat' VARCHAR(50) NOT NULL,  
'oficina' VARCHAR(50) NOT NULL,  
'control' VARCHAR(50) NOT NULL,  
'n_compte' VARCHAR(50) NOT NULL,  
'observacions' VARCHAR(500) NOT NULL,  
'categoria' VARCHAR(50) NOT NULL,  
);```

This class “soci2”=member is the main class of the system, in this class all the personal information about the member, bank data and the member number is kept. The other two classes “aspirant” and “educador” have a generalization connection with “soci2”. “soci2” has the common fields and “aspirant” and “educador” have the specific ones.

The most important fields that we can find in this class are: “id_soci” we created this field to have a primary key, then from the first member to the last member we will have a unique number that can identify every member. This field is used in all the other classes to identify the member.

We have “n_soci” that is the member number, it is surprising but not all the members have it. This is because the FileMaker database was not used when the company was established. Another important field is “nif” that is the ID of all the members like the other field not all the members have it and this is because from the beginning the company thought that it was not important know this information. “data_alta” and “data_baixa” in these fields we can find the date when the member joined and when the member canceled the membership.

**Class “aspirant” = candidate.**

```sql
CREATE TABLE `aspirant` (  
'id_aspirant' INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,  
'aspirant_id_soci' INT(11) NOT NULL ,
);```

There are two types of members in the company the “aspirants” or candidates and the “educadors” or teachers. The “aspirant” have the field “aspirant_id_soci” that is the same as
`n_curs` VARCHAR(50) NOT NULL,
`convalidables` VARCHAR(50) NOT NULL,

`id_soci`.

"id_aspirant" is the auto increment field that we created that works as a primary key.

This “aspirant” or candidate has to do a course to become a teacher and for this reason we have the field “n_curs”.

One “aspirant” can attend from 0 to N meetings (“assambleas”) and for this reason there is an association between these two classes 1 \(\rightarrow\) 0..N.

One “aspirant” has to pay from 0 to N membership fees depending on the number of years that they will be in this company, for this reason “aspirant” has an association with “quotes” 1 \(\rightarrow\) 0..N.

One “aspirant” can only do one course to become a teacher this is why there is an association 1 \(\rightarrow\) 1 with “cursos_formacio”.

And finally one “aspirant” cannot do other kinds of courses that the company offers if they are not a teacher. For this reason there are no connection between “aspirant” and “cursos_fcontinuada”.

**Class “educador” = teacher.**

```sql
CREATE TABLE `educador` (
    `id_educador` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
    `educador_id_soci` INT(11) NOT NULL ,
    `poblacio_treballa` VARCHAR(50) NOT NULL ,
    `n_curs` VARCHAR(50) NOT NULL ,
    `d_educador` VARCHAR(50) NOT NULL ,
    `convalidables` VARCHAR(50) NOT NULL ,
)
```

“educador” or teacher. In this class we should keep the city that she/he works “poblacio_treballa”, the course code that she/he did to become a teacher “n_curs” and the date that she/he become a teacher “d_educador”.

Like all the classes, we created an auto increment field “id_educador” that works like a primary key and “educador_id_soci” is the field to identify the member.

This class has an association 1 \(\rightarrow\) 0..N with “assambleas” which means that one teacher can attend from 0 to N meetings.

Like “aspirant” the teacher has to pay his/her for this reason there is a association connection with “quotes” 1 \(\rightarrow\) 0..N. A teacher can do from 0 to N “cursos_fcontinuada” other kind of courses to keep current.

The teacher can not repeat the formation course then there is not a connection between “educador” and “cursos_formacio”.
Class “cursos_formacio” = formation course.

```sql
CREATE TABLE `cursos_formacio` (
    `id_cursos_formacio` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
    `cursos_formacio_id_soci` INT( 11 ) NOT NULL ,
    `n_curs` VARCHAR( 50 ) NOT NULL ,
    `data_curs` VARCHAR( 50 ) NOT NULL ,
    `codi_curs` VARCHAR( 50 ) NOT NULL ,
    `localitat` VARCHAR( 50 ) NOT NULL ,
    `n_participants` VARCHAR( 50 ) NOT NULL ,
    `formadora` VARCHAR( 50 ) NOT NULL
)
```

Every year the company organizes different formation courses. Only the “aspirants” candidates can do this type of course because after this course you become a teacher “educador”.

In this table the course number “n_curs”, course date “data_curs”, course code “codi_curs”, location “localitat”, participants number “n_participants” and teacher’s name “formadora” is kept. Apart from that we created the auto increment field “id_cursos_formacio” because like always in the database there was not a primary key field. And the field “cursos_formacio_id_soci” is the connection with who did the formation course.

Class “cursos_fcontinuada” = other kind of formation courses.

