DESIGN OF A WEB BASED INTERFACE AND DATABASE STRUCTURE FOR STRUCTURAL ENGINEERING EXPERIMENTAL DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TOLGA KURT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

SEPTEMBER 2014

Approval of the thesis:

DESIGN OF A WEB BASED INTERFACE AND DATABASE STRUCTURE FOR STRUCTURAL ENGINEERING EXPERIMENTAL DATA

submitted by **TOLGA KURT** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Ahmet Cevdet Yalçıner _____
Head of Department, **Civil Engineering**

Assoc. Prof. Dr. Özgür Kurç _____
Supervisor, **Civil Engineering Department, METU**

Examining Committee Members:

Prof. Dr. Barış Binici
Civil Engineering Department, METU _____

Assoc. Prof. Dr. Özgür Kurç
Civil Engineering Department, METU _____

Assoc. Prof. Dr. Murat Altuğ Erberik
Civil Engineering Department, METU _____

Assist. Prof. Dr. Mustafa Tolga Yılmaz
Department of Engineering Sciences, METU _____

Assist. Prof. Dr. Onur Pekcan
Civil Engineering Department, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: TOLGA KURT

Signature :

ABSTRACT

DESIGN OF A WEB BASED INTERFACE AND DATABASE STRUCTURE FOR STRUCTURAL ENGINEERING EXPERIMENTAL DATA

Kurt, Tolga

M.S., Department of Civil Engineering

Supervisor : Assoc. Prof. Dr. Özgür Kurç

September 2014, 80 pages

This study is conducted in order to design and implement a web-based database application which stores structural and earthquake engineering experimental data and related documents in a main server in a standardized manner and makes it possible for researchers and engineers to add and share experimental data easily. The system is developed in accordance with the standards defined in the scope of the SERIES project which is a project operated by a consortium consisting of related university laboratories and institutions of European countries. The implementation was first tested in Structural and Earthquake Engineering Laboratory of Civil Engineering Department, METU and then has become a part of the SERIES network. The application has two main components one of which is used to store information about projects, specimens, experiments, related files, sensor and loading devices at the server side. The other component of the application is a website allowing users to attain a functional user interface which makes it easy to use the application, and it is accessible from everywhere

with internet connection. After the database was created using MySQL database management system, it was attached to the website which was developed using HTML, CSS, JavaScript and PHP programming languages. The system provides functionalities to create, modify, search, and delete projects, specimens and experiments, to upload/download related files, to plot signals and other related features. The application is thoroughly tested with the experiment data obtained from previous projects conducted in METU and is accessible via a public URL in production mode.

Keywords: Structural and Earthquake Engineering Experimental Database, Database Design, Internet Application for Structural Engineering, Online Experiment Data

ÖZ

YAPI MÜHENDİSLİĞİ DENEY VERİLERİ İÇİN BİR WEB ARAYÜZÜ VE VERİTABANI YAPISI TASARIMI

Kurt, Tolga
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Özgür Kurç

Eylül 2014 , 80 sayfa

Bu çalışma, yapı ve deprem mühendisliği deneylerine ait verilerin ve ilişkili belgelerin standart bir yapıda depolanması ve araştırmacılar ile mühendislerin deney verilerini kolaylıkla ulaşıp kullanabilmesi için tasarlanmış ve uygulamaya konulmuş, web tabanlı bir veritabanı yazılımı hakkındadır. Çalışma kapsamında geliştirilmiş olan sistem, Avrupa'daki ilgili üniversite laboratuarları ve enstitülerden oluşan bir konsorsiyum tarafından yürütülen SERIES projesinde tanımlanmış standartlarla uyumludur. Geliştirilen uygulama, ilk olarak ODTÜ İnşaat Mühendisliği Bölümüne bağlı Yapı ve Deprem Mühendisliği Laboratuvarı'nda denenmiş olup, akabinde SERIES dağıtık veritabanının bir parçası olarak kullanılmaya başlanmıştır. Sistemi oluşturan iki temel kısımdan birincisi projeleri, deneyleri, numuneleri, yükleme ve ölçüm cihazlarını ve ilişkili dosyaları sunucu tarafında depolayan veritabanı uygulamasıdır. İkinci kısım ise, bu veritabanının tüm internet üzerinden erişilebilir olmasını sağlayan ve kullanıcılar için kolay bir

veritabanı arayüzü olarak işlev gören internet sitesidir. Veritabanı bileşeni MySQL veritabanı yönetim sistemi ile programlanmış ve sonrasında HTML, CSS, JavaScript ve PHP programlama dilleri ile oluşturulmuş olan internet sitesine entegre edilmiştir. Sistem genel olarak, kullanıcıların projeler, deneyler, numuneler ve diğer ilişkili varlıkları yaratmasını, düzenlemesini, silmesini ve bunlar içinde arama yapmasını, bu varlıklara ilişkili dosyaları sisteme yükleyebilmesini ve daha önce yüklenmiş olan dosyaları indirebilmesini, deney datalarını grafik olarak çizdirebilmesini sağlamaktadır. Uygulama, ODTÜ'de daha önce yürütülen proje verileriyle denenmiş olup, genel bir URL ile üretim ortamında erişilebilir haldedir.

Anahtar Kelimeler: Yapı ve Deprem Mühendisliği için Deney Veritabanı, Veritabanı Tasarımı, Yapı Mühendisliği için İnternet Uygulaması, Çevrimiçi Deney Datası

To my family

Adnan Kurt, Beyhan Kurt, Tuğba Kurt Tanır, Filiz Fidan

ACKNOWLEDGMENTS

I was able to complete this study only with the guidance and encouragement of my supervisor, Dr. Özgür Kurç. I cannnot thank him enough. It was a great honor to work with him.

And Filiz Fidan, my dearest one. It is hard to adequately express my gratitude.

Last, but not the least, I would like to sincerely thank to my parents and sister. They were always there in my hour of need.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Information sharing is a significant accelerator in the development of approaches and methodologies of engineering as a result of improvements in computer science and Internet technologies. Since the formulations and assumptions that are related to the field of structural and earthquake engineering are generally dependent on the experimental results, test data is the main basis of the developments in these fields. Therefore, conducting experiments and sharing experimental data among researches are very crucial in the development of structural and earthquake engineering knowledge in general.

Conducting experiments within the fields of structural and earthquake engineering are generally very time consuming processes and installing experimental setups requires high expenditures. This situation limits the variety and the amount of experiments. Thus, every experiment is of utmost importance and sharing the results of these experiments is a significant part of the process. Unfortunately, due to the lack of a common platform which allows researchers to share and access information, some of the test data get lost in time. In addition to that, since the experimental data are stored on unstandardized mediums, they become incomprehensible after a period of time.

Considering these problems mentioned above, in order to benefit from the experiments with maximum output while saving time and cost, structural and earthquake engineers should avoid repeating experiments unnecessarily. With

the help of a standardized storage medium and a common sharing platform, test data can be accessible by researchers worldwide and researchers can easily access to information needed. Moreover, data stored in this way do not become incomprehensible in time.

It is quite possible and easy to develop storage and sharing platforms as such owing to the current status of database systems and internet technologies. In this study, a web based database management system developed for this purpose is explained in detail. The system design has been developed and put into work within the Structural and Earthquake Laboratory of the Civil Engineering Department of METU. By means of this system, experimental data can be stored in a standardized, consistent and reliable way, as well as it can easily be managed and served owing to the web interface. Following the development procedure, the system was tested by entering the test data of the previous researches conducted within the laboratory, the bugs were fixed and the system has been made ready to put in production environment.

1.2 Review of Previous Work

Over the course of many years, database systems belonging to the different branches of civil engineering have been used in order to store experimental data. However, the majority of the databases were not accessible through the Internet or accessible only with the help of some front-end applications that cannot be used without a special operating system (OS). Thus, it was not possible to access most of the data and it was not possible for everyone to access the available data. Recently, due to the rapid development in web technologies, database systems are now connected to each other via World Wide Web without no additional software besides an internet browser (which is at least almost always bundled with the OS), in order to provide accessibility for more people. Hereby, people are now free of the constraints of operating systems as well as the hassle of installation and setup of specific client side applications and can have access to stored data using any device with an Internet connection.

In this section, some web based experimental database systems are presented. Most of these databases have been developed to store experimental data about the earthquake performances of various structures. Expanding data pool has been examined statistically and made use of to fix the formulas used in order to solve seismic problems. Other presented web-based database studies are on finding out the best way to manage storm water and storing field experiment data and the locations of them in a database. These geographical databases provide efficiency of time and cost reduction for future projects. In addition, there exist real time database management systems, for checking the sensor readings and the situation of the structure, which are accessible from remote computers with Internet connection.

NEEScentral, one of the most extensive web-based test data databases, was a national web-based study performed by NEES Consortium, Inc. in order to contribute to the accumulation of knowledge about theoretical and experimental earthquake engineering. With the support of the National Science Foundation, NEEScentral has been developed for communication between NEES members and the database. By the use of the portal, the contributors can be informed of recent researches, new simulation models and industry practices. The extensive database of the NEEScentral enables re-conducting a number of experiments and is fully controlled by users. Moreover, there are a great number of experiment data and a number of users managing the database in NEEScentral. In order to sustain the order, it keeps the files that are related to tests and projects directly to the file system in which other data is all stored in the DBMS. [15]

Another web-based database is the PEERspd, the goal of which is to support studies on the models of seismic performance for reinforced concrete columns. This database has been built upon the database that the National Institute of Standards and Technology had been using previously and further developed in the University of Washington. There were 107 rectangular and 92 spiral reinforced concrete column tests stored in the previous database. As soon as the PEERspd was introduced, these numbers increased to 274 for rectangular and 160 for spiral reinforced columns. In addition, in order to obtain an interactive interface, a web-based user interface was started to be used. The initial

NIST database contained digital top force-displacement history, key material properties, and the definition of test geometry. However, today it can involve load-displacement configuration, maximum damage deflection occurred before various damage levels, axial load information, column reinforcement details, key images and drawings (if there is), comments (e.g. extraordinary features), references and links to access more information. [17]

Another project is NGA. This study has been conducted under the scope of the PEER Strong Motion Database, a corporate project of the South California Earthquake Center and Pacific Earthquake Engineering Research Center-Lifelines Program. This project contains 173 earthquakes, 1400 seismic observation center and 3500 recordings with various components. All of the information is accessible through the web-based graphical user interface. [16]

In Japan, where seismic activity risk is very high, a number of experiments have been conducted on the structural steel components to identify the seismic performances of steel structures during a number of pseudo-dynamic and cyclic load tests. Numerical Database for Steel Structures was created in Japan, with the purpose of storing and publishing the details and results of experiments. NDSS is a shared and a distributed database system and it is possible to store and process seismic experiments (pseudo-dynamic and cyclic load tests) and ultimate strength tests with the help of it. The outputs can be in formats of alpha-numerical data, video or image. Access to the primary metadata server and the databases running in distinct servers is possible via the Internet. The system has several tasks such as conducting quantitative analysis on main framework and transferring the results to other nodes on the network. The distributed database server utilized in this study proved to have many benefits compared to a concentrated server. For instance, no performance bottlenecks occurred on any nodes of the distributed system or in the network, every user has full control over their own study while determining their own the extent of the data they will receive as well as whether there is a need for any data change, and by the use of their own servers, privileged users can modify the database. [10]

Another online database focusing on geotechnical info sharing is the Visual Data

Center of the COSMOS center. COSMOS VDC is known to be a seismic activity research engine that is extensive, unrestricted, online and interactive and was developed for engineers, seismologists and others working on the earthquake field. Strong motion earthquake data from a wide variety of resources in increasing numbers can be accessed easily via the system. The data can be downloaded as unformatted files storing unprocessed acceleration records, modified acceleration records, displacement, velocity, and spectra of response. The table of metadata variables such as S-wave speed for peak acceleration and stations reporting a specific earthquake can also be downloaded. VDC has more than one interface which allows users to choose data with respect to earthquake observatory. The response signals in the files can be monitored before download. VDC is restricted by earthquakes with the intensity of 5.0 in the areas with high earthquake possibility and 4.5 and larger in the areas with lower seismic activities. There are hundreds of earthquakes, thousands of stations and researchable metadata of acceleration records in the database and their numbers have been increasing constantly. [4, 3]

Federal University of Pernambuco performs researches in geotechnical laboratories and field. They created a geotechnical database using information technology tools and field test outputs from two distinct research areas. The essential goals of the database can be listed as providing a more effective and faster method for analyzing statistically the geotechnical parameters that are obtained from advanced researches and in situ tests, providing details about the equipment used and field experiments conducted by UFPE, helping the use of other field experiments in Brazil to spread, making the empirical evaluation/calibration in the literature easier and more effective to use working with soft clay in Receife, becoming a pedagogical tool to support undergraduate and graduate lectures in geotechnical engineering. The database is accessible via a certain application or WWW.[5]

There are RAM-based storages along with traditional databases. The National Institute of Standards and Technology is researching new health monitoring techniques for buildings in order to develop sensor technologies that will provide energy conservation and comfort for residents, increase the number of accessible

data in building control and health monitoring systems, providing new tools to detect and identify the problems in the buildings, and provide security for the building residents by promoting the interoperability of data from various sources. One of the existing focus areas is the wireless sensor networks. By using wireless technology, data transfer throughout the building is possible with placing sensors. Building Environment Division of NIST developed a remotely controlled sensor grid system in order to gather data from the sensors that were placed in the building, store them in the database and then make it possible to monitor the building over the web. [14]

1.3 Purpose and Scope

The main purpose of this study is to develop a platform to store and share experimental data within the body of the METU Structural and Earthquake Laboratory. The main constraint of this study is to develop a system in accordance with the restrictions that were identified within the SERIES project, a distributed database structure, which was carried out to bring the data, acquired in related laboratories, the majority of which are located in Europe centered universities, in common use. [21] Within this constraint, the primary purpose of the platform is to store and present the experimental data acquired within the laboratory along with every kind of information, image, video and documentation related to the experiments. Moreover, it is necessary to prevent the experiment results to become incomprehensible in time, by exhibiting them with the information of the projects to which they belong, every type of materials, structural components and specimen, sensor and loading devices used within these projects. In addition to these, it is also aimed to develop a web based interface possessing the necessary features that allow both the administrators and the end users to easily use and access to the system. Finally, it is necessary to develop the means providing data transfer from the database developed within the scope of this project to the common interface developed within the scope of the SERIES project.

The platform that has been developed within the scope of the project basically

consists of a database and a web-based interface. The database has been implemented using MySQL Database Management System. The interface developed for access to this database and to manage the content has been programmed using HTML, CSS, JavaScript and PHP languages and means developed using them. The system is running on a server with the Debian operating system, which is run within the body of the laboratory.

1.4 Overview of the Study

In the second chapter, preliminary information is given to ensure the comprehensibility of the study. Within this context, first the detailed information about the databases and the database management systems are covered. What an Entity-Relationship diagram and how such a diagram works are explained at this point. In addition, the general structure of a web application, the layers of a web application and how these layers should be designed in order for them to be flexible and expandable is discussed.

The focus of the third chapter is the database which was specially designed for the study. All of the entities constituting the database and how the relations between these entities are established is explained giving the reasons and detailed information about the tables created as a result is provided.

In the fourth chapter, the web application which has been developed in order to access to the database and manage its content is explained and the features that can be practiced using the web application are touched upon. Additionally, the design philosophy and development specifics are explained in order to provide a legacy documentation for possible further developments.

The fifth chapter focuses on the achievements gained through the study. Moreover, the constraints and the limitations of the study and possible solutions are discussed. Finally, suggestions for further development are given.

CHAPTER 2

ESSENTIAL INFORMATION

2.1 Introduction

In this chapter some detailed information that is necessary to understand the graphical user interface and the database, which has been developed within the scope of the study is given. First of all, the concept of database, types of databases and database management systems are explained in detail. Then the Entity-Relationship diagrams that are used to state the structure of a web base is touched upon. Finally, the typical structure of a web application developed in order to create a graphical interface and the relation it establishes with the database is explained.

2.2 Databases

2.2.1 Overview

Databases and database technologies have a very significant role in almost every field where computers are widely used such as business, e-commerce, engineering, medicine, law and education. A database can simply be defined as collecting the data that are meaningful and recordable. However, the general definition of a database faces some restrictions since it cannot represent every aspect of the real world. In addition, a database should be designed, constructed and filled for a special purpose and those who are actively interested in the content. [22]

2.2.2 Database Management Systems

A database management system is the collection of the software applications which allows users to construct and sustain a database. The initial purpose of the DBMS is to enable users to save and access efficient and suitable database. By using DBMS, users can store data in the database according to their types, structure and contents and he database is accessible by more than one user or program at the same time. Also the data is protected against hardware/software errors and unauthorized or ill-intentioned access and the data is kept secured and maintained through a life cycle. [22, 8]

PostgreSQL and MySQL are widely used DBMS solutions and free; while typical commercial DBMSs can be listed as Oracle, Sybase Adaptive Server Enterprise, Microsoft SQL Server. These DBMSs can connect to some desktop and web applications via internet or intranet.

