European Polytechnical Institute, Ltd. in Kunovice

Study specialization: Electronic computers

Record keeping of the Grades at the EPI, Ltd.

(Bachelor thesis)

Author: Jakub Vilímek
Head of work: Ing. Hynek Dušek

Kunovice, May 2006

Evropský polytechnický institut, s.r.o.

Osvobození 699, 686 04 Kunovice
☎ a fax: 572549018, 548035, e-mail: epi@vos.cz
http://www.vos.cz/epi

Student(ka)
Jakub Vilímek
Medinská 531
190 14 Praha 9

VÁŠ DOPIS ZNAČKY / ZE DNE	NAŠE ZNAČKA	ZODP.VEDOUCÍ/VYŘIZUJE	KUNOVICE
	BP_EP05/06	Ing. Dušek/Čápková	15.11.2005

Zadání bakalářské práce

Vážený studente, vážená studentko,

jako téma Vaší bakalářské práce ve studiu oboru Elektronické počítače Vám zadávám

Evidence známek EPI, s.r.o.

Osnova:1. Technické a programové zabezpečení projektu
2. Databáze pro evidenci známek
3. Ovládání (administrace) přes webové rozhraní
4. Projekt Evidence známek EPI: srovnání s evidencemi známek na jiných školách
5. Testování projektu, přínos pro zadavatele

Bakalářská práce bude zpracována pro: Evropský polytechnický institut, s.r.o. Kunovice

Tento dokument je součástí Vaší bakalářské práce.

S pozdravem

Ing. Oldřich Kratochvíl, Dr.h.c.
rektor

Evropský polytechnický institut
s. r. o.
Osvobození 699
686 04 KUNOVICE

I declare that I myself worked out my bachelor thesis headed by Ing. Hynek Dušek and I mentioned all used literary and technical sources in the bibliography.

Kunovice, May 2006

I would like to thank all for their beneficial methodical help and their professional findings during elaborating my bachelor thesis.

Kunovice, May 2006

Content :

INTRODUCTION

In my bachelor's graduation project I would like to focus on, as has been implied by the name already, record-keeping of the grading of the students of European Polytechnic Institute (EPI) using modern database program MySQL (My Structured Query Language). The project was initiated by the Institute itself, because so far, it has been using a simplified database built on the basis of Microsoft Excel application (see Appendix 1 for details).

The newly implemented system is supposed to not only substitute the current one, but to enrich it by several innovations and improvements as well. These include easier manipulation of the files, transparent layout, the possibility of easier access to the data from any inter- or intra-net connected computer or increased security features of the system.

In order to summarize, I could say that the primary goal of my project is the introduction of the fully functional database to a school server and PHP (Hypertext Preprocessor) programmed pages allowing the communication with such database.

Before I start focusing on particular parts of my bachelor project, I would like to explain the term database, or more precisely the database system. Database can be described as a pool of data (a set of interconnected data) + system of database management (developing tools and libraries) to manage all accesses to the database. The database system, on the other hand, includes a broader set of elements. Apart from the aforementioned pool of data and the management system also the technical resources of the database and, last but not least, the particular users of the database.

In layman's terms, the database can be described as the baggage room in any railway station, where the baggage stands for the data and the storage person impersonates the access management system, which is supposed to distribute the stored baggage. The technical resources in this comparison would be the baggage room facility (including the carts, shelves, computers or the customers' counter). It is obvious that we, as clients-users want our valuables to be stored well, available on a short notice and in required volume, and we also want the particular procedure to be carried out fast, without unnecessary delay and on reasonable budget. From this point of view it is fairly irrelevant whether the object is some valuable baggage or information in the form of 0's and 1's.

The databases can be among others specified as relational, object, deductive and others. I found out the best solution to reach the goal of the project would be to use the relational database, namely MySQL and PHP programming language (see chapters 1.2.1. and 1.2.2.). These databases are defined by two main features: a) the user assesses the data via tables and b) operations over the tables produce new tables.

Now, after having defined the basic terminology concerning databases, I would like to get to a brief summary of the structure of my paper. Its introduction part is dedicated to technical and programming resources. The next phase will feature the core of the project – projecting, creating and describing the database. Once the database will be created, there will be an effort to explain the control and administration of the database via the web interface (the description of the programmed part of the project – the PHP). In the final part I will focus on comparison with other available products, will test the program with the user and the project will be finished with summary and conclusion.

1. Technical and programming resources of the project

1.1. Technical resources

In this subchapter I would like to specify and determine the technical resources needed for the project. For the project Record-keeping system EPI, Ltd. I intend to use database system MySQL, web server Apache and PHP. In order to test effectively in the home-office environment, I had to provide for some kind of server, which substitutes the web-server based in the internet.

I used the notebook IBM X30 (lent by the company Coca-Cola Beverages) as the server. The parameters of the notebook follow:

CPU: Pentium III 800 MHz
RAM: 256 MB RAM
LAN: 100Mb
HDD: 20GB
Operating system: Microsoft Windows XP Professional

For the SQL server, web server Apache and PHP interpreter, the configuration stated above was sufficient. Launching of "server" triggered automatically the launching MySQL server as well as web-server Apache.