```sql
CREATE TABLE `cursos_fcontinuada` (
    `id_cursos_fcontinuada` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
    `cursos_fcontinuada_id_soci` INT( 11 ) NOT NULL ,
    `fcontinuada` VARCHAR( 50 ) NOT NULL
)
```

For the moment there are seven different kinds of courses. In the next chapter we will add more fields in this table but for the moment we are redistributing the information and the only field is “fcontinuada” which is the name of the course. Also we created the auto increment field “id_cursos_fcontinuada” like a primary key and the cursos_fcontinuada_id_soci that identifies who did this course.

Class “assambleas” = meetings.

```sql
CREATE TABLE `assambleas` (
    `id_assamblea` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
    `assamblea_id_soci` INT( 11 ) NOT NULL ,
    `text_field` VARCHAR( 100 ) NOT NULL
)
```

All the people involved in the company can attend the meetings “assambleas” more or less every year there is one. This table is important for the company because they want to know what users are more active in the association work. We created the auto increment field “id_assamblea” and the “assamblea_id_soci” to know who attended the meetings and “text_field” which is the place where the meeting was. In the next chapter we will add more information as well.
Class “quotes” = membership fees.

```
CREATE TABLE `quotes` (  
`id_quota` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,  
`quota_id_soci` INT( 11 ) NOT NULL ,  
`codi_quota` VARCHAR( 50 ) NOT NULL ,  
`data_quota` VARCHAR( 50 ) NOT NULL ,  
`import_quota` VARCHAR( 50 ) NOT NULL  
)
```

The company was keeping this information by adding new fields every year in their high table, but this is not the best way and for this reason I created the table where they can find the membership code "codi_quota", the membership year “data_quota” and the membership cost “import_quota”. Here the auto increment field is important because one member can pay more than one fee “id_quota”, and we have to redistribute this information. And “quota_id_soci” shows who paid the fee.

After creating all these new tables the information must be moved to the correct table place. To do this task firstly the pseudo code is created and then the php script. The pseudo code is:

```
Mysql_connect()
Mysql_select_database()

var members = Select all members from table socis

while (members != NULL)
{
    foreach(member)
    {
        insert inside table socis2 this values (’, member->n_soci, member->nif, member->nom, member->cognom, member->adreca, member->codi_postal, member->poblacio, member->provincia, member->pais, member->tel1, member->tel2, member->tel3, member->fax, member->email, member->adreca2, member->professio, member->estat_laboral, member->foto_any, member->autonoma, member->data_alta, member->data_baixa, member->motiu_baixa, member->m_pagament, member->banc, member->entitat, member->oficina, member->control, member->n_compte, member->observacions, member->categoria)

        var id_soci = mysql_insert_id()// id that we will work to do the connections with the other classes

        insert inside table aspirants this values (’, var id_soci, members->n_curs, members->convalidables)
    }
}
```
insert inside table educador this values (" ", var id_soci, 
member->p_educador, member->n_curs, member->
d_educador, member->convalidables

insert inside table assambleas (" ", var id_soci, member->
assamblea)

insert inside table cursos_fcontinuadas (" ", var id_soci, 
member->f_continuada)

insert inside table cursos_formacio (" ", var id_soci, 
member->n_curs, member->data_curs, member->codi_curs, 
member->localitat_curs, member->n_participants, member->
formadora)

//var i will take this
values=[99,00,01,02,03,04,05,06,07,08,09,10] that is from
the year 1999 to 2010

from i=99 to i=10 do
{
    if(strlen(member->quota_pagada[i])>0) // if
        strlen return 0 it means that is empty, then we
do not have to introduce anything.
        {
            insert inside table quotes (" ", var id_soci, 
member-> quota_pagada[i] , member->
data__quota[i], member->import_quota[i])
        }
    }
}

It is important to do php script because since the company provided us with the
database, they have added more information. Then having an automatic script do the
migration process would make the task easier.

At this point we have the information migrated and well organized. Now it is time to
meet again with the company to add the new information that they want for the new
system.
5.4.7 Add new information
The company expects to grow quickly this year so they want to do a new design that accounts for this possibility.

Applying again the pattern “Involve the Users”, we created a new design with the company supervisor. This design is not new it only has more classes and some of the old classes have more fields.

In conclusion from the first design (Figure 5.2) we arrive to Figure14 with a migration process in the middle (Figure 5.8).

Figure 5.9: Final diagram
These are the new classes and the corresponding tables with the new information. Also a brief description about the classes is included.

**Class “proveidors” = supplier**