Database systems differ from typical file storage systems in terms of many additional features. The typical file storage system is supported by traditional operating systems. The content is stored in permanent records in various files and required other applications to extract or add records. Prior to DBMS, organizations used to store data in such systems. However, there are some difficulties to store information in an organized manner in such systems in terms of satisfying redundancy, consistency, integrity, security and management of data. Moreover, there occur some severe problems when isolating or accessing data directly or simultaneously. Such difficulties triggered the development of modern database management systems. [18]

A basic feature of a database defined within a DBMS is that the system includes both the database and outright definition of the structure and limitations of the database. The definition is kept in the DBMS catalog where file structures, type and format of data items, and numerous limitations on data are stored. The metadata catalog stores the information which identifies the organization of the preceding database. The other characteristics of DBMS can be listed as storage, retrieving and updating data, controlling data entrance, allowing simultaneous

access by multiple clients, enabling recovery and maintenance. The workflow and the components of a DBMS can be seen in Figure 2.1. [22, 23]

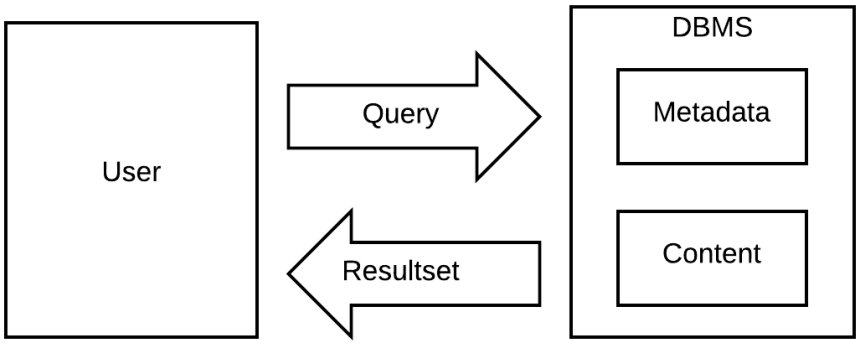

Figure 2.1: Workflow and components of a typical DBMS.

In order to grasp the structure of the database, the basic concepts like entity, property and relationship should be known. An entity is defined as an "object" or a "thing" that can be distinguished from other objects and it is shown in a database. For example, a document, a project, web site members and groups of them are entities. Entities are identified in a database by a bundle of properties which are the attributes that the entity possesses. For example, a project entity possesses a project name property while the name of a file is a property of a document entity and user name and password are properties of member entities. A relationship, on the other hand, is the corporation between two or more entities. Entities are connected to one another by their properties. [8] For instance, the relationship between a document and a project can be established over the "related project ID" property of the document.

During the creation of a database, the concepts get real-world implications and the tables represent the entities in the database and the columns of the tables are formed by the properties. The data or information is inserted as rows in these tables. Every individual row is a combination of the properties that defines the related entity. When it comes to identifying entities, usually, one or two properties are enough to identify an entity uniquely. So that, it is a common (or compulsory in some DBMSs) practice to give an extra property which is

called the primary key (PK) of the related entity. The relationships between entities are conducted over primary keys. Foreign key is the attribute that indicates the primary key belonging to another table. Primary and secondary keys provide data consistency of the database. For example, a common feature of many DBMSs is to avoid the deletion of an entity whose PK is defined as FK of another entity, since after the deletion of such an entity makes the other entity undefined in some way.

Shortly, a Database Management System (DBMS) can be considered as a mediator between physical storage, computer, operating system and the user. DBMS provides a customized software language named database language in order to accomplish a variety of tasks. Data Description Language and Data Management Language are major database languages. Data Description Language basically executes these functions: creation of tables, file databases and data dictionaries, customization of storage design on each table in the disk, assignation of the integrity limitations in a numerous number of tables, adjustment of security and authorization info of every individual table, specification of the structure of every table and sustaining the whole structure of the database. Data Management Language, on the other hand, provides users such authorizations as saving, adding, and managing, removing, accessing and updating data. [11]

DBMSs can be divided into three groups with respect to their data storage and management model, the number of computers among which the data is distributed and the data storage platform.

In terms of their management and storage system, it is possible to categorize DBMSs in four classes. First one of them is the Hierarchical Model is the type that keeps relations in a binary structure. Thus, the objects are created by one-to-many method. Then comes the Network Model which can be considered as the extended version of the previous model. If the one-to-many relation is violated by using many-to-many, this model is named Network Model. Third and the dominant model is the Relational Model in which the "Tables" stores data and relationships are established between tables via entries of these tables. Lastly there is the Object Oriented Data Model which is expected to be the

future version of management systems. In these systems, data is stored by individual data entries, and using encapsulated data and operations. [8, 11, 19]

In terms of the number of the computers over which the database is run, DBMSs divide into two groups. First group contains the centralized DBMSs which operate on a single computer. The only way for all of the data storage and database management to be carried out is to make use of a computer. Second one is the set of distributed DBMSs which run on different computers that are linked to each other over a network. Every individual computer has its own database and management system; however, one can have access to the whole database using only one computer because every computer works together in a common network. [18, 19]

According to the data storage platform, DBMSs can be categorized into two. Former group is the traditional disk based databases are the most common types of databases which prevails consistency and reliability over performance. Latter one is the RAM based databases which are relatively a new technology which is said to be real-time due its enormously fast data access and manipulation time. [20]

Modern databases need wide storage areas. Therefore, ferromagnetic disks are very suitable for storing data since they do not cost much. Most of the databases today are on disks and they satisfy the user needs with an easy to use management system and an interface; however, the mechanical structure of the disk limits its speed to a certain extent. Although the disk based databases have enough speed for most of the applications consisting human users, they are not enough for the applications that are controlled real-time. Access time of a disk based database can be developed by usage of RAMs and a flash storage. However, this method still do not provide sufficient amounts of speed especially for the databases with intensive data operations. [2]

A RAM-based storage has higher speeds than of a disk-based storage for the individual read and write operations. However, RAM based storage is considerably more expensive than a disk based storage. Thus, RAM applications should be used only when the data to be stored is in limited amounts. These

special databases require all data to be stored in RAM if high performance is needed. Real-time data management systems are developed in order to achieve this purpose. [2, 7]

Real-time database systems should come up with convenient results for many simple queries (commonly via a primary or secondary key) and be able to administer simple data by incessant updates. Moreover, they are expected to be available almost always; its downtime should not pass thirty seconds in a year and have really short response duration which should be around a few milliseconds. [7]

After investigating all of these possibilities, the study was decided to be performed on a disk-based centralized relational database management system at the beginning. Since it is the type that is used widely and easy to construct this model is chosen. Since the database will only be used in METU firstly, it is not necessary to have a distributed database. In addition, today there are no real-time readings obtained from sensor networks to necessitate a real time database implementation. Thus, data is stored in ferromagnetic disks instead of RAM. DBMS may be extended according to needs and become the database that is distributed real-time, in the future.

2.2.3 Entity-Relationship Diagrams

Entity-Relation Diagrams (ERD) are schematic notations of the entities, properties and relationships between them. ERD is used for data modelling and can be regarded as a blueprint of the database. ERD represents all of the entities, their properties and the relations between them.

Today, many ERD notations exist such as IDEF1X, Chen's notation, Bachman notation, Crow's Foot notation, minimum-maximum and UML notations. [12] How they construct the entities, modality and the cardinality of a relation determine what differentiate these notations. Cardinality means the maximum numbers of times a property in an entity refers to that of another entity. Modality, on the other hand, means the minimum number of that. Cardinality, the

symbol of which is put on the outer ends of the relationship line, and it can have the values 1 or many. Modality, on the other hand, can be 1 or 0 and the symbol is put inside, besides the cardinality symbol. In order to draw the ERD of the database, Crow's Foot model is used in this thesis. A sample entity is shown in Figure 2.2. In addition, modality and cardinality of 1 is plotted as a straight line while a circle represents 0, and many is plotted as a foot with three toes in Crow's Foot notation. Both ends of the relationship line represents modality and cardinality. A sample connection between two entities is depicted in Figure 2.3. and all modes of modalities and cardinalities are shown in Figure 2.4.

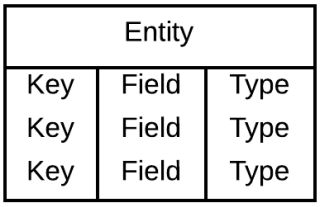

Figure 2.2: A sample entity in ERD standards.

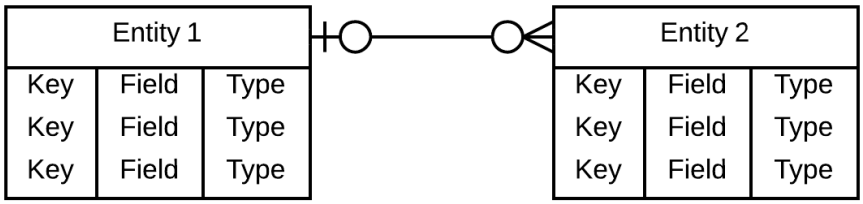

Figure 2.3: A sample connection between two entities in an ERD.

Along with the cardinality and modality relationships of entities, there are two relationship types indicated in the ER diagrams. The relationship which is indicated as a beeline in ERD is named Identifying Relationship while the non-Identifying Relationship is shown as a dotted line in ERD. Identifying relationships refer to child tables which are not possible to identify without the parents and is usually observed in many to many relationships. Creating a new

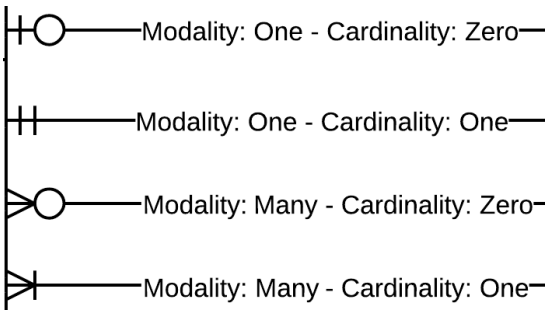

Figure 2.4: All modes of modalities and cardinalities in ERD.

intersection table which contains the primary key properties of the entities in many-to-many relationship helps reduce the data repetition. The existence of the data which is entered into the intersection table depends on being identified in the parent table. This is the reason why man-to-many relationships are identifying relationships. If the data that exist both in the parent and child tables can be identified although they do not exist in the other table, this relationship is called non-identifying relationship. [12]

The entities are shown as three-columned boxes in the ER diagrams that are presented in this study, and their names are typed on the top of the boxes. They are used to indicate the many-to-many relationship between the entities and they prevent redundant data recurrence. The entities are shown as tables in the database. The columns of the table indicate the properties of a connection table or an entity. In the ERDs, the list of the properties of primary key and foreign is present inside the boxes that are located under each table name. A property possesses some features that indicate whether the value of the property is decided by the user or the database, whether the property is a foreign key or a primary key, whether the value of the property should be filled or not. In addition, ERDs feature the value types of all properties.

The PK abbreviation located in front of the property name indicates that that property is a primary key for the entity. One property is, in fact, enough for determining a tuple for entities; however, two properties for two connection tables form the primary key and four properties are needed in order to create a primary key for one connection table. Foreign key properties are marked by

using a red diamond symbol and provide information consistency of the data in the database. The connections between the entities are provided by a primary key on another table and the related foreign key of the table. The relationships between the tables are indicated by Crow's Foot notation.

The value type of a property is defined on the right side of the property name. Nine value types are used in the present database system, which are CHAR(n), LONGTEXT, VARCHAR(n), DOUBLE, INT(n), TINYINT(1), LONGBLOB, DATETIME, and TIMESTAMP. Char, longtext and varchar are text-based properties and the characters inside the parenthesis are the character number limit. Double and int indicates that the property is quantitative and whether the number contains a decimal part or not. In order to indicate whether the property is a Boolean value, Tinyint (1) is used. 1 or 0 is stored in the database depending on the truth value of the value. Timestamp and datetime are used in order to store time in the database. Finally, to identify the binary data in the database longblob is used. On the right side of some properties there written UN, NN and Al. UN, which is used for integer value types, means unsigned. The capacity of a UN can be doubled; however, entering a negative value for this property is impossible. NN means not null and that means when the data is entered into the database, the value field cannot be left unfilled. Al property is "auto increment" which indicates that the symbol value is increased automatically every time a data is entered. This field cannot be filled manually. Auto increment properties perfectly fit to be a primary key since a new unique value is automatically entered for auto increment fields whenever a data is entered.

2.3 Internet and Web Applications

2.3.1 Overview

A web site is a collection of the text based files named as web pages that are linked to each other and the content of them can be displayed via HTML viewing software. Internet resources are created by Hypertext Markup Language (HTML) that is used to generate documents, link to other resources which

can include images, audio objects or videos. In addition, it can include software scripts named "commands" written using various software languages like JavaScript. All these various types of information are provided by the standard HTML elements composed of "tags," specific keywords used to mark the content in the web-page. All of the public web pages constitute the World Wide Web, which is also known as WWW, W3, or generally, Web.

2.3.2 Structure of a Web Application

To access a resource on the internet, users make requests via their browsers formatted as URLs. These URLs include the hostname of the server, the name of the resource and the dynamic content modifier variables. Upon transmitting this request to the relevant web server software, the content that user requested is prepared and sent back. Hypertext Transfer Protocol (HTTP) provides access to and transfer of HTML pages. HTTP offers a standard format to determine the requests of resources on the web and how servers and browsers react to various commands. [13]

When the first WWW was created in 1990s, the Web was basically consisting of single servers hosting simple and static HTML pages. However, it is now a platform with various domains and helps both deliver information and run applications. The HTML format which defines the display of the web page basically stands for the application's surface. Users are in interaction with the HTML markup while the application is operating on a distant web server. Depending on the circumstances, there might be more than one remote server on which the application runs. [9]

2.3.3 Layered Design

Data layer, application layer and presentation layer are development layers of a web application. The developer should decide at the data layer where and how to store the data of the application that is being developed, the data format and database management systems as well as the construction of these systems and

if external data sources will be used. At the application layer, on the other hand, protocols, markup and programming languages and application architecture that will be used should be determined. The navigation format between the web pages of the application is also described at this layer. If third person resources or remote service are needed, their integration is processed at this layer. Finally, the developer should focus on the application's user like HTML templates and layout of the application at the presentation level. The decisions should be made user-oriented, with special concern to the appeal, utility and accessibility, to ensure the satisfaction of users. [25] Generalized schematic view of layered web application design is exhibited in Figure 2.5.

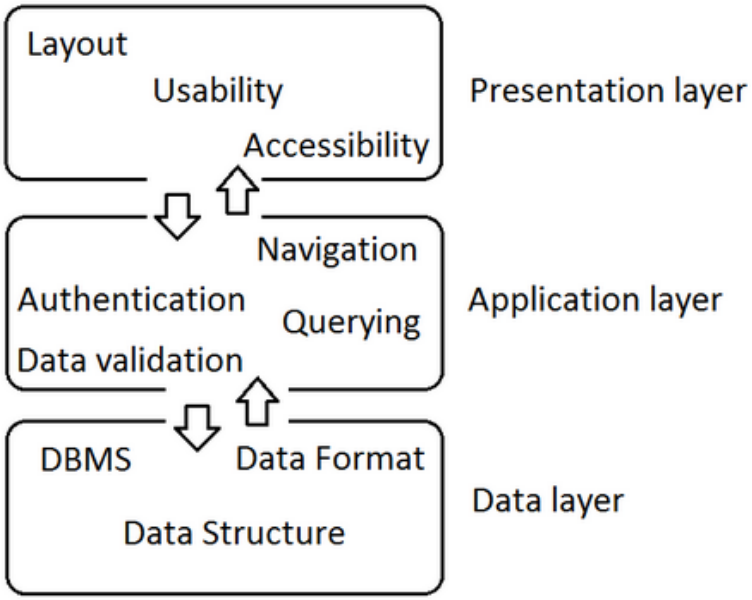

Figure 2.5: Schematic view of layered web application design.

CHAPTER 3

THE DATABASE

3.1 Introduction

This chapter explains the process of design and implementation of the database. All main tables and their properties are discussed in detail as well as their relationships.

3.2 Overall Database Structure

It is possible to say that the database consists of three parts in general. These parts with regards to each other can be seen in Figure 3.1.

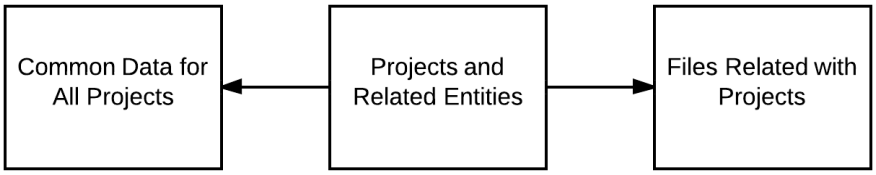

Figure 3.1: Three main parts of database.

The first one of them is the projects and their related entities. These entities are the experiments, specimens, structural components, materials, loading and sensor signals and the configurations producing these signals. All of these entities were configured in the context of the projects. A rough sketch of the data model

of this part is shown in Figure 3.2. Detailed explanation of the entities and relationships which are shown in Figure 3.2. are given in following sections of this chapter.

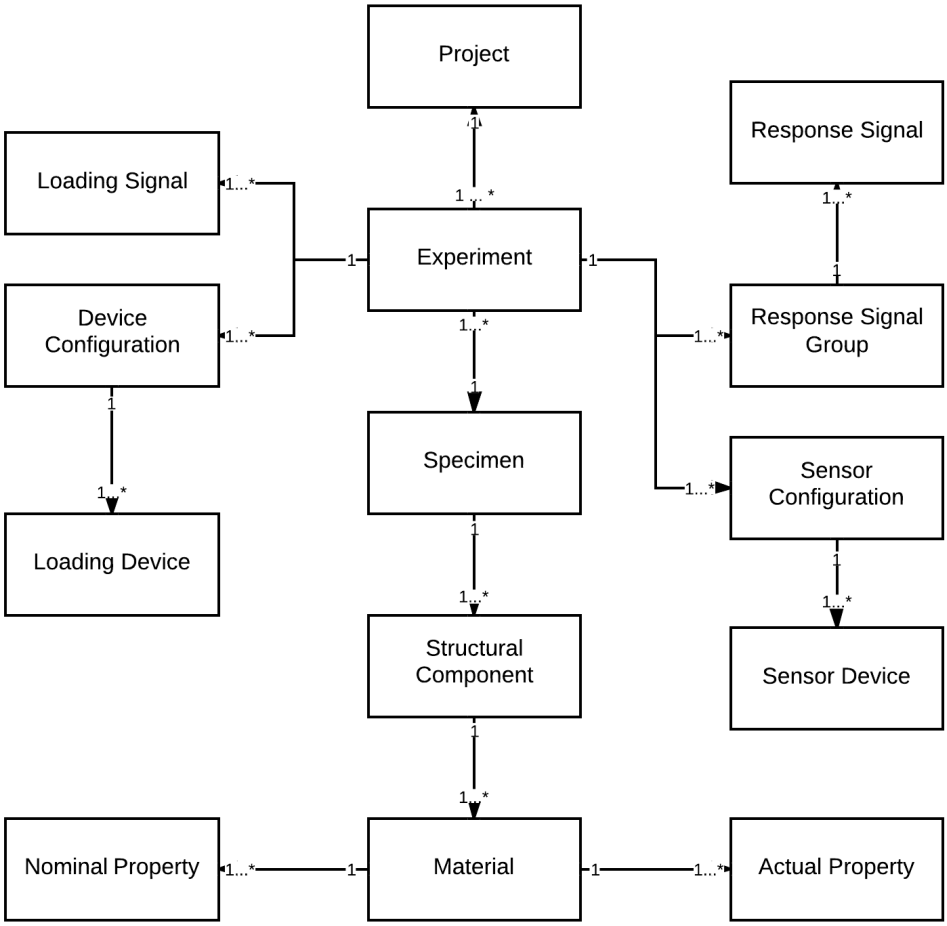

Figure 3.2: Whole data model in a nutshell.