1.2. Software equipment

To realize the project I needed not only the hardware equipment, but also several software applications. The list of used programs and applications follows:

I used the following programs/applications or utilities.

- MySQL
- PHP
- Apache
- CSS
- PSPAD
- PhpMyAdmin
- Dbdesigner

The applications such as word, Excel or Visio are widely and commonly known and as such I won't mention them further. Products MySQL, PHP and Apache each will get a separate chapters, but the remaining PSPAD, PhpMyAdmin and Dbdesigner will be explained in one chapter, namely chapter 1.2.4 Other applications.

1.2.1. MySQL

Picture 1: Logo MySQL

Source: www.mysql.com

MySQL could be characterized as relational database system. It has been developed by the company TcX, which intended the tool for internal use at firs, but later it was able to compete with such giants of the market as Microsoft and Oracle.

The strong point of the tool is the sheer amount of data it is able to absorb without negatively influencing the performance. Another indisputable advantage is the compatibility with all the operating systems. This means that we can change the Windows into Linux and still don't have to worry about the data and functionality of database.

Yet other advantages of MySQL are: the possibility of interfering with the source codes, opportunity to use both intra and internet, sufficient security against misuse of the database, direct interfaces for C/C++, PERL, JAVA, PYTHON, PHP and/or ODBC, etc.

All of these advantages however are shadowed by the main advantageous feature of the product and that is price. Non-commercial use is for free (it is so called open source project), except for the Windows system. Even the commercial use of the product will not cost the user a fortune. Mere (relatively speaking) $ 200 will bring him or her the ability to adjust the source code thus enabling him or her to get the version with those exactly longed for features he or she desires. The authors' statement also mentions the possibility of improving the performance by up to additional 40%.

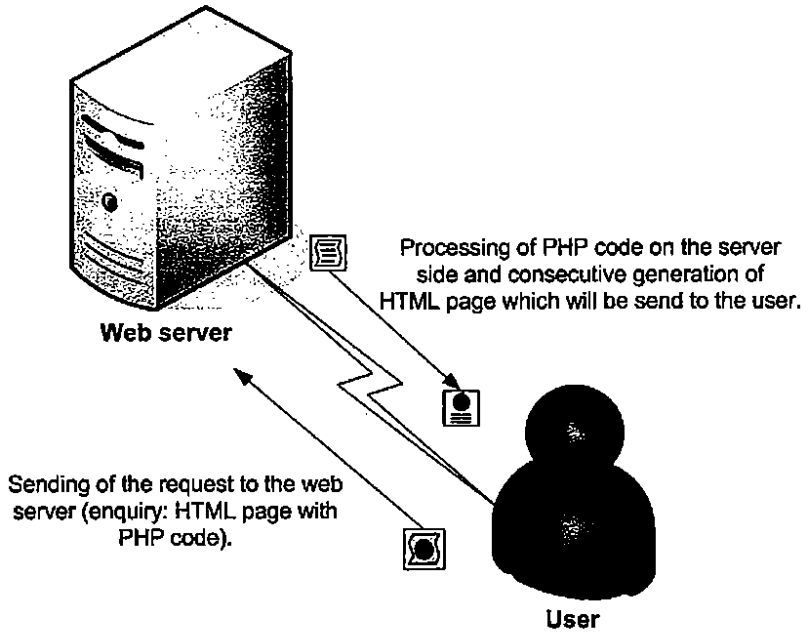
Nothing is perfect, though and even MySQL is not an exception to the rule. The tradeoff for the aforementioned advantages is the lack of transaction management, which would enable to insert a command into a transactions block. These transactions will take effect in the database only if all the commands included in the transaction get executed successfully.

However, the advantages of the product significantly outweigh the mentioned weak point explaining the popularity of the product in the market. Thus it was no wonder I decided to choose it also as my main database program for my project.

1.2.2. PHP

PHP is widely used scripting language created specifically for web and located on the server-side. This means easy programming on the server-side and allows PHP to be used to create all kinds of interactive web pages. We can insert a PHP code into our HTML (HyperText Markup Language) that will be executed on the side of the server everytime the particular page is supposed to be displayed. The code is translated by the web-server , which generates the HTML or other output to be sent to the user. The page that had been already generated by the PHP is fixed and cannot be further edited.

Picture 2: How does PHP work ?



Author: Jakub Vilímek

PHP is object oriented programming language, very similar to C++, Perl and other OOP languages. Thus, users skilled in handling the OOP programming have a distinct advantage when working with PHP. PHP belongs to languages which don't require the exact definitions of variables, plus the variables can change types. These features give the programmer a significant freedom when writing the code.

PHP was created in 1994 and originally was the creation of one man – Rasmus Lerdorf. Afterwards, several talented personnel joined his team and the system underwent three major changes. Today we are faced with high quality product I decided to use for the creation of the web application of the Record-keeping system of the EPI, Ltd.

Just like the MySQL, the PHP is the open-source product, thus allowing us to adjust, change and also further distribute the product.