```sql
CREATE TABLE `proveidors` (  
    `codi_proveidor` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,  
    `nif` VARCHAR( 50 ) NOT NULL ,  
    `nom` VARCHAR( 50 ) NOT NULL ,  
    `cognoms` VARCHAR( 50 ) NOT NULL ,  
    `adreca` VARCHAR( 50 ) NOT NULL ,  
    `codi_postal` VARCHAR( 50 ) NOT NULL ,  
    `poblacio` VARCHAR( 50 ) NOT NULL ,  
    `provincia` VARCHAR( 50 ) NOT NULL ,  
    `pais` VARCHAR( 50 ) NOT NULL ,  
    `tel1` VARCHAR( 50 ) NOT NULL ,  
    `tel2` VARCHAR( 50 ) NOT NULL ,  
    `fax` VARCHAR( 50 ) NOT NULL ,  
    `email` VARCHAR( 50 ) NOT NULL ,  
    `tipus` VARCHAR( 50 ) NOT NULL ,  
    `autonoma` VARCHAR( 50 ) NOT NULL ,  
    `banc` VARCHAR( 50 ) NOT NULL ,  
    `entitat` VARCHAR( 50 ) NOT NULL ,  
    `oficina` VARCHAR( 50 ) NOT NULL ,  
    `control` VARCHAR( 50 ) NOT NULL ,  
    `n_compte` VARCHAR( 50 ) NOT NULL ,  
    `observacions` VARCHAR( 500 ) NOT NULL
)
```

This class has all the information about the suppliers from the company, like bank accounts to do the payments, name, location, contact telephones, fax etc. Every supplier provides the company with different types of products for this reason we have the class “productes”, and the connection is $1 \rightarrow N$ because 1 supplier is able to provide one product or more.

**Class “producte” = product**

```sql
CREATE TABLE `producte` (  
    `proveidor_id` INT( 11 ) NOT NULL ,  
    `codi_producte` INT( 11 ) NOT NULL ,  
    `nom` VARCHAR( 50 ) NOT NULL ,  
    `preu_unitari` INT( 11 ) NOT NULL ,  
    `observacions` VARCHAR( 500 ) NOT NULL
)
```

The class “producte” has the fields “codi_producte” which is the primary key and the field “proveidor_id” that is the field to know who supplies this specific product, as well the name and the price of the product is kept in this table.

**Class ”mitjanscomunicacio” = media**