The second part, on the other hand, was developed for the data that are common to all projects. These data include the loading and sensor devices of the laboratory, the individuals took part in the projects and the experiments, the clients that will use the system in the authorization level, the unique keys that are used in order to categorize some of these entities and designed available to be broadened. These entities are shown in Figure 3.3.

The third part contains documents, images and videos that are identified for

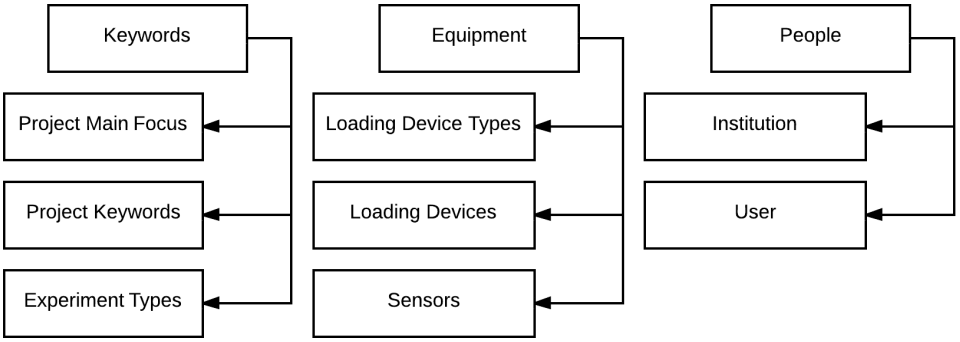

Figure 3.3: Common entities for all projects.

the projects and the related entities of the projects and designed as separate elements. Structure of this part is presented in Figure 3.4.

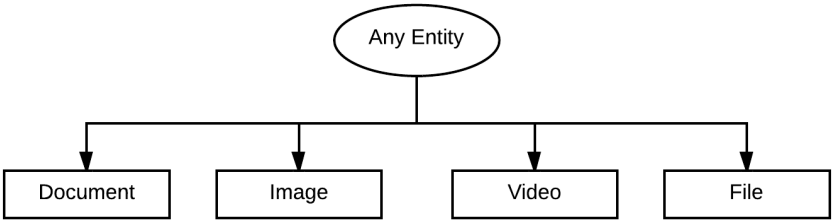

Figure 3.4: File types in database.

The database was implemented using structured query language. As mentioned in the second chapter, SQL systems have a long-standing background, they were tested numerous times and they are reliable and robust systems. Moreover, it is very easy to find qualified stuff to work when there occurs a necessity to further develop and modify the system. MySQL was chosen among a number of SQL implementations. The database management system of MySQL is one of the most powerful amongst competitor systems and has been used successfully in numerous projects for years. One of the reasons why MySQL was used in this project is that the operating system through which the system will run is a distribution of Linux and the Linux distributions are known to work trouble-free with MySQL. Another reason is that the PHP language through which the

graphic interface will be developed. MySQL and PHP have almost the same development history and PHP language is far more compatible with MySQL compared to another SQL implementation. Finally, the developers have more control over the MySQL compared to its competitors.

The database design and the implementation are completed with conformance to the restrictions of the SERIES project. What is meant by this is not the whole structure of the database but the general guidelines and names given to the tables with similar functions and to the table columns. By these means, the query interfaces developed in the scope of the SERIES project can be used with little modification on the database that was directly developed in the context of this project.

Since the system requirements were identified in a large and flexible way, the relation of one table with the other is not established directly in these tables, instead additional tables were created for such relations. Although the implementation facility, in a sense, was put on the back burner, the database resulted in a structure that can be advanced to cover many possible future requirements.

Three database users who are able to connect to the database server were identified. First one of them is the root user who has the administrative privileges and it can only be controlled by the system designers. The second one is the user called "seed" who has the CRUD (Create, Read, Update, and Delete) privileges and this user is controlled by the graphical user interface. The privileges of this user only apply to the existing tables; it cannot create a new table or a user and cannot manage other database users' privileges. Nevertheless, the access credentials of this client should be kept carefully. In order to ensure that, the credentials of this user is defined in a configuration file with .php format and is placed on the server in a folder that is only readable by the web server user defined in that server.

Finally, a user named "series" was created for those who will directly connect to the database without using a graphical interface from the SERIES central server and this user have only the reading privilege in certain tables. The credentials of this user were delivered to the relevant people from the SERIES project.

Amongst these users, only user “series” have the privilege to connect to the database from remote servers. “root” and “seed” users may only connect to the database server from the server computer on which the database server is hosted, i.e. localhost. Table 3.1. depicts the users in short.

User	**Purpose**	**Privileges**	**Access restriction**
root	Administrators	All	localhost
seed	Server side language	CRUD	localhost
series	SERIES central server	Select	% (any host)

Table3.1: Database users.

To manage the database directly, the phpMyAdmin system is used, which runs through the web server and was installed in the computer hosting the database server. This management interface is available in http://dede.ce.metu.edu.tr:8080. The database consists of 93 tables and its server runs in the computer with IP address 144.122.103.100. The database server is serving over 3306 standard MySQL port. Access to the database is limited to the computer on which the database is installed, except the series client. Table 3.2. shows the access information organized for administrators for future references.

Key	**Value**
Public URL to main site	http://dede.ce.metu.edu.tr
Public URL to PhpMyAdmin	http://dede.ce.metu.edu.tr:8080
IP address of the server	144.122.103.100
Hostname for MySQL server	144.122.103.100:3306

Table3.2: Main access information for developers

Currently there are 30 projects, 65 experiments and 66 specimens recorded within the database in date of the creation of this document.

3.3 People Management

In order to manage the information of the people who take part in the projects and use the system in a level that requires authorization, a table called "Person" is created. Thereby, in the other tables, requiring person information, these people are given reference and it becomes easy to obtain accurate information in the operations where the person based content grouping is the case. The Person table consists of a unique key (idPerson), forename (foreName) and surname (familyName), e-mail (contactEmail), phone number (contactPhone), a unique key belonging to the institution of the person (institutionID) and a unique key defined for a user who is connected to the people having the access in authorisation level (idUser).

The idPerson, foreName, familyName, contactEmail and institutionID information are compulsory to fill in order to create a record on this table. The institutional table which is defined by the institutionID key mentioned above is used in order to define the projects or the experiments to which the institutions are related through individuals. Moreover, it makes it possible for the people, who are members of the institutions defined as partners in content access restrictions, which we will mention in the future, to access some of the content. In the institution table, institutions are defined with their unique keys (idInstitution), institution name (institutionName), the short name of the institution (institutionAcronym), contact information of the institution (institutionContact), location (location) and website (institutionWebsite). This data belonging to individuals should be defined before any project creation. Therefore, person management is designed separately from and prior to project management.

In addition to these, the users, which are defined by the idUser key, are defined in a separate Users table with their user names and passwords. Only the people who are also defined in the Users table have access to the system. In this table, there is a unique key (id), a unique key of the person who defines the user (idPerson), a user name (username) and a password (password) for each record. The values in the password column are created with a cryptographic hash function while it is inserted into the database. Likewise, the password

control operations made in order to evaluate log in requests are made with the same cryptographic hash function. SHA1 is used as cryptographic hash function algorithm. The rationale behind hashing passwords is to reinforce security of the system. Here it is relevant to give some details on this subject.

ERD of Person, Institution and User tables are given in Figure 3.5. including all details about the columns.

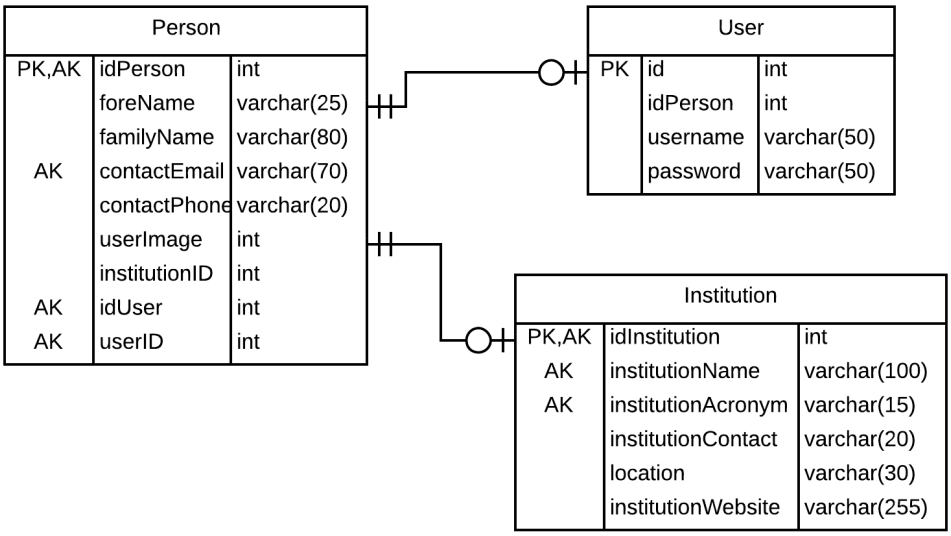

Figure 3.5: ERD of people management.

If all credentials of a user (namely the user name and password pairs) is stored in the database as is and if the database is viewed by unwanted people through hacking or misuse, all credentials can be exposed to the public. In this case, after securing the system, all users should change their passwords or their accounts will be blocked. Besides, it is known that some people use the same password for different accounts. Since such adverse effects in addition to a security breach can occur, it is suggested to keep a modified version of these credentials rather than what they are exactly. This is called encrypting data. There are two main encryption types which are based on reversibility. There are cases that enforce the use of reversible encryption such as sending a classified message from a client to another. If it is not necessary, irreversible encrypting is almost always the most secure solution. Irreversible encryption is done via a function

which takes the value to be encrypted as input, and it outputs another value which cannot be decrypted since the function is irreversible. So that, to know the input of a certain output, it is necessary to operate the same encryption function on the same input. [6] The application of this logic to login systems is straightforward. A user's password is encrypted before it is recorded in the database. Then, when the user wants to log in to the system, he or she provides the same password. The developer adjusts the system to run the same encryption algorithm when recording the password to the database and controlling it during a log in operation. Finally, even if the database security is breached, the hacker only gets the encrypted versions of the passwords which cannot be reversed. By these means, no credential information can be exposed to the public.

3.4 Laboratory Management

Laboratory data refers to the data belonging to the laboratory's loading and sensor devices used in experiments.

The data of the loading devices are stored in Device and DeviceType tables. Since the loading devices can be grouped according to their types, an additional DeviceType table was created. In the DeviceType table, there is a unique key related to the type (idDeviceType) and the name of the type (deviceTypeName). The main data of the loading device is located in the Device table. Every particular device is defined with a unique key (idDevice), a device type key displaying the device type and referring to the DeviceType table (deviceType), subtype of the device (device Subtype), the product name of the device (deviceProductName), the device tag created to provide ease when used in a laboratory (deviceLabel), the stroke capacity value of the device (strokeCapacityValue), the unit of its stroke capacity value (strokeCapacityUnit), the force capacity value of the device (forceCapacityValue), the unit of its force capacity value (forceCapacityUnit), calibration information (calibration), notes related to the device (notes) and a definitive reference to this device given from the local database, if there is any (inventoryReference). Only the notes related to the device and the local database reference information is optional to create a

record for a device. All of the other information must be inserted during the device registration. Figure 3.6. exhibits the ERD of device definitions.

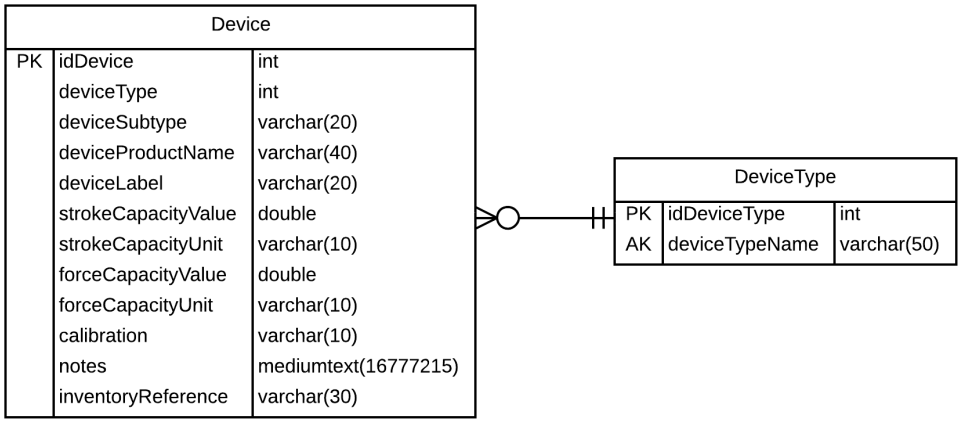

Figure 3.6: ERD of device management.

The data related to the sensor devices are indexed in the Sensor table. Since there is not a practical advantage of grouping the sensor devices with respect to their types, no sensor device type table is used as it was in the loading devices.

Every individual sensor device is created with respect to its own unique key (idSensor), sensor device type (sensorType), sensor device subtype (sensorSubtype), the product name (sensorProductName), the label created for laboratory use (sensorLabel), accuracy value (sencorAccuracyValue), the sensor range unit (sensorRangeUnit), information about calibration (calibration) and if there is a reference from a local database to the device, the information identifying this reference (inventoryReference). Figure 3.7. shows the Sensor table in detail.

At least, a minimum amount of information about these devices should be introduced before the creation of any project since any project will need a device configuration. Therefore, in the system design, they were created separately from and prior to the project management part.

Sensor		
PK	idSensor	int
	sensorType	varchar(50)
	sensorSubtype	varchar(20)
	sensorProductName	varchar(40)
	sensorLabel	varchar(20)
	sensorAccuracyValue	double
	sensorAccuracyUnit	varchar(10)
	sensorRangeValue	varchar(10)
	sensorRangeUnit	varchar(10)
	calibration	varchar(10)
	inventoryReference	varchar(30)

Figure 3.7: Details of sensor entity.

3.5 Files Management

To save a file in the system means to save that file directly to the file system controlled by the operating system. Database, on the other hand, holds only the references of these files. However, every file reference which is saved in the database may not exist in the local file system. Some files are accessible through a URI out of the system and they are defined in the database via this URI. At this point, the foresight of the person who is creating data in the system about the permanence of the files out of the system is of importance. If a file exists in an outer source, but the permanence of this file is suspicious, what should be done is to save the copy of this file in the system, if possible.

Moreover, for the files that are directly relevant to the projects or to the entities within the projects, a number of various data related to these files should be stored. Thus, the ideal way is to save the files on their own tables rather than the tables of the related entities. Considering this, additional tables were created. Usually, the files are document, image or video files. However, there are also other types of files which do not fall under these categories. Since it would be more appropriate to store various data most of which are not common to every file kept in every category, it was decided to create distinct tables for each

category. Therefore, Document tables were created for documents, Image tables were created for images, Video tables were created for videos and File tables were created for other files.

Every individual input in the Document table is identified with a unique key (idDocument), document title (documentTitle), author of the document (documentAuthor), the creation date of the document (documentCreationDate), document format chosen among the formats of DOC, PDF, TXT (documentFormat), document size in bytes (documentSize), the written abstract of the document if available (abstract), the scope of the document (scope), the information which can be INTERNAL, EXTERNAL or ENCLOSED, and shows whether the document exists in the file system (documentLocation), URI of the document (documentURI), if there is any, URI of the original document (originalDocumentURI), the counter which records the number of downloads (downloadTimes), the hash value of the document file which was created by MD5 algorithm to make it possible to understand whether the file was changed or not after recording to the database (hashValue), the value indicating if the document can be downloaded (downloadAllowed), the value which controls the accessibility of the document and can be PRIVATE, PARTNER or PUBLIC (privacy), the unique key of the user who inserts the document in the system (creator_idUser), the unique key of the user who made the most recent change in the document record (lastModification_idUser) and the most recent modification date of the document record (lastModification_time). Document entity is shown in Figure 3.8.

Every individual input in the Image table which holds the references of the image files are identified with the unique key of the image file (idImage), the name of the image file (imageName), the creation date of the image file (imageDate), the format of the image file whether in JPG, BMP, GIF, PNG, or TIFF (imageFormat), the value which indicates the location of the image file and can be INTERNAL, EXTERNAL or ENCLOSED (imageLocation), the person who created the image (imageAuthor), URI of the image file (imageURI), the size of the image file (imageSize), a brief information about the content of the image file (summary), the hash value of the image file which was created using MD5 algorithm (hashValue), the value indicating whether the image file can be down-

Document		
PK,AK	idDocument	int
	documentTitle	varchar(150)
	documentAuthor	varchar(300)
	documentCreationDate	timestamp
	documentFormat	enum(3)
	documentSize	double
	abstract	text(65535)
	scope	varchar(150)
	documentLocation	enum(8)
	documentURI	varchar(255)
	originalDocumentURI	varchar(255)
	enclosedDocument	longblob(4294967295)
	enclosedOriginalDocument	longblob(4294967295)
	downloadTimes	int
AK	hashValue	tinyblob(255)
	downloadAllowed	enum(3)
	privacy	enum(7)
	creator_idUser	int
	lastModification_idUser	int
	lastModification_time	timestamp

Figure 3.8: Details of document entity.

loaded (downloadAllowed), and the value which indicates who has access to the image file and can be PRIVATE, PARTNER or PUBLIC. Figure 3.9. shows detailed entity diagram of Image table.

Image		
PK,AK	idImage	int
	imageName	varchar(50)
	imageDate	timestamp
	imageFormat	enum(4)
	imageLocation	enum(8)
	imageAuthor	varchar(100)
	imageURI	varchar(255)
	enclosedImage	longblob(4294967295)
	imageSize	double
	summary	tinytext(255)
AK	hashValue	tinyblob(255)
	downloadAllowed	enum(3)
	privacy	enum(7)

Figure 3.9: Details of image entity.

The records in the Video table usually refers to the video files created during the experiments. Every individual record in this table is identified with

a unique key (idVideo), the name of the video (videoName), the date of the record (videoDate), the format of the video which can be MPG, AVI, MOV, FLV (videoFormat), the value indicating the location of the video which can be in the system or external (videoLocation), URI of the video (videoURI), the size of the video in bytes (videoSize), a brief description of the video (summary), the hash value of the video file created using MD5 hashing algorithm (hashValue), the binary value which indicates whether the video can be downloaded (downloadAllowed), and the value which indicates who has access to the video and can be PRIVATE, PARTNER or PUBLIC. Video table diagram is given in Figure 3.10.

Video		
PK	idVideo	int
	videoName	varchar(50)
	videoDate	timestamp
	videoFormat	enum(3)
	videoLocation	enum(8)
	videoURI	varchar(255)
	enclosedVideo	longblob(4294967295)
	videoSize	double
	summary	tinytext(255)
AK	hashValue	tinyblob(255)
	downloadAllowed	enum(3)
	privacy	enum(7)

Figure 3.10: Details of video entity.

Every individual input in the File table, where other types of files are created is identified with its own unique key (idFile), file name (fileName), the creation date of the file (fileDate), the format of the file that can be freely entered (fileFormat), a brief description of the file (description), the value which indicates the location of the file and can be INTERNAL, EXTERNAL or ENCLOSED (fileLocation), URI of the file (fileURI) and the hash value of the file which was created using MD5 hashing algorithm (hashValue). File table details can be viewed in Figure 3.11.

The entities which the file is related to is not mentioned in these tables and every file record is meaningful only if it is in the context of the entity that is

File		
PK	idFile	int
	fileName	varchar(100)
	fileDate	timestamp
	fileFormat	varchar(50)
	Description	mediumtext(16777215)
	fileLocation	enum(8)
	fileURI	varchar(255)
	enclosedFile	longblob(4294967295)
AK	hashValue	tinyblob(255)

Figure 3.11: Details of file entity.

related to it, that is to say it has no meaning by itself. Therefore, the records on these tables are entity agnostic. Also, since it is generally the case that there are one-to-many relations between the files and related entities, relations between these files and entities is established in separate tables.

3.6 Project Management

Project management is the most important and the core component of the database. The fundamental elements such as experiments, specimens, constructional components, materials, loading and sensor device configurations are connected to a project whether directly or indirectly. Therefore, not only the user interface but also the database was designed accordingly.

Every project has some specific information independent of other entities although it can only be identified with the entities which are constituting it. Thus, a Project table, in which the information that can be said unique to the project is present, was created. At the same time, since the projects can be grouped according to its main subjects, a ProjectMainFocus table was created where the main subjects of the projects are stored in order to access all the projects related to a specific group easily and properly in the future. In addition to that, every project can be related to more than one subject except from their

own main subjects. Keywords labeling the projects can be introduced similar to the labeling system which is commonly used on the Internet. These keywords are not defined during the project is entered, but defined beforehand in a large spectrum. Thus, the person entering the project can categorize them meaningfully choosing from a large spectrum of keywords. A ProjectKeywords table was created in order to define keywords. The ProjectMainFocus and the ProjectKeywords tables contain a unique key for every individual input and a name identifying the input.

Every project record in the Project table, which is the main table containing the data of the projects, is identified with a unique key (idProject), the title of the project (projectTitle), the short name of the project (acronym), the start date of the project (projectStartDate), the ending date of the project (projectEndDate), a brief description of the project (projectDescription), the creation date of the project in the database (projectCreationDate), the current status of the project which can be one of the values from the set of NEW, FORESEEN, IN PROGRESS or FINISHED (projectStatus), the organization providing financial support (fundingOrganization), a unique key giving reference to the table of main focus subjects for the project (projectMainFocus), the value which indicates whether the project can be updated and can be OPEN or CLOSED (projectUpdate), the value which determines access restrictions, and can be PRIVATE, PARTNER or PUBLIC (privacy), the unique key of the user who creates the project (creator_idUser), the unique key of the user who updated the project most recently (lastModification_idUser), the last modification date of the project (lastModification_time).

It should be noted that since there is one single main focus of the project, the reference of this focus is directly given within the Project table. On the contrary, since a project can be labeled with more than one keyword, an additional Project_ProjectKeywords table was created. Every individual input in the table belongs to a specific project-keyword relation and contains a unique key of the project in this relation and a unique key of the keyword in the relation.

In addition to these, since there might be documents that are specific to cer-

tain projects and these documents might be more than one, a table named Project_Document where this relation is saved was created. Every individual record in this table indicates a project-document relation and identified with a unique key for the related project (Project_idProject), a unique key for the related document (Document_idDocument) and the value which indicates the status of the document and can be PRELIMINARY, FINAL or JOURNAL (documentType).

Also, there is another table called Project_Person, which contains the information of the people who are specific to the project and has been defined before. Similarly, the inputs in this table contains the unique keys belonging to the project and the person in relation and the person's role which can be PRINCIPAL, INVESTIGATOR, LOCAL, and COINVESTIGATOR within the project.

Tables used in project management are given as ERD in Figure 3.12.

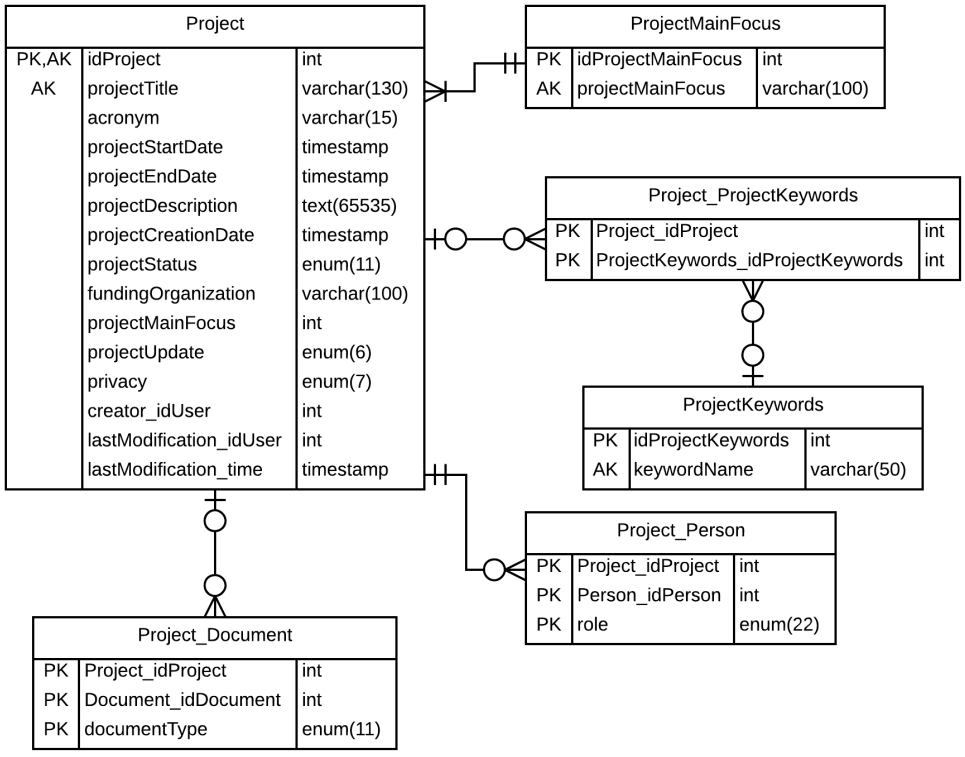

Figure 3.12: ERD of project management.

3.7 Material Management

Materials are the elements that constitute the structural components. Just after the creation of a project, material pools should be created so that the structural components and the specimens which are formed by these structural components can be identified. Although it is sufficient for the materials to be specific to projects according to the requirements of the system, there have been made no restrictions in the database for the sake of possible future changes. So that, any material can be identified with any project in the database. However, since it would make it hard to use the system for the person entering the project, as we will mention later, the materials can be identified specific to the projects in the user interface.

The materials introduced can have many properties. These properties fall under two categories which are actual and nominal. Any material can have more than one actual or nominal property. Also, it is possible for a material to have properties from both categories. The properties of actual and nominal materials are not in a symmetrical form, which means they are identified differently. As the existing projects suggest, a specific material property can only be applied to specific materials. Since these properties cannot be applied to a number of materials, every material property is identified specific to a particular material when the material is being entered.

Moreover, every property, when they are being defined, should be available to be identified with more than one related document regardless of the category they belong to. Extra tables were created for the related documents. Apart from material property documents, since there can be more than one document that is directly related to the material, an additional table of material documents was created.

Hence, the Material table in which the materials are identified, the Nominal-Property table for the nominal properties of the materials, the ActualProperty table for the actual properties of the materials, the Material_Document table for the documents that are directly related to the materials, the NominalProp-

erty_Document table for the documents of nominal properties, ActualProperty_Document for the documents of actual properties were created. Finally, the Project_Material table was created in order to connect several materials to a specific project.

The material table has a relatively simple structure. Every particular record in this table is identified with a unique key (idMaterial), material name (materialName), and a binary value which indicates whether the material has a specific document (hasdocument).

Every record in the Material_Document table indicates a material-document relation and consists of only two columns. The first column indicates the unique key of the material (Material_idMaterial), the second one, on the other hand, indicates the unique key of the document (Document_idDocument).

Every particular input in the ActualMeanProperty table, which is one of the material properties, tables is identified with a unique key (idActualMeanProperty), name of the property (ActualMeanPropertyName), the numerical value of the property (ActualMeanPropertyValue), the unit which indicates the numerical value of the property (ActualMeanPropertyUnit), the number of the samples that the numerical value is obtained with (numberOfSamples), observations about the property (observations), if there is a graphic related to that property, its value vectors (valueVectorX, valueVectorY), the binary value which indicates whether there is a document related to the property (hasDocument), and the unique key indicating to which material the property is related (SCMAT_id).

Every individual record in the ActualMeanProperty_Document table, where the documents related to the actual properties are stored, contains unique keys belonging to the property and document and the meaning of this document in the context of the relation. Every record is identified with a unique key belonging to the property (ActualMeanProperty_idActualMeanProperty), a unique key belonging to the document (Document_idDocument), and the type of the document that can be MATERIALCURVE or ADDITIOANALDOC.

Every individual record in the NominalProperty table, where the records of the

nominal properties which are one of the types related to the material, is identified with a unique key (idNominalProperty), property name (nominalPropertyName), the numerical value of the property (nominalPropertyValue), the unit indicating the numerical value of the property (nominalPropertyUnit), observations related to the property (observations), the value vectors of the graphic of the property, if there is one (valueVectorX, valueVectorY), the binary value that indicates whether there is a document referring to the property (hasDocument), and the unique key indicating to which material the property belongs (materialID).

Every individual record in the NominalProperty_Document table, where the documents of the nominal properties are stored, is identified with a unique key belonging to the nominal property in the relation (NominalProperty_İdNominalProperty), a unique key belonging to the document in the relation (Document_idDocument), and the type of the document which can be MATERIALCURVE or ADDITIONALDOC.

Complete ERD of material management related tables is given in Figure 3.13.

3.8 Structural Component and Specimen Management

The structural components consist of materials. The specimens, on the other hand, consist of structural components. A specimen needs at least one structural component in order to be identified. Experiments can only be conducted on the specimens. Although some sort of specimens have only one structural component, it is necessary to create a specimen that will wrap this structural component.

Specimens are basically stored in the Specimen table. Considering that there might be documents and images specific to the specimens and the number of these types of files can be more than one, the Specimen_Document table was created for document-specimen relations and the Specimen_Image table was created for image-specimen relations.

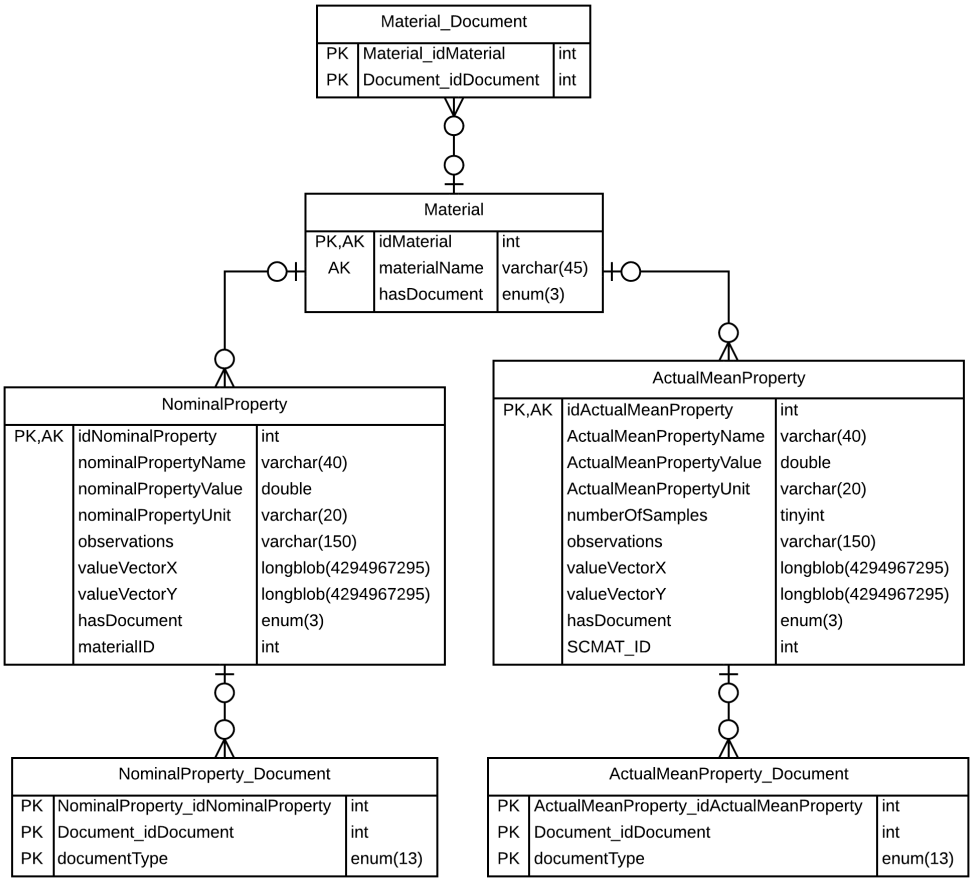

Figure 3.13: ERD of material management.

For the structural components the StructuralComponent table is used essentially. The structural components should be identified for only one specimen. Therefore, the relation between the structural components and the specimen is directly indicated in the structural components table.

The structural components can be categorized in a large and a flexible spectrum. Thus, for these categories an additional StructuralComponentType table was created and every individual structural component is created with reference to an input in this table while they are entered. The structural components can also be related to more than one document like the specimens. Therefore, an additional StructuralComponent_Material table was created.

Considering that a structural component can contain more than one material

and a material can be used in more than one structural component, an additional StructuralComponent_Material table was created for structural component-material relations.

The structural components are not directly related to the project. This relation is conducted through the specimen to which the structural component is connected. A specimen can be identified for only one project; however, a project may contain more than one specimen. Therefore, the relation between the projects and the specimens can be created directly in the specimens table, with an identifier referencing the related project.

Every individual record in the specimen table is identified with a unique key (idSpecimen), the name of the specimen (specimenName), the unique key of the project belonging to the specimen (projectID), the maximum length of the specimen (max_Length), the maximum width of the specimen (max_Width), the maximum height of the specimen (max_Height), the maximum depth of the specimen (max_Depth), the mass of the specimen (specimenMass), the access restrictions for the specimen information that can be PRIVATE, PARTNER or PUBLIC (privacy), the unique key of the person who enters the specimen to the system (creator_idUser), the unique key of the user who made the most recent modifications on the specimen record (lastModification_idUser), the last modification date of the specimen (lastModification_time).

The tables where the references of documents and image files related to the specimens have symmetrical structures. Every individual record in the Specimen_Document table is identified with a unique key belonging to the specimen (Specimen_idSpecimen), the unique key belonging to the document (Document_idDocument), and the value which indicates the type of the document in the context of this relation and can be COORDREFSYSTEM, SCALING, GEOMETRY, CONSTRUCTION, TRANSPORT, DEMOLITION, PRELIMINARY, ONGOING or FINAL (documentType). The first two columns in the Specimen_Image table which stores the images are created in a similar way. The image type information can hold one of these values from the set of MAIN, CONSTRUCTION, DEMOLITION or TRANSPORT.

Specimen and directly related entites are shown as a ERD in Figure 3.14.