```sql
CREATE TABLE `mitjanscomunicacio` (  
    `codi_mitja` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,  
    `nom` VARCHAR( 50 ) NOT NULL ,  
    `grup_editorial` VARCHAR( 50 ) NOT NULL ,  
    `carrec` VARCHAR( 50 ) NOT NULL ,  
    `tel` VARCHAR( 50 ) NOT NULL ,  
    `tel2` VARCHAR( 50 ) NOT NULL ,  
    `fax` VARCHAR( 50 ) NOT NULL ,  
    `email` VARCHAR( 50 ) NOT NULL ,  
    `tipus` VARCHAR( 50 ) NOT NULL ,  
    `observacions` VARCHAR( 500 ) NOT NULL
)
```

In this class the company wants to keep all the information related to the media that has collaborated with them. Such as a magazine or newspaper that was interested in their business. Each media contribution along with some articles, for this reason there is a connection between them $1 \rightarrow 0..N$ because one publication can publish articles related to the company.
**Class “colaborar” = contribute**

CREATE TABLE `colaborar` (
`codi_col` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
`mitja_codi` INT(11) NOT NULL,
`tipus` VARCHAR(50) NOT NULL,
`observacions` VARCHAR(500) NOT NULL,
`import` INT(11) NOT NULL
)

In this class the company wants to keep all the articles published in the media. In the future it may be interesting to know which publication is most active and if they have a good image of the company.

**Class “cursos_complementaria” = other courses and Class “cursos_gratuits” = free courses**

CREATE TABLE `cursos_complementaria` (
`codi_complementaria` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
`formadora1` VARCHAR(50) NOT NULL,
`formadora2` VARCHAR(50) NOT NULL,
`lloc` VARCHAR(50) NOT NULL,
`duracio` VARCHAR(50) NOT NULL,
`nom` VARCHAR(50) NOT NULL,
`data` VARCHAR(50) NOT NULL,
`preu` INT(11) NOT NULL,
`observacions` VARCHAR(500) NOT NULL
)

CREATE TABLE `cursos_gratuits` (
`codi_gratuit` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
`formadora1` VARCHAR(50) NOT NULL,
`data_curs` VARCHAR(50) NOT NULL,
`localitat` VARCHAR(50) NOT NULL,
`nom_contacte` VARCHAR(50) NOT NULL,
`tel1` VARCHAR(50) NOT NULL,
`tel2` VARCHAR(50) NOT NULL,
`email` INT(11) NOT NULL,
`data_finalitzacio` VARCHAR(50) NOT NULL,
`webcontacte` VARCHAR(50) NOT NULL
)

If the company grows they will offer two other types of courses. “cursos_complementaria” and “cursos gratuits”, the first one all the members can do without any restriction and the second type only the families can do for free.

In these classes the company wants to keep for example the name of the teacher that will do the course, where will be, duration, price etc..
Class "familia", “membres”, “colaborador” = family, members, collaborator

CREATE TABLE `familia` ( 'codi_familia' INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY , 'codi_donacio' INT( 11 ) NOT NULL , 'codi_gratuit' INT( 11 ) NOT NULL , 'codi_id_soci' INT( 11 ) NOT NULL )

CREATE TABLE `membres` ( 'codi_membre' INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY , 'codi_donacio' INT( 11 ) NOT NULL , 'codi_complementaria' INT( 11 ) NOT NULL , 'codi_id_soci' INT( 11 ) NOT NULL )

CREATE TABLE `colaborador` ( 'codi_colaborador' INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY , 'codi_donacio' INT( 11 ) NOT NULL , 'codi_quota' INT( 11 ) NOT NULL , 'codi_id_soci' INT( 11 ) NOT NULL )

These three new classes are new types of members, but they are a generalization of the class “socis” for this reason all the common fields are in this class.

For example if you are a family you can only do donations and attend the free courses but if you are member you can also attend to other courses and if is a “colaborador” you can not attend any type of courses. This type of member is only for people that want to continue linked with the association without active participation.

Class “donacions” and “atendrefamilies” = donations and family attended

CREATE TABLE `donacions` ( 'codi_donacio' INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY , 'donacions_id_soci' INT( 11 ) NOT NULL , `programa` VARCHAR( 100 ) NOT NULL , 'observacions' VARCHAR( 500 ) NOT NULL , 'import' INT( 11 ) NOT NULL )

CREATE TABLE `atendrefamilies` ( 'codi_familia' INT( 11 ) NOT NULL , 'families_id_soci' INT( 11 ) NOT NULL )

Of these two last classes the first one “donacions” donations, is the most important for the company. All the people linked with this company can make a donation and they want to keep all the necessary information to know who is the most active member and who gives the most money.

And the class “atendrefamilies” is a log of all the families that the company has helped.

Apart from these new classes the company wants to add the following new fields within the existing classes.