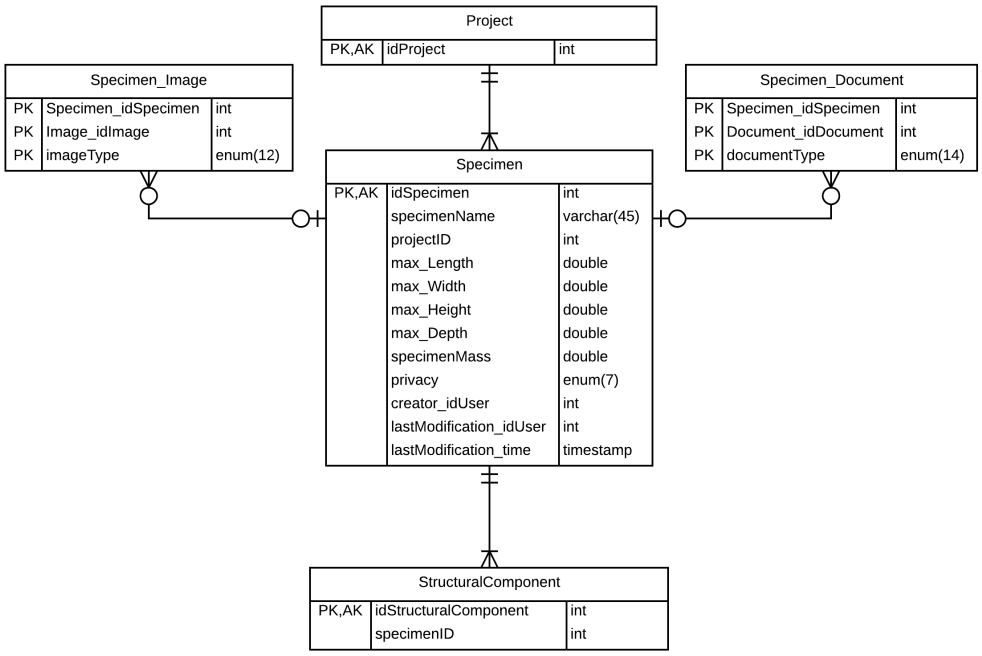

Figure 3.14: ERD of specimen management.

Every individual input in the StructuralComponent table where the structural components are stored is identified with a unique key belonging to the structural component (idStructuralComponent), the name of the structural component (structuralComponentName), the unique key of the specimen to which the structural component belongs (specimenID), the unique key referring to the category of the structural component (SCTypeId), the specific notes about the materials constituting the structural component (materialDescription).

Every individual record in the StructuralComponentType table which was created for the categories that are used in order to group the structural components is identified with a unique key (idStructuralComponent) and the name of the category (typeName).

Every individual record in the StructuralComponent_Document table where the documents related with the structural components are stored is identified with a unique key belonging to the structural component in relation (Struc-

turalComponent_idStructuralComponent) and a unique key belonging to the relevant document (Document_idDocument).

Every individual record in the StructuralComponent_Material table where the records of the materials constituting the structural components are stored is identified with a unique key belonging to the relation (idSCMAT), a unique key belonging to the structural component in the relation (StructuralComponent_idStructuralComponent) and a unique key belonging to the material (Material_idMaterial).

Structural component table with its surroundings are shown as ERD in Figure 3.15.

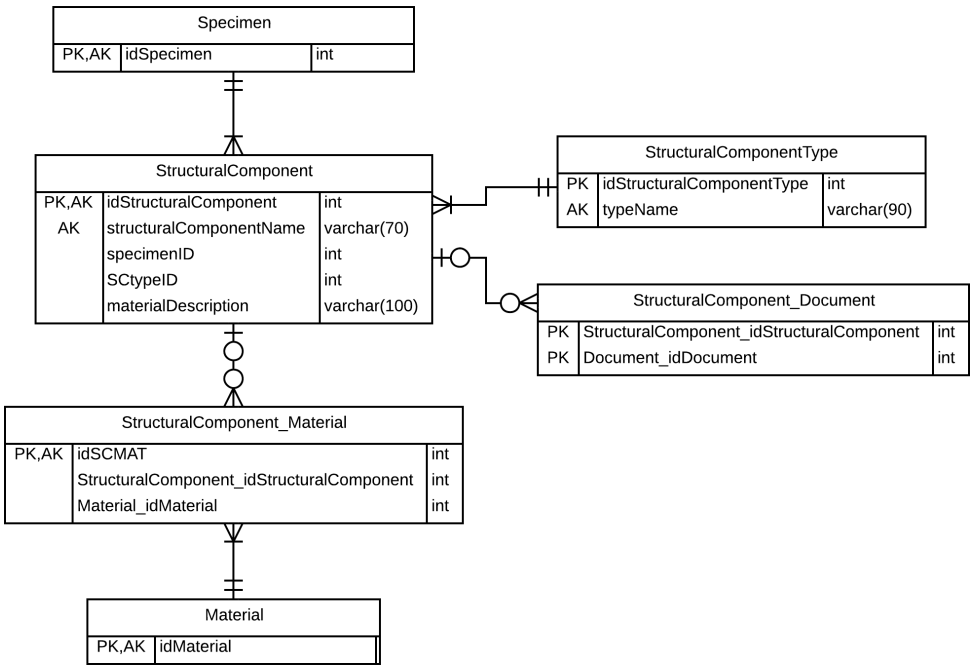

Figure 3.15: ERD of structural component management.

3.9 Sensor and Loading Device Management

The loading and sensor devices in the laboratory are used in groups. Generally, the groups that are created when a project is started, are used for more than one

experiment. For every individual experiment conducted in the scope of a project it is easier to create these device groups beforehand, then to give reference to these device groups in the experiments instead of identifying the same loading and sensor devices for every particular experiment of the project.

The structures of the DeviceConfiguration table which holds the loading device configurations and the SensorConfiguration table which holds the sensor device configurations are very similar. Every input of the DeviceConfiguration table is identified with a unique key belonging to the configuration (idDeviceConfiguration), configuration name (configurationName), and the notes about the configuration (configurationNotes). The SensorConfiguration table has a symmetrical structure. For the devices that configurations contain, the DeviceConfiguration_Device and the SensorConfiguration_Sensor tables were created. The table created for the devices in the loading device configuration has a very simple structure. Every individual record in the DeviceConfiguration_Device table is identified with a unique key belonging to the relation (idDeviceConfigurationDevice), a unique key belonging to the configuration (DeviceConfiguration_idDeviceConfiguration), and a unique key belonging to the device (Device_idDevice). The SensorConfiguration_Sensor table where the sensor devices in the sensor configurations are stored contains lots of information. Every individual record in this table is identified with a unique key (idSensorConfiguration_Sensor), a unique key belonging to the configuration in the relation (SensorConfiguration_idSensorConfiguration), a unique key of the sensor device in the relation (Sensor_idSensor), the initial position of the sensor device in the configuration (sensorLocation), the final location of the sensor device (sensorLocationFinal), the initial direction of the sensor device (sensorDirection), the final direction of the sensor device (sensorDirectionFinal), and the notes about the configurations belonging to the device (notes). Sensor configuration ERD is shown in Figure 3.16.

The system requires that a device configuration should only run with a specific project; however, there is a high possibility for a variety of projects to use the same device configurations. Therefore, the device configurations have not been defined specific to a specific project. Instead, with the prediction that

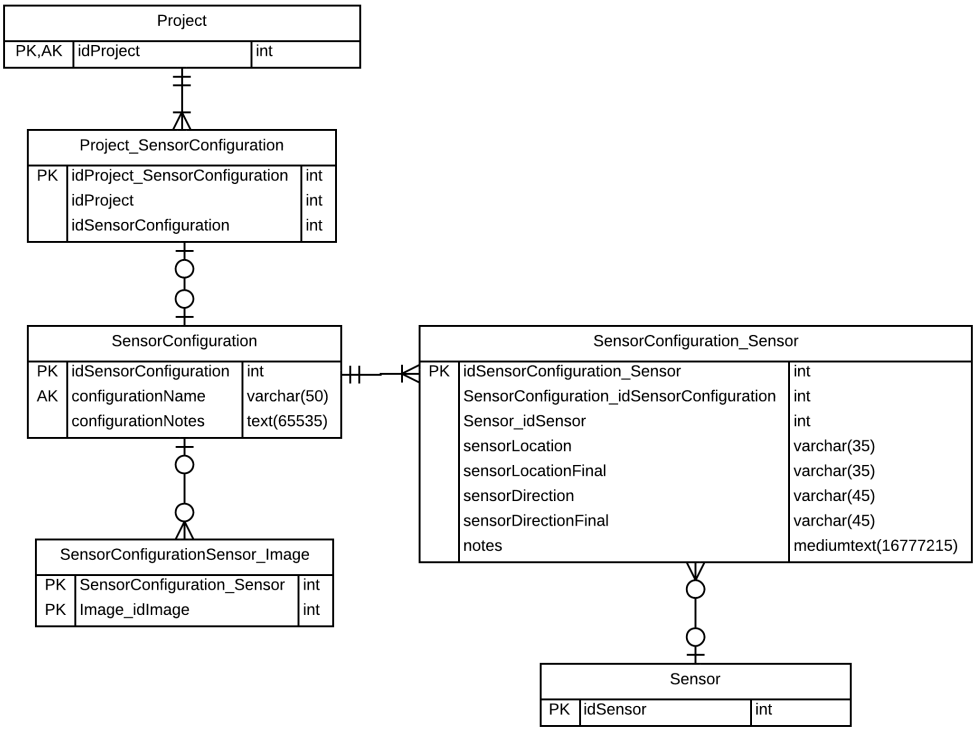

Figure 3.16: ERD of sensor configuration management.

they might refer to more than one project, this relation was created in the separate tables Project_DeviceConfiguration and Project_Sensor_Configuration. In this table, there are unique keys belonging to the related configurations and projects.

3.10 Loading and Response Signal Management

A number of experiments conducted in the scope of a project are performed using the configurations consisting of loading and sensor devices in the laboratory. Since the signals of the loading devices are prepared before the experiments are carried out, they are few in number and each of them are generally created per project, the experimenter prepares them as separate files. The response signals, on the other hand, are obtained in bulk groups corresponding to the sensor device configurations. Most of the time, the person who is carrying out the experiment obtains all the response signals in a single file when the experiment

is over. The readings in this file are equidimensional readings that correspond to a shared time axis.

Since the loading and the response signals have many properties in common, a common Signals table was created in order to record them one by one. The signals created or achieved related with an experiment frequently depends on a common time axis; therefore, an additional SignalTimes table was created for this coordinate.

Every individual record in the Signals table is identified with a unique key (idSignal), signal name (signalLabel), the type of the signal indicating that it is created through a sensor or calculation which can be COMPUTED or MEASURED (type), the unit of the signal value (unit), URI of the signal data that is stored usually in CSV format in the local file system (signalValuesURI), the binary version of the value vector (valueVector), the unique key of the device through which the measurement was performed in the configuration to which it belongs, the value indicating if the signal is obtained directly as a result of measurement (sensorConfig_sensorId), if the signal is a sensor signal, the unique key of the experiment from which the signal was obtained (expCompId).

Every individual record in the SignalTimes table composed of its own unique key (idSignalTimes), the label of values (timeLabel), and the binary versions of the value vector (timeVector).

The loading signals are located in the OriginalLoadingSignal table referring to the records in the signal tables. The OriginalLoadingSignal_Document table was created for the files that are specific to the loading signals. With respect to the system requirements, a loading signal is related to only one project. Thus, it would be sufficient to refer a project or project entity in the loading signal table directly. However, considering the future requirements, another table called Project_OriginalLoadingSignal is created to relate a loading signal with more than one project.

Every individual record in the OriginalLoadingSignal table indicates the original version of a specific loading signal and identified with a unique key (idOriginal-

LoadingSignal), loading name (originalLoadingName), loading type that can be NATURAL, ARTIFICIAL, NATURAL-MODIFIED (nature), the source of the loading signal (source), the maximum loading value (peakExcitationValue), the unit of the maximum loading value (peakExcitationUnit), the binary value indicating whether there is a document belonging to the loading (hasDocument), the unique key belonging to the actual signal to which they refer (signalID).

There are unique keys, which belong to the loading signal and to the related document, in the OriginalLoadingSignal_Document where the folders of the original loading signals are kept. Similarly, there are unique keys, which belong to the related loading and to the project, in the Project_OriginalLoadingSignal table which relates the original loading signals to the projects.

Although the loading signals are, in fact, directly related to experiments, they are tied to the projects in our design. Since, experiments can sometimes do not directly relate to the original version of the loading signals and the original loading signals must be defined in a project without referring to any experiment, another concept called detailed loading characteristics is developed. This situation occurs mainly in experiments which are just computational, and they are conducted using artificial signals which are obtained through the modification of the original loading signals with the help of a specific coefficient. Therefore, the DetailedLoadingCharacteristic table was created where the detailed loading characteristics are stored. In order to prevent any complication in the system, no original loading signal can be directly associated with experiments. Even if the signal will be used without any modifications, first a record is created in the DetailedLoadingCharacteristic table and the relation between this detailed loading record and the original loading record is saved within the DetailedLoadingCharacteristic_OriginalLoadingSignal table. These detailed loading records are associated with the experiments within the ExperimentComputantion_DetailedLoadingCharacteristic table.

The detailed loading characteristics wrap original loading signals. Every individual record in this table contains a reference to the related original loading record, the constant coefficient applied to the original loading record and the

direction of the loading, if available. The table which associates the experiments with the detailed loadings contains the unique keys of the experiment and the loading. If there is an additional constant coefficient which is applied to the detailed loading, it is also included in this table.

The response signals, on the other hand, as distinct from the loading signals, are entered into the system in groups. For this reason, the SignalGroup and the SignalGroup_SignalInfo tables were created. Every individual record in the SignalGroup table is used in order to associate the signal groups with the projects and identified with a unique key (idSignalGroup), the name of the signal group (signalGroupName), the unique key of the project to which the group belongs (projectID). Every record in the SignalGroup_SignalInfo table is created to store the information of a single signal in the signal group and identified with a unique key (idSignalGroup_SignalInfo), a unique key belonging to its group (idSignalGroup), the signal label (signalLabel), the type of the signal which can be MEASURED, ARTIFICIAL or NATURAL-MODIFIED (type), the unit of signal values (unit), the unique key belonging to the specific sensor device in the sensor device configuration from which the signal reading is obtained.

The sensor signals are directly associated with the experiments within the Signals table. It is known that every individual row recorded in the Signals table is associated with a specific experiment. Since it is recorded that which entry in the Signals table corresponds to which entry in the SignalGroup_SignalInfo table, the experiments and the measurements can be associated.

3.11 Experiment Management

The experiment records are one of the most important parts of the system. Many other records are just created in order to mediate experiment entries or elaborate the experiments. Many components mentioned above are linked to the projects through the experiments. Similar to the case in the projects, the experiments can be identified only with the help of numerous tables. Note that, in the context of the system, what is meant by the experiments is not only the actual experiments

carried out in the laboratories but also the computational experiments that are only based on computation. In this respect, the experiments are divided into two main classes, namely EXPERIMENT and COMPUTATION. Regardless of the class they belong to, almost every property of every experiment is shared; therefore, the difference is provided by a feature in a common table instead of creating two separate tables for these separate classes.

The main information related to the experiments are kept in the Experiment-Computation table. It was decided to keep the main focus of the experiments in the ExperimentComputationType table and give reference to this table from the main table in order to provide convenience for the search operations that will be done in the future. Another reason why this table was created is that how the main focuses of the experiments will progress in the future cannot be estimated today.

Every individual input in the ExperimentComputation table where the main records belonging to the experiments are kept is identified with a unique key belonging to the experiment (idExpComp), the name of the experiment (expCompName), the value which indicates the class of the experiment and can be EXPERIMENT or COMPUTATION (class), the value referring to the main experiment focus table (type), the date of the experiment (date), the information of how many times was the experiment repeated (numberOfRepetitions), the maximum excitation value obtained in the experiment (peakExcitationValue), the unit of the maximum excitation value obtained in the experiment (peakExcitationUnit), the unique key of the specimen over which the experiment was carried out (SpecimenID), the value which manages the access restrictions to the experiment and can be PRIVATE, PARTNER or PUBLIC (privacy), the unique key belonging to the client who enters the experiment to the system (creator_idUser), the unique key belonging to the client who made the last modification on the experiment record (lastModification_idUser), the date of the last modification made on the experiment record (lastModification_time), the loading configuration used in the experiment (DeviceConfigurationID) and the sensor configuration used in the experiment (SensorConfigurationID). The unique key of the focus, the name of the focus and the class of the focus are

kept in the table where the experiment focuses are kept.

Additional tables were created to provide the links between the experiments and the other entities. The ExperimentComputation_DetailedLoadingCharacteristic table was created in order to relate the detailed loading characteristics to the experiments. The ExperimentComputation_Document, the ExperimentComputation_Image, the ExperimentComputation_File and the ExperimentComputation_Video tables were created in order to store the documents, images, videos and other types of files specific to the experiment. Finally, in order to re-identify the people in the perspective of the experiment, another table called ExperimentComputation_Person is created. Every one of these additional tables contains the unique identifiers of the components and the experiments in relation.

CHAPTER 4

THE GRAPHICAL USER INTERFACE

4.1 Introduction

This chapter explains the graphical user interface developed to manipulate the database system mentioned above in detail. All the implementation details required to develop the interface as well as the design philosophy is given for further development. Firstly, overall application structure together with the languages and tools used to develop the interface is explained. Then the components that is common to all parts is pointed out. In following parts, main functionalities offered by the interface is described under related pages of the website.

4.2 Overall Web Application Structure

The ultimate goal of the system is to store specific data in a relational database satisfying the criteria of being economic and reliable and to provide access when there is a need for these data in an appropriate way. Considering it just in the perspective of this goal, the study is nothing but a database management study. Database management, on the other hand, can only be made through some specific user interfaces that might not be even graphical. However, that type of use requires special training of the staff. The purpose of the graphical user interfaces in a project like this is to provide utilization of the system for those who do not have expertise on database management systems. The users considered here are not only the clients who only view the data, but the users

who will also use the system in an authorized level, i.e. enter, update or delete data. So that, interface, which was developed for the users who want to enter and access to data, should be of a type that is both easy to access and use as far as possible.

The graphical interface software that can be used to manage a certain database generally divide into two. First one of them is the applications which were developed specific to operating systems. They generally require installation beforehand and are named as desktop software. The second one, on the other hand, are the website applications which are developed for the web browsers installed on the computer and do not require installation. The shared feature of these programs is that they both need a computer that has access to the Internet. In the context of this project it is decided to implement the graphical user interface as a web application.