ALTER TABLE `cursos_formacio` ADD `nom_contacte` VARCHAR(50) NOT NULL , ADD `tel1` VARCHAR(50) NOT NULL , ADD `email` VARCHAR(50) NOT NULL , ADD `webcontacte` VARCHAR(50) NOT NULL , ADD `curs_cat` VARCHAR(50) NOT NULL , ADD `curs_illes` VARCHAR(50) NOT NULL , ADD `preu` INT(11) NOT NULL ,

ALTER TABLE `cursos_fcontinuada` ADD `formadora1` VARCHAR(50) NOT NULL , ADD `formadora2` VARCHAR(50) NOT NULL , ADD `tel2` VARCHAR(50) NOT NULL , ADD `lloc` VARCHAR(50) NOT NULL , ADD `duracio` VARCHAR(50) NOT NULL , ADD `nom` VARCHAR(50) NOT NULL ,

Inside the classes “cursos_formacio”, “cursos_fcontinuada” and “assambleas” the company wants to add these new fields that provide general information which they did not have before.

From prices, teacher’s names, location fields etc.
<table>
<thead>
<tr>
<th>SQL Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD <code>preu</code> INT( 11 ) NOT NULL,</td>
</tr>
<tr>
<td>ALTER TABLE <code>assambleas</code></td>
</tr>
<tr>
<td>ADD <code>data</code> VARCHAR( 50 ) NOT NULL ,</td>
</tr>
<tr>
<td>ADD <code>adreca</code> VARCHAR( 50 ) NOT NULL ,</td>
</tr>
<tr>
<td>ADD <code>nom_perso_contacte</code> VARCHAR( 50 )</td>
</tr>
<tr>
<td>ADD <code>telefon</code> VARCHAR( 50 ) NOT NULL ,</td>
</tr>
<tr>
<td>ADD <code>localitat</code> VARCHAR( 50 ) NOT NULL ,</td>
</tr>
<tr>
<td>ADD <code>provincia</code> VARCHAR( 50 ) NOT NULL ,</td>
</tr>
<tr>
<td>ADD <code>c_autonoma</code> VARCHAR( 50 ) NOT NULL ,</td>
</tr>
</tbody>
</table>
6. Conclusions and Future Work

In this chapter, we will present the conclusions that we have extracted from the project development and the future tasks that have to be done to correctly accomplish all the company requirements.

6.1 Conclusion

Problems such as the one the AEMI company faced could be looked at from an ad hoc approach, but this individual assessment faces several problems. For instance, using general patterns allows for example multiple people to understand and solve the same problem, keep an order, following the same style, etc.

FileMaker is versatile and easy to use but when the company began to need functionality, scalability and replication it is better to change the platform. MySQL is an excellent alternative due to its adaptability and scalability. By using the migration process that we described in this project and using the adequate “patterns” the new database developers can use the best database for their growth needs.

The aim of this thesis was to solve the specific platform migration problem using general “patterns”. During the project development the following goals had been reached:

The first goal was to assess the applicability and usefulness of the patterns to solve the specific problem of the platform migration. Once we determined and adapted the best patterns for our project we applied these patterns as a guide. Then in Figure 5.7 we can see that these patterns worked because all the project information followed a guideline and the information was correctly migrated. Thus this goal was achieved.

The second goal was do a redesign with the migrated database. After completing the migration process we had to improve the organization of this information, because in the future they wanted to add more data, which had been impossible to do in FileMaker. Then in Figure 5.8 we can see the redesign and applying the algorithm in page25 we can see that the information reorganization was a success.

The third goal was to add the new information that the company provided us inside the migrated system. After the redesign process we added the new information (Figure 5.9). The final result and the final database that we provided to the company is shown in Figure 6.1.

![Database field’s names](image-url)
In Figure 6.1 we can see all the tables created and if for example we choose the table “cursos_fcontinuada” = formation courses, then in Figure 6.2 we can see that all the migration fields are filled in and the empty fields are the new information added and ready to be filled in.

![Figure 6.2: “cursos_fcontinuada” table information](image)

### 6.2 Future Work

There is a lot of work after all of this. The first task would be to find more examples of general patterns that can be used in migration projects. For example some of the patterns that we used are general patterns that can be applied to all types of projects, but we modified them for the purpose of our project.

In addition, we finished the migration process with all the information in mySQL inside the company server. Now a new project has to be done to manipulate this information. In order for the information to be accessed through the internet by all the members, a lot of new functionalities must be produced. Such as a sign up for the company courses, print certificates, print membership cards, pay the fees, etc… In order for all of that to be possible the database will need a small redesign and a lot of work hours.
7. Bibliography