The first motive of this decision is that the web applications are easier to access in comparison with the desktop software, since they do not require installation. A web application does not require installation because the users are served only what they will view. The main business logic of the software does not need to run on client's computer. The whole installation of the system, in fact, was made in the remote server in which the web application is hosted. When a user accesses to a specific URL address via a browser, they do not access to the whole system, but the resulting view of the source they want access to. This view is named a web page. Web pages are usually small in size since they store a specific content statically. A general scheme of data flow in web is given in Figure 4.1. Traditional data flow in traditional systems is also shown in Figure 4.2. Moreover, users only access to the pages to which they want to access, not to all pages of the system when browsing a website. Also, the copies of the pages accessed are stored in cache storage on the user's computer. By this means, a user can browse a web site repeatedly with minimum data transfer between the server and the client. As a result of all the reasons mentioned, it is possible to run the system providing the same functions without any installation on the users' computers.

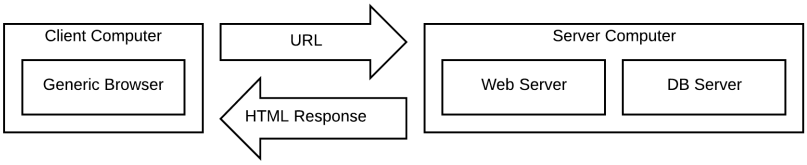

Figure 4.1: Web based database client software work flow.

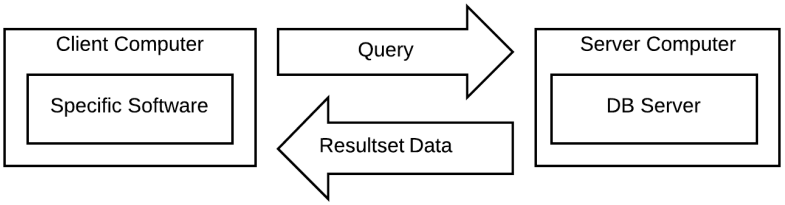

Figure 4.2: Traditional database client software work flow.

The second motive is that web applications do not have any other dependencies apart from browsers. The desktop applications, on the other hand, should be developed accordingly or at least adjusted specific to an operating system. Since the possible users of the system have already been using many different operating systems or different versions of a specific operating system, it would be very hard to develop a desktop application to run seamlessly on all these environments. Likewise, the web applications also depend on many browsers; however, since there are common standards for these browsers, the development procedure should be carried on in accordance with these standards only. Thereby, a single web application can be run having the same functions and appearance in every major browser.

As mentioned before, a user who wants to access a certain page in a central computer should be sent that page. The prior necessity for this is a presence of a web server software in this central computer. A web server application creates a service on the computer it is installed that is used to listen to a specific port for resource requests and serves those resources according to the specifications implemented by the developer. There are many web server software products

such as Apache, IIS, Nginx, and Lighttpd have already been used throughout the world. Apache was found convenient for the central server created for this study since it is known to be reliable, secure, compatible with server operating systems which are Linux distributions and maintained by a large and devoted community.

The webpages aforementioned are created in a similar way with the desktop applications using certain computer languages. The common feature of all web applications is that the backbone of every page of them is coded using HTML (Hyper Text Markup Language). If a web page was not coded directly by HTML, it was necessarily coded by the software (like Adobe Dreamweaver) which translate into or a computer language (like Jade) that exports in this language. HTML is a markup language and is used to mark the content of a certain webpage. The text areas, links, images and the other components in webpages were created using HTML. In modern web applications an additional language called CSS (Cascading Style Sheets) was developed in order to separate the content from the specific appearance of the content. It is a language that is used to make formal adjustments upon pages. This language, which was used to apply a specific view to the content marked up by HTML, has transformed into a de-facto standard. The graphical interface developed within the context of this study was created through direct HTML and CSS coding.

Apart from the codes related to marking up and shaping the content written by HTML and CSS languages, a software running in the server hosting the website is needed in order to save the content provided by the authorized user in the database and prepare it for presentation. This software should be custom developed in accordance with the requirements of the project. To develop this part, there exist many programming languages such as Python, Perl, Ruby, and PHP which can run server-side and are compatible with Apache. Within the context of this study PHP is chosen. One of the most important reasons for this choice is that it is a language that has been tested for a long time and confirmed as reliable. Moreover, it is known that PHP runs in a compatible way with MySQL. The main mission of the part which was developed using PHP is to store the input obtained from the user after processing in a suitable way, to

create a website ready for presentation by accessing to the data in the database in accordance with the user's demand, and to perform all these in a way that fits to the authorization levels of users.

At the same time, JavaScript programming language was used for the functions that should work client-side in order to provide a visually rich experience for users. JavaScript, a language, which is the only language supported by all of the browsers, is the only choice for such operations.

All the technology stack used in the system is shown diagrammatically in Figure 4.3.

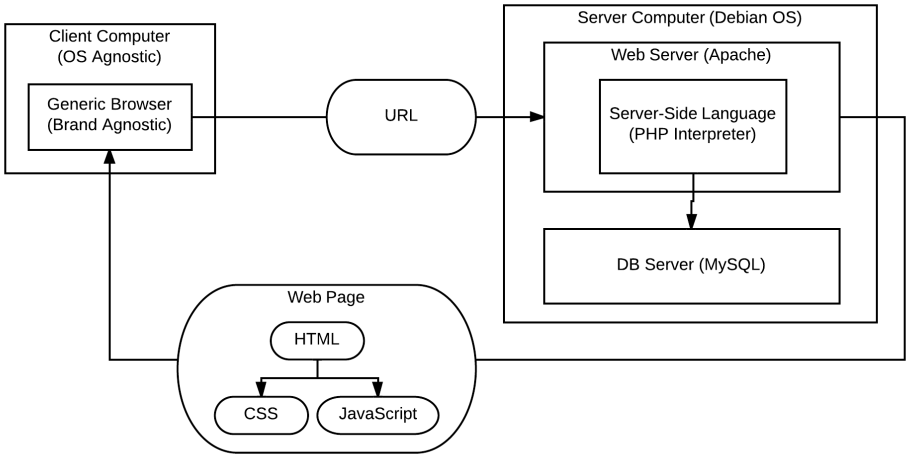

Figure 4.3: Technology stack used in this study.

Except from all these language alternatives, software libraries and frameworks that are created using these languages were utilized. The essential toolkits used in this project are listed as a JavaScript library jQuery for the visual functions that will be executed on the client-side, jQueryUI framework which contains tools to ensure the aesthetic quality of the visual composition, the jpGraph library which was written in PHP and is run on the server side in order to have analytical graphics plotted in desired quality, the Asido library which was written in PHP and is run on the server side in order to have the images loaded by users optimized and converted into formats that are proper for storing.

No special integrated development environment was used when developing the graphical interface; instead, simple text editors were used. The Mercurial model was chosen for version control and bitbucket.com provided the cloud storage area which is suitable for this model.

The website contains more than a hundred files in .php format. Many of these files have more than one responsibility. The resulting complexity of the system was reduced to a minimum level by a suitable folder structure.

The pages like homepage, pages belonging to the projects and to the entities related to the projects, pages for general functions were placed in the root directory. The root directory contains only these type of pages. /css folder was created for the pages in .css format, which manage every type of visual elements and /js folder was created for the files in .js format, which were created for the functions that run client-side. /rs folder is used for the static images in the website while /data folder is used for every document, image and other types of files that belong to the input users entered. The functions created for the common operations in the main files were stored in the /inc/custom folder being grouped in appropriate classes due to object oriented design principles. The design of the main files was made with a consistent method. The main files in the root directory change their functions due to the variable indicated in the URL and include the pages having the codes belonging to the related function. For instance the project.php page in the root directory is responsible for project management. This page dynamically includes the project.php file in the /inc/forms folder when the variable indicated in the URL is in add type. A symmetrical form was followed for all other components. Just as the /inc/forms folder has been created for addition operations, for listing operations the /inc/lists folder, for detailed view pages the /inc/pages folder, for tree view the /inc/trees folder were created. All of these folders have been filled symmetrically. The common parts like the header and the footer which are used in every page are called from /inc/htmlclient or /inc/htmleditor folders, dependent on whether the user viewing the pages has logged in or not. A schematic representation of the file structure of the server can be seen in Figure 4.4.

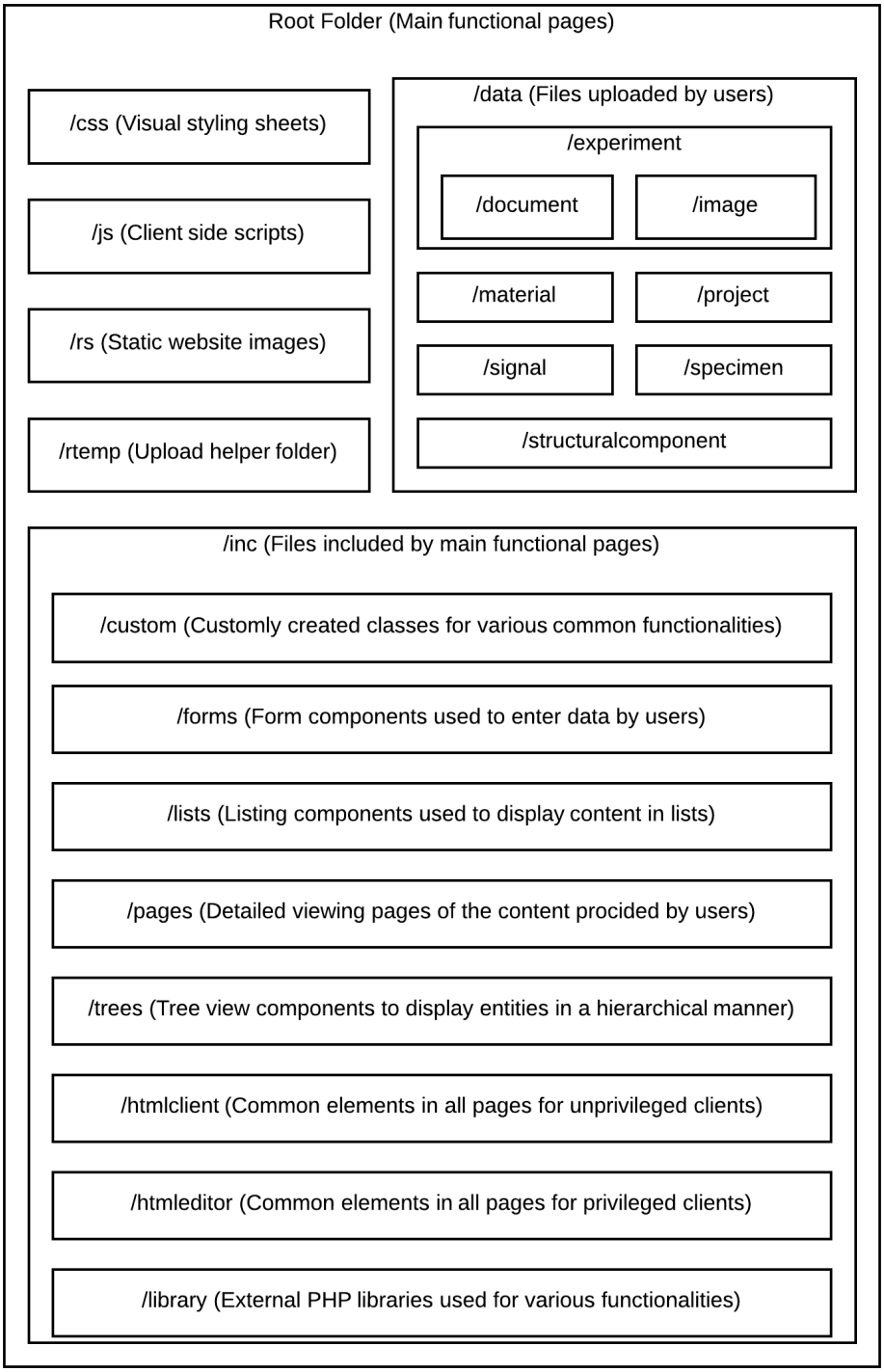

Figure 4.4: Generalized schematic view of server file system structure.

In the next chapters, the parts of the graphical interface, which was developed for the management of the components belonging to the system, will be investigated

one by one.

4.3 Common Parts

There is a header section in every page. The links used to change the language of the system and the navigation bar used to navigate in the system are placed in the header. The header section is shown in Figure 4.5.

Figure 4.5: Common header part of the web application.

The structure and the content of the navigation bar changes depending upon whether the user has signed in to the system or has opened a project or not. While the users who have not signed in to the system can only see the links that are used in order to view the content, the links showing the pages that can add data are located separately for the users who have signed in. The change in header bar for privileged and unprivileged users is depicted in Figure 4.6. and Figure 4.7. respectively.

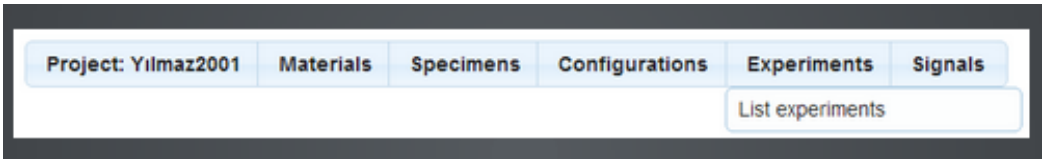

Figure 4.6: Navigation bar view for unprivileged users.

Also, before a project is chosen, there are only the links related to the system in general and laboratory management in the navigation bar; however, once a project is chosen the links in the navigation bar are changed so that the user can view the entities related to the project. These versions of the navigation bar is shown in Figure 4.8. and Figure 4.9. This type of design provides access

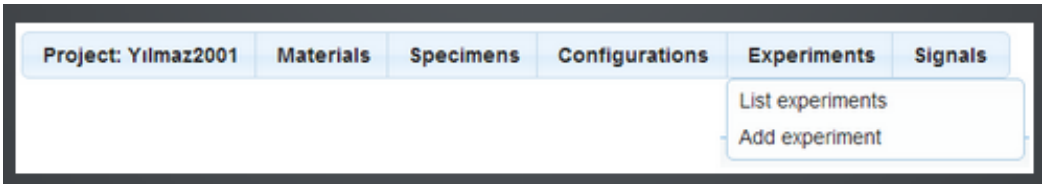

Figure 4.7: Navigation bar view for privileged users.

to the whole system with a single navigation bar.

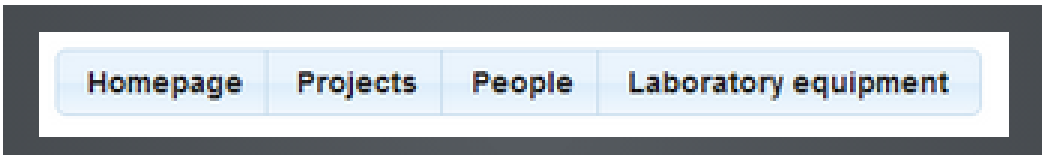

Figure 4.8: Navigation bar view before a project is chosen.

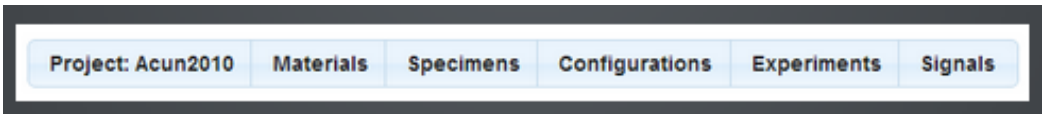

Figure 4.9: Navigation bar view after a project is chosen.

The sections in the page's content, on the other hand, are separated from each other by using visual frames. This frame structure is a generic structure which is used to group and separate all the content in the website. Additionally, these frames have been adjusted to behave like an accordion to avoid the visual complexity. By this means, users can close the other contents when they want to focus on a specific content in a specific page. A sample frame showing the accordion structure can be seen in Figure 4.10.

Apart from these, all of the elements in the page with similar functions have been applied a similar visual design. The elements such as buttons, links, input boxes, forms does not change visually based on pages. Such elements are used in a common structure as far as possible because they accelerate the process of being accustomed to the system.

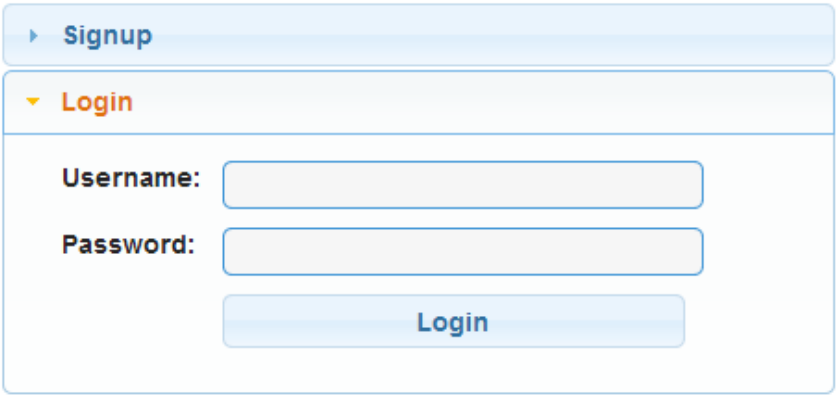

Figure 4.10: Generic frame structure to contain specific content or functionality.

4.4 Homepage

The homepage of the interface is created by the file named index.php which is located in the root directory of the server.

The search section is located in the top left region of the homepage. With the help of this section, users can search the projects, experiments, specimens, materials or people against a keyword. The search results are shown in another page. This section is available to make a number of improvements and it can even be the case that it can be moved to another page specifically tailored to implement the search functionality. An advanced version of this section has not been implemented within the context of this project.

In the top right on this page there are the forms which are used to request membership and sign in. Users can request for membership by filling the membership form. If the system administrator approves these requests, these users can sign in the system using log in form. The users are identified with a user name and password pair.

In the middle of this page, there is a list of projects that have been entered in the system in alphabetical order. The users that have not yet been signed in can only view the projects in which the privacy value is set as PUBLIC. The users that have signed in can view both the projects belonging to their institutes

whose privacy is set as PRIVATE and the projects belonging to institutes that identified themselves as PARTNER. When any of these projects is clicked on, this means a project is chosen system wide and the user is directed to the project information page. Moreover, the links in the navigation bar are set to target to the pages related to the pages related to the project components and a release button to turn back to the homepage is added.

Main components of homepage can be seen in Figure 4.11.

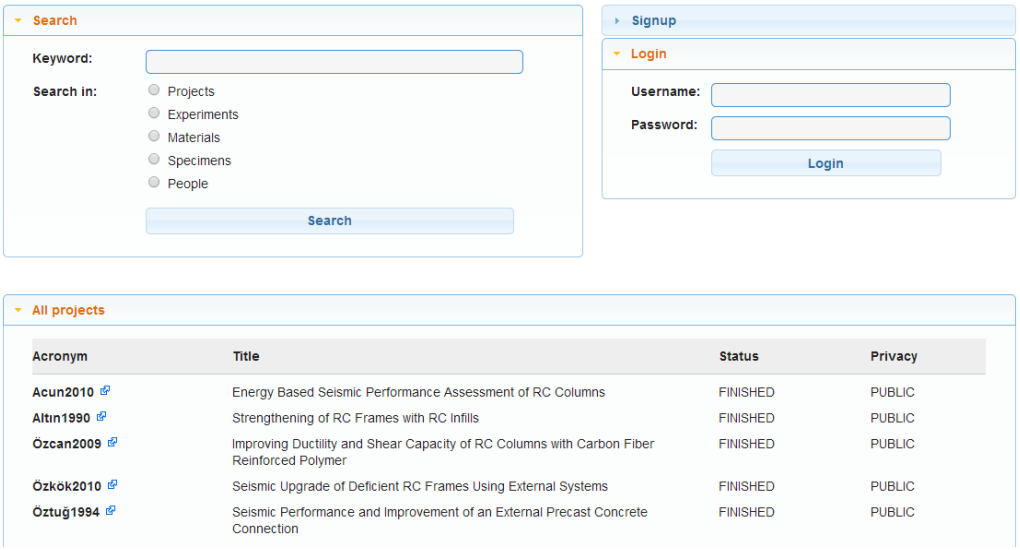

Figure 4.11: Main components of homepage.

4.5 Project Pages

The procedures related to the projects are carried on through the project.php page. The function of this page is changeable depending on the variable in the URL. When a project in the project list is clicked on, the page where the information and the documents about the project are viewed can be accessed. The links on the navigation bar change to reference to the components belonging to the project once a project page is opened. At the same time, a button which is used in order to turn back to the home page is located on the navigation bar releasing this project.

When there is no chosen project, a person with sufficient privileges can have access to the project adding link located on the navigation bar. A project can be added by filling the sections for the data which are specific to the project mentioned in the database topic. Project addition form is shown in Figure 4.12. to be an example to show all addition forms since they are extremely similar. The added projects take place in the project list without additional approval. Project listing component is placed on homepage and can be seen in Figure 4.11.

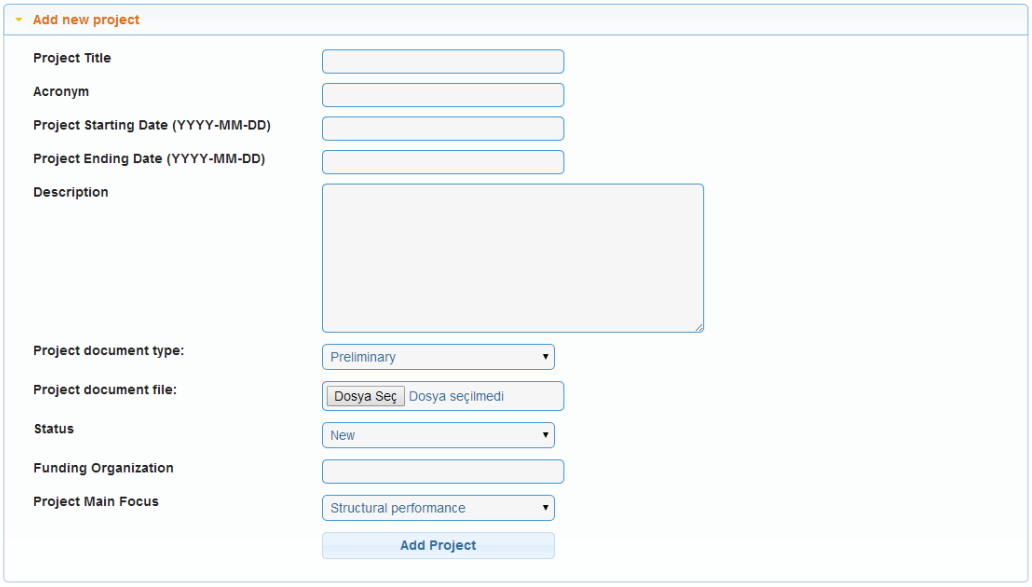

Figure 4.12: Sample for all addition forms.

The details such as the title of the project, the acronym of the project, related dates, and the main focus of the project are indicated in the project details viewing page. The users with sufficient privileges can directly edit these properties on this page. The users with sufficient privileges to delete the projects see a delete icon by the projects in the project listing page. Project details viewing component can be seen in Figure 4.13.

4.6 Material Pages

The operations belonging to the materials are made through the material.php page. The function of this page is changed depending on the variable in the

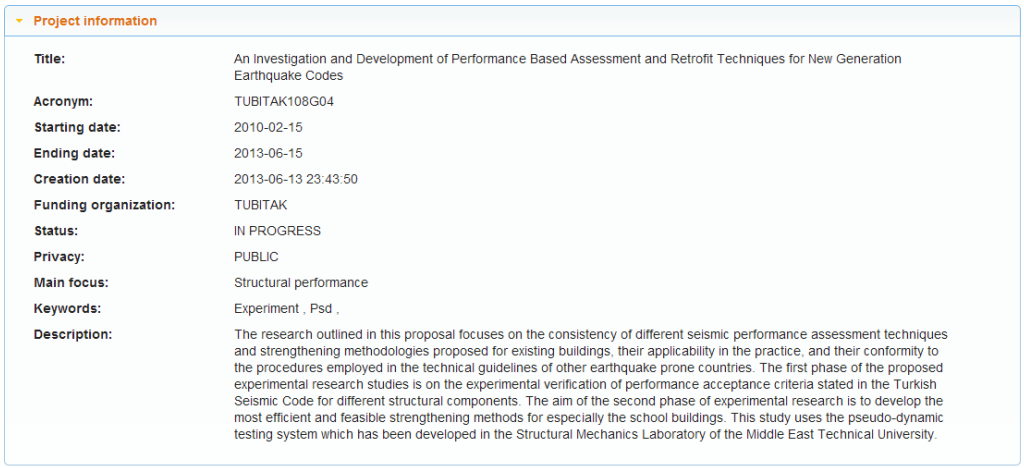

Figure 4.13: Sample for all detailed view components.

URL. The list of the materials which are identified as belonging to the project is available on the page which was accessed through the material list link after the project is chosen. A user with sufficient privileges sees an icon near the materials in the list that can be used to delete the material nearby.

The form page which is used in order to add a specific material has a dynamical structure. It was mentioned before that the materials have a variety and an uncertain number of properties. A small form, including input boxes for all attributes of the properties constitute the material, is added to the page through a button in the material addition form. The user can add the actual or nominal properties via this small form. The user can submit the whole form when the main form and small property forms has been filled with relevant data. In Figure 4.14., material addition form is given to be an example for all expandable addition forms. By this means, the material becomes added to the chosen project. This material which has been added is viewed simultaneously in the material list page. When a material on the material list is clicked on, a page is opened where all the information and the actual and nominal properties of the material are present. The material properties on this page can directly be edited by a user with sufficient privileges.

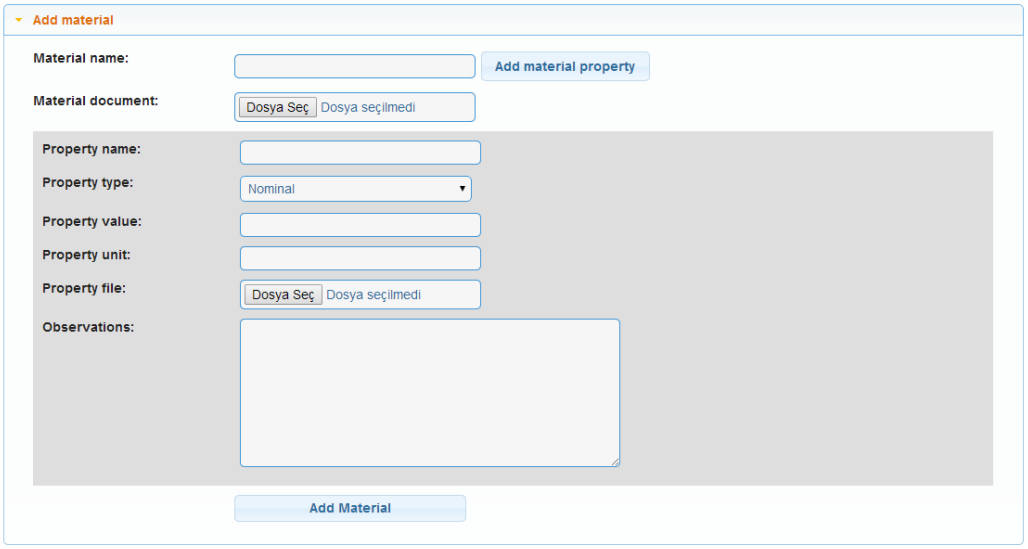

Figure 4.14: Sample for all expandable addition forms.

4.7 Specimen and Structural Component Pages

The operations of specimens are made on the specimen.php while the operations of structural components are made on the structuralcomponent.php pages. Both of these pages change their functionalities according to the variable declared in URL.

As it was mentioned before, there is a hierarchical order amongst the materials, the structural components and the specimens belonging to a project. In order to view and manage this hierarchy easily, a visual specimen tree has been designed. This specimen tree is viewed on the left side of all the pages belonging to the specimens and the structural components. Specimens are located in the top level of the specimen tree. Under each specimen there exist the structural components that constitute them. Similarly, the materials that constitute the structural components are listed under every structural component. All of the specimens and entities related to these specimens in the project are easily accessible through this tree. The users, having sufficient privileges are able to directly add a component to any level in the hierarchy through this tree. Tree view of specimens, structural components and materials is exemplified in Figure 4.15.

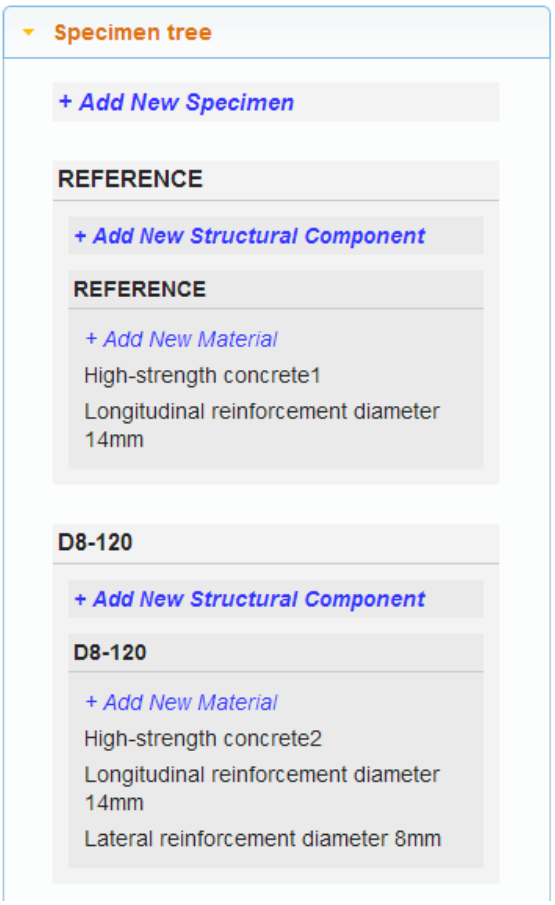

Figure 4.15: Sample for all tree views.

The reason why the materials have been created separately and formerly, is that a material can be used in a number of structural components. However, the structural components can only be used in a specific specimen. Even if the same structural component will be used in two different specimens, they should be identified as two distinct structural components. The reason for this was explained in detail in the previous sections.

Specimens are identified before their components. A specimen can be created by filling the input with the properties specific to the specimen. This specimen that has been created is located both in the specimen list and in the specimen tree. The information added can be viewed in the specimen details page by users with the necessary permission.

In order to add a certain structural component, first the specimen to which this

structural component belongs should be identified. Then a structural component can be added to the system using the link in the navigation bar or the link located under the specimen in the specimen tree to which it belongs. Structural components can be added with the information and documents that are related to it and choosing the specimen which it belongs to. The materials that constitute a structural component are not added during the identification process of a structural component. A link appears in the specimen tree, under each structural component, to associate a material after a structural component has been created.

The specimens and structural components can be edited and deleted through the list and detail view pages by a privileged user, like the other components.

4.8 Device Configuration Pages

The operations related to the loading and sensor devices are made on the device-configuration.php and sensorconfiguration.php pages. The device configurations, as is known, belong to the projects. However, the devices that constitute the device configurations are identified throughout the laboratory.

The operation of adding a specific device configuration is made in the similar way of material adding operation. The device configuration insertion operation is made via a page containing a form consisting of input boxes for the name of the configuration and a brief description of the configuration. In this form there is a button which is to add a device to the configuration. When this button is clicked on, a select box is added to the form. If the configuration is a sensor device configuration, except from the select box, the input boxes are viewed which are also used in order to indicate the location and the direction of the sensor device. These additional forms are filled up with the devices located in the laboratory. The user can choose a device through these boxes. In order to add more devices, it is necessary to add more boxes. After all the devices are identified in this way, the device configuration can be added to the system.

Then, clicking on the items on the pages belonging to the created configurations,

the related configuration page can be accessed. On the page of the related configuration the information about the configuration and the list of the devices constituting it are available. The devices located here contain a link referencing to the device pages. By this means, more detailed information about the related device can be reached.

Another important detail is that when the configurations of the sensor devices are added the order of the sensor devices should be in accordance with the order of the signals that are to be obtained. Because, the system matches each signal in the signal groups that have been added with a sensor device located in the sensor device configuration, in order to simplify the user's work. This matching is done with the presupposition that they are sorted in the same way. The details about introducing the sensor devices and the sensor groups to the system are given in the next sections.

The deleting and editing operations of these configurations are the same with it was mentioned in the all other entities.

4.9 Experiment Pages

The operations related to the experiments are made through experiment.php page. This page, like many others, changes its function depending on a variable in the URL.

In the left side of the experiment page the experiment tree is located. The purpose of this tree view is to locate the experiments under the specimens. This tree view is viewed exactly the same in all the pages related to the experiments and shows all the specimens in the chosen project and all the experiments belonging to them.

As it was stated before, more than one experiment can be identified through a specific specimen. Since the experiments are directly belonging to a specimen, the related specimen should be identified beforehand so that the experiment can be added. Additionally, since every experiment is carried out with a certain

loading and device configuration these configurations should be added formerly.

When these primary conditions are provided a privileged user can add experiments. In order to add an experiment, the form created for information about the experiment such as the name, date, category should be filled. Since there is a high possibility that there might be more than one image file belonging to an experiment there exist image file addition boxes that can be expanded dynamically. Users can add as many images as they want to the experiment through this form. While filling this form, the related specimen and loading and sensor configurations should be specified. Once this operation is completed that means the experiment is completed also and is located in the experiment list. The experiment is also added to the tree view located under the related specimen.

After the creation of a specific experiment all information, documents and images about the experiment, the specimen to which the experiment belongs, the configurations by which the experiment has been conducted can be viewed when the view page of the experiment is navigated. Since all of the entities mentioned here are indicated as links, the detail pages belonging to these entities can be accessed directly by clicking on. At this stage the signals that are the important parts of the experiment cannot be viewed since they are not added already. The reason that they are not added to the system is that the addition of the loading and sensor signals is a complex operation. It is decided that these operations will be completed later in a separate process. The signals obtained after these operations can be added to a chosen experiment. In other words, the experiment entry operation cannot be completed just filling the experiment entry form. It is completed after the signals are added.

An experiment can be deleted and its properties can be edited through the list and details pages, as it was the case in the other components.

4.10 Loading and Response Signal Pages

The operations that are necessary to create and view the loading and sensor signals belonging to the experiments are made through the loading_signal.php,

response_signal.php and signal_group.php pages. All of these pages can change their functions depending on the variables in URL.

Identifying the loading signals and attaching them to the experiments are simple operations. A file containing the time and force vectors of the signal, along with the information and documents belonging to the signal, should be indicated in the form while a loading signal is added. The format of this vector file has been strictly specified. It is obligatory for this file to be in .csv format. The content of the file should be composed of two columns and each value in each column should be separated by a semi-colon. The first column signifies time axis while the second column signifies the corresponding forces. The units of the axis are indicated through the input boxes that are also located in the form. While the loading signals are entered, the device configuration that applies this loading is not indicated also because this configuration has already been related when the experiment which the signal will be associated with was entered. All of the loading signals are listed on a listing page. All information, documents and a loading signal belonging to the signal that has been chosen from the list are available on a detailed view page. Moreover, based on the data in the file which indicates the signal vectors, a signal graph is plotted and it is viewed on the signal details page.

After the loading signals are entered in this way, they are associated with the experiment through the "Attach loading signals to experiments" link which is located on the navigation bar. When the related experiment's page is navigated after this attachment is done, the files of the vectors belonging to the system and the vectors in the scatter graph form can be viewed. A sample plot of a loading signal is shown in Figure 4.16.

The sensor signals can be added to the system after a more complex procedure. It was mentioned earlier that the response signals, obtained after an experiment, are many in number and obtained as groups. Moreover, the time axis of these signals is common to all. Since it would be a long and exhausting procedure to add these signals one by one to the experiments, paying attention to the user experience, it is decided that they will be entered at the same time. Therefore,

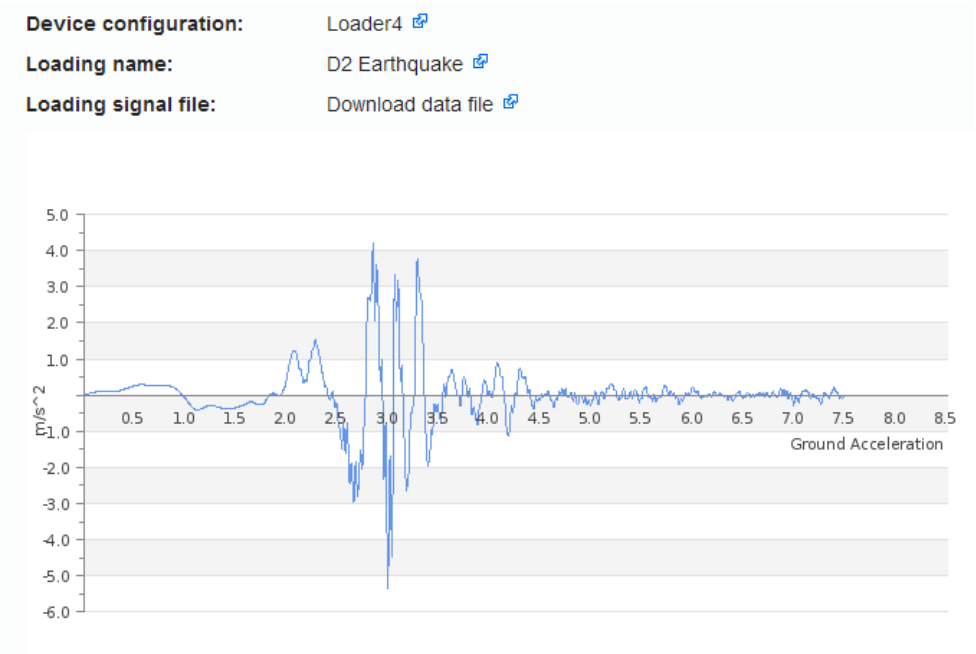

Figure 4.16: Sample plot for an experiment loading signal.

as it was explained before, the database and the graphical user interface has been designed in accordance with that fact.

In order to enter the response signals, the groups belonging to the response signals should be created in advance. The name and whether the common axis of all signals in the group is in the type of time or ID are indicated in the form that is used to create a signal group. The same user adds as many small forms as the signals in the group to the main form. The properties such as the label, the unit, privacy and type of the signal that is to be added are indicated in each small form. The order of these forms should be the same with the order in the file which contains the response signals obtained. By this means, after a signal group is added it becomes accessible from the signal groups list.

Afterwards, another form is used in order to add the signals belonging to that group. In this form, the signal group to which the signals that are to be added and the experiment to which these signals belong are indicated. Finally, the file containing the signals is entered to the form. The format of the file containing the signals has been strictly designed. The format of this file must be .csv. The first column in the file should indicate the vector belonging to the common axis

of the signals. Later, all the signals should be in the order which was stated in the signal group as separate columns. All of the columns should be separated with a semi-colon.

After these operations are completed, all of the signals in the signal group can be listed separately in the page where the response signals are listed. The signal graph, as automatically plotted, is located along with the information about the signal on the detail page of any signal.

Once these operations are completed, the operation of entering a specific experiment is completed also. It is possible to see that the plotted response signals are located on the experiment page when any experiment detail page is navigated back. Since there are a number of response signal belonging to an experiment, they are not plotted separately in order to keep the experiment page in acceptable size. Instead, a single scatter graph used to present all of the response signals is included. Under this graph a form is located which provides users to have any signal plotted according to time or to any other signal. The first box in this form signifies the vector that will correspond to the X axis while the second box signifies the vector that will correspond to the Y axis of the graph. After the users make their decision, they can have the graph plotted by clicking on the "Redraw" button without refreshing the page. This functionality is shown in Figure 4.17.

The raw data files belonging to the loading and response signals are also accessible on the experiment details pages.

Deleting and editing operations of the loading and response signals are hidden on the listing and detail viewing pages, like the other entities. A privileged user can see and use these icons.

4.11 EDF and Connection with SERIES

Exchange data format is a special format developed in the scope of distributed database studies conducted in the SERIES project. The purpose of this format

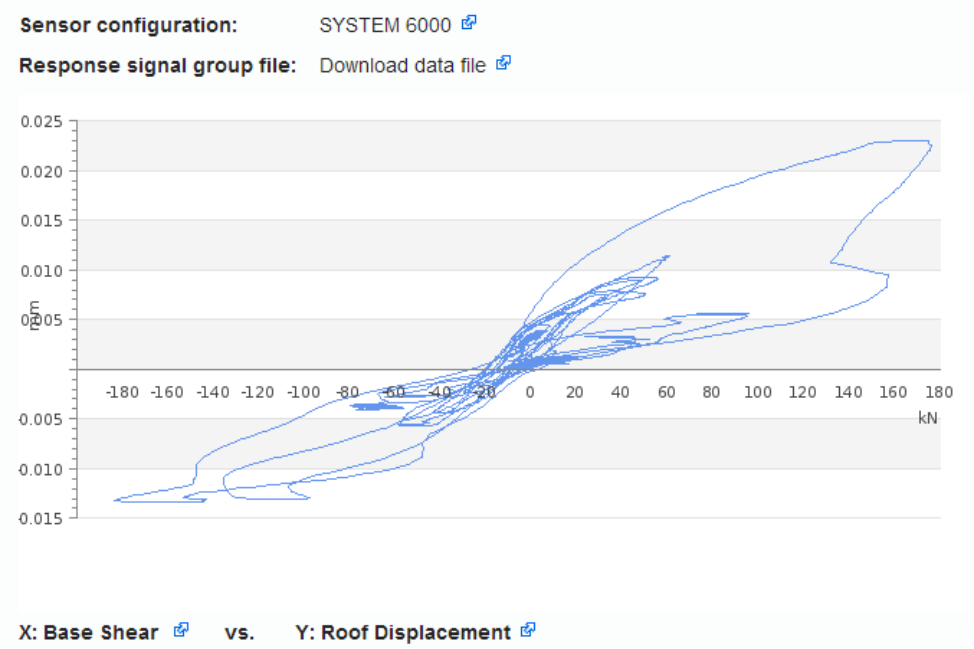

Figure 4.17: Sample plot for an experiment response signal.

is to develop a common interface for all the data that is distributed amongst many local databases created in different institutions in Europe. So that any partner in the SERIES project can have access to any other partner's data, provided that, access is not restricted to a different group of users. [24] General idea of SERIES network is shown in Figure 4.18.

It is known that, there are many ways to store the data related to an earthquake engineering study. Each implementation needs its own custom user interface to enter, manipulate or view data. Sharing data that are belonging to different databases can only be made via raw data. Since each raw data is structured in a unique way, it is very hard for any institution to view or improve this data in a meaningful manner. Exchange data format is a specification that defines an intermediate data format, so that all partner institutions can convert their own data to this intermediate data format or convert any data defined in that format to their own format. So that, institutions can view and improve other institutions' data in this way. As a result, a common knowledge database is established. [24, 1]

The structure of the database that is developed in the context of this project

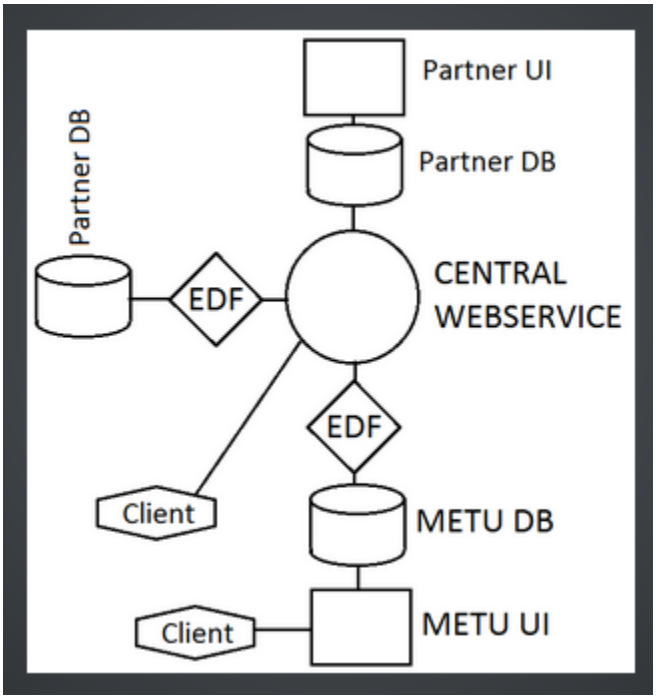

Figure 4.18: Schematic representation of SERIES network.

is very similar to the structure defined in exchange data format. Projects are defined at the top level of the data hierarchy. Specimens are tied to the projects. Experiments belong the specimens and finally signals are attached to the experiments.

Since both structures are very similar, it was very easy to implement converters. All converters are written in database level and local users of the system do not need to explicitly convert the data in the database, they are implicitly converted to the scheme of the exchange data format. The database user defined for the series central server directly accesses to the converted data scheme.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Overview

In this study, a system consisting of a database, web-based graphical user interface and some auxiliary components related with these parts are implemented in order to store and share structural and earthquake engineering experimental data. In the forthcoming section, the results and outcomes including solutions developed to overcome certain difficulties of this study will be discussed. Then the restrictions and limitations of the system will be mentioned. Lastly, possible improvements that can be done to the system will be suggested.

5.2 Discussion of Results

Currently the design and implementation of all the core requirements of the system is completed. At the end, it is seen that an SQL based database and a web-based graphical user interface is perfectly adequate to implement such a system. The specific choices on the database and web technologies, namely, MySQL and PHP played well with each other in a Linux environment. It is reported that the EDF export of the system is working well from the partners in the SERIES project.

At present, the system contains enough number of projects and related entities to evaluate all the aspects of the system. From a performance point of view, the whole system can operate on a single mid-class server computer which is not

dedicated to this system alone.

Although the database consists of 93 tables, it is observed that even the people with no special training are able to enter data since the design of the graphical interface is intuitive and minimal. This utility has been provided by using visual elements commonly and a navigation bar that can change its shape and content depending on the state. In addition, since the order of the links on the navigation bar and order of the operations that should be completed to enter the data of a project are the same, a wizard-like structure has been created. Another factor which affects users' experience positively is that the hierarchy between specimens, structural components, materials, and the hierarchy between specimens and experiments are managed by tree views.

It should be noted that, users who use the system in the viewing level easily get accustomed to the other operations, which require privileges since the data viewing and the addition/deletion/update operations are implemented on the same screens. That type of interface, which allowed developers to design both the viewing pages and manipulation pages at the same time, also accelerated the developing process.

Special attention should be paid to the introduction of structural components and formation of response signal groups. These facilities are the most considerable differences between the databases developed in this study and developed in the study carried out in the scope of the SERIES project. The introduction of the structural components made it easier to enter the specimens that are complex and the specimens that have a number of the same components. Also the response signals have been grouped and matched with the sensor device configurations; therefore, attaching signals to the experiments, the operation that requires much more time compared to the other operations, while entering data to the database took much lesser time than which was suggested by the system developed in the SERIES project.

5.3 Restrictions of the System

Although the system in general works without problems in accordance with the requirement specifications, it also has some restrictions and limitations that are caused by whether the current status of technology or the system design itself.

First of all, the .csv format, in which the data belonging to the signals, has been strictly identified. Although the researchers obtain the experiment data generally in groups, in some circumstances, these data are obtained unorganized. Some data, on the other hand, might not be separated in columns; although, they are recorded in groups. Because of the strict CVS format requirement of the system, it may be necessary to make adjustments on the format of the obtained experiment data in some circumstances. The algorithm which parses the .csv files could have been implemented in a more flexible way and these problems could have been solved; however, based on the scope of the study there found no need for such an advancement at this stage.

Although the server computer that hosts the system is enough by itself with respect to the performance, it is important to keep another server computer in reserve for the sake of the persistence of the system. Although a storage server is used and the system is backed up to external hard disks with regular intervals in order to avoid data loss in the current implementation, any malfunction in the server computer means that the system will stop running. Although there has been no malfunctioning until now, the existence of a redundant server could have quenched that concern. Preparing a redundant server to keep in reserve is a simple job; however, there has been no attempt to this so far.

Although, the only limit for the documents, images, videos and files in other types that are added to the entities is the storage capacity, there is an additional size limitation for file uploads which is forced by the PHP language in which the graphical interface has been implemented. This limit has been increased to 32 MB by the developers and it is now possible to upload many files. However, it might be the case to exceed this limit especially when adding the video files. At this point, the system developers suggest to upload the video files that are larger

than this size to another video server and to save references on the system.

Since the database that has been developed within the scope of the study is going to run in the Structural Mechanics Laboratory of METU Civil Engineering Department, it has been assumed that the entities in the laboratory belong to this laboratory only. Even if the system would have been copied as a whole and run having the same properties, it would be possible to provide service to every laboratory with a single application which runs in a single server. The current state of the system requires a separate server for each laboratory to make use of the same system because a single application can be used by only one laboratory. This means there are more than one application that need maintenance.

5.4 Possible Improvements

A number of features which are considered to be included in the system have not been passed over during the system design and the system has been designed in accordance with the possible enhancements. First of all, although the graphical interface has been implemented in accordance with the desired fundamental data which belong to all of the components, the graphical user interface can easily be developed if there will be any need to add more detailed data. Since the relations, especially inter-components relations, are kept in separate tables as far as possible, a number of one-to-one relations in the current graphical user interface can be turned into one-to-many or many-to-many relations without any change in the database.

Although the presentation of the graphical interface is sufficient for the current needs, a sort of PDF exporter can be developed to provide offline use of the data in the system. By this means, researchers can take the detailed reports of the project and components they want from the system, and use and share them in other media or mediums.

The search feature of the system supports the search operations in fundamental level. Since the database design is based on relations, the search function can be further developed.

REFERENCES

[1] A. Bosi, I. Kotinas, I. L. Martinez, S. Bousias, J. L. Chazelas, M. Dietz, M. R. Hasan, S. P. G. Madabhusi, A. Prota, T. Blakeborough, and P. Pegon. The SERIES Virtual Database: Exchange Format, Local DBs and Central Portal Interface. `http://www.series.upatras.gr/sites/default/files/SERIES_NA1_Exchange%20Format.pdf`. [Accessed October 18, 2013].

[2] R. Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011.

[3] Consortium of Organizations for Strong Motion Observation Systems. Strong-Motion Virtual Data Center Portal. `http://strongmotioncenter.org/vdc/scripts/default.plx`. [Accessed February 4, 2014].

[4] Consortium of Organizations for Strong Motion Observation Systems. VDC Fact Sheet 2014. `http://www.cosmos-eq.org/VDC/vdcfactsheet2014.pdf`. [Accessed February 4, 2014].

[5] R. Q. Coutinho, T. R. J. Oliveira, and L. M. Santos. Database of in situ test results from recife soft clays. In *Innovations and Applications in Geotechnical Site Characterization*, pages 142–154, 2000.

[6] N. Ferguson, B. Schneier, and T. Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. Wiley, New Jersey, 1st edition, 2011.

[7] I. Gashi, P. Popov, and L. Strigini. Fault tolerance via diversity for off-the-shelf products: A study with sql database servers. *Dependable and Secure Computing, IEEE Transactions on*, 4(4):280–294, 2007.

[8] P. S. Gill. *Database Management Systems*. I. K. International, New Delhi, 3rd edition, 2008.

[9] A. S. Godbole and A. S. G. A. Kahate. *Web Technologies: Tcp/ip to Internet Application Architectures*. McGraw-Hill, New Delhi, 1st edition, 2003.

[10] Y. Itoh, T. Ishiyama, C. Liu, and Y. Fukumoto. Database for structural steel experiments under a distributed collaboration environment. In *International Conference on Advances in Experimental Structural Engineering*, pages 773–780, 2005.

[11] V. K. Jain. *Database Management Systems.* Dreamtech Press, New Delhi, 1st edition, 2002.

[12] M. Levene and G. Loizou. *A Guided Tour of Relational Databases and Beyond.* Springer London, London, 2nd edition, 1999.

[13] J. S. Marcus. *Designing Wide Area Networks and Internetworks: A Practical Guide.* Addison-Wesley, Reading, 1st edition, 1999.

[14] National Institute of Standards and Technology. NIST Data Gatewayl. `http://srdata.nist.gov/gateway/`. [Accessed January 22, 2014].

[15] Network for Earthquake Engineering Simulation. NEES Project Warehouse. `https://nees.org/warehouse/all`. [Accessed December 12, 2013].

[16] Pacific Earthquake Engineering Research Center. PEER Ground Motion Database. `http://peer.berkeley.edu/products/strong_ground_motion_db.html`. [Accessed December 18, 2013].

[17] Pacific Earthquake Engineering Research Center. Structural Performance Database Database. `http://nisee.berkeley.edu/spd/`. [Accessed December 18, 2013].

[18] R. Panneerselvam. *Database Management Systems.* Prentice Hall, New Delhi, 6th edition, 2002.

[19] S. K. Rahimi and F. S. Haug. *Distributed Database Management Systems: A Practical Approach.* Wiley, New Jersey, 1st edition, 2010.

[20] S. Ram. Guest editor's introduction: heterogeneous distributed database systems. *Computer*, 24(12):7–10, 1991.

[21] Seismic Engineering Research Infrastructures for European Synergies Commission. About SERIES Project. `http://www.series.upatras.gr/`. [Accessed October 12, 2013].

[22] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts.* McGraw-Hill, Palo Alto, 6th edition, 2011.

[23] S. K. Singh. *Database Systems: Concepts, Designs and Application.* Dorling Kindersley, New Delhi, 1st edition, 2011.

[24] Team of University of Patras. 2nd version of distributed database and of data access portal including user manual, documentation and guidelines. `http://www.series.upatras.gr/sites/default/files/Deliverable_D2.5_public.pdf`. [Accessed October 18, 2013].

[25] P. Vora. *Web Application Design Patterns.* Elsevier Science, Burlington, 1st edition, 2009.