1.2.3. Apache

Picture 3: Apache logo

Source: www.apache.org

Apache is simple, but efficient web-server, available for both Windows as well as Linux platform. Standard configuration of this server is sufficient for flawless running of the www server, and additional features are supported via modules.

Apache is downright perfect for creation of dynamic webpages, how does it relate to realization of this project, though?

I used Apache as the local web server, substituting it for servers anywhere in the internet. As previously explained, the PHP scripts get executed on the side of the server and so when getting opened in the regular web-browser, they produce blank output. However after the installation of Apache webserver and its PHP module, the scripts can be executed locally as well, removing the need to log in the internet, upload the scripts via FTP (File Transfer Protocol) in order to run a mere control of our work.

1.2.4. CSS

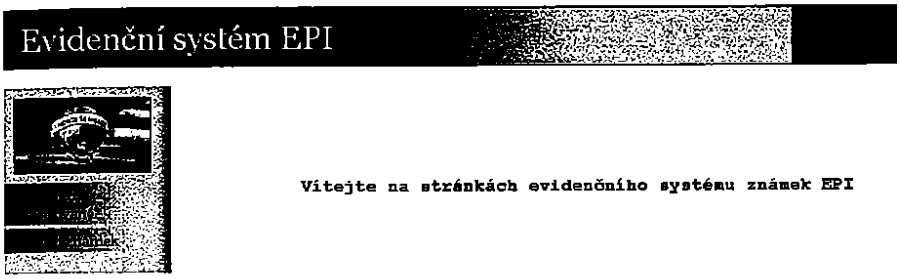
The aim of the HTML was not only defining the links between the documents, but above all the description of their structure – to determine what is the headline, what belongs to a paragraph, table, which portion is the list and which are the list's items. With growing demands raised towards HTML, the producers took care of various plug-ins and modules extending HTML way beyond its standard. These resulted in shattering of the HTML into numerous kinds, which corresponded to particular versions, even builds of the browser.

The solution of this problem are the CSS (Cascading Style Sheets) styles. Implementing and adhering to the standardized rules of the browsers allows the programmers to produce the structure of the document easily and comfortably for all the browsers there are.

Cascade styles thus interpret the formatting of the documents, define the way of presenting them to ensure uniform look despite using varying endpoint equipment, describe the look of the pages and styles of the pages' elements without interfering with the actual content of the page.

The CSS styles are used in the project mainly to simplify the structure definition in each document. Due to the fact that the project will get used for office work, I chose monotonous color scheme of the structure implementation (see picture 4)

Picture 4: HTML page using CSS styles



Author: Jakub Vilímek

The big advantage of cascading styles is, as stated above already, the possibility of handling both styles and document separately. This results in lower demand for free space.

Splitting of the presentation and structure is also profitable from the economic point of view. You will be able to find a way to reduce the overhead costs fantastically for example by reducing the size of one page from 150 KB to 25 KB. It also brings significant difference in graphic management and site maintenance – everything will be simpler. And we are referring to saving time, money and people that need to be hired.
(STANÍČEK, 2003)

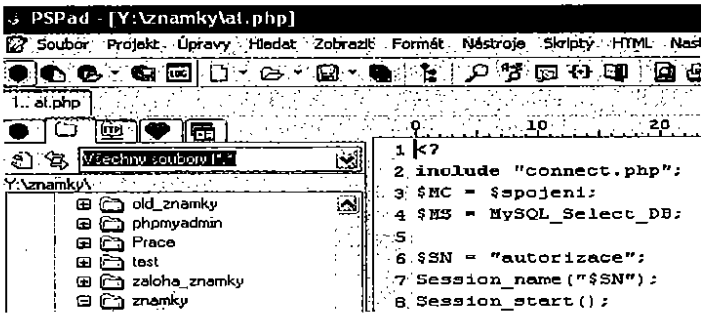
The styles are thus a perfect solutions to create and maintain design uniformity of the project. The structure is stored in one file and in order to change anything, I can do it by changing particular variables only.

1.2.5. Other applications

Here, on this place, I would like to mention other helpful software I used when creating the database or programming web administration of the project.

- The most important supporting application turned out to be the PSPad. It is a freeware universal editor, lacking the features that are redundant and even considered impeding by some users. Its strongpoints are thus simplicity, lucidity and speed. All the scripts used in the project have been written in this editor.

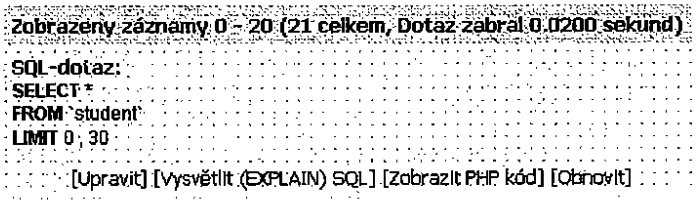
Picture 5: PSPad screenshot



Author: Jakub Vilímek

- There was another application that proved to be very helpful, PhpMyAdmin. This web application is originally intended for management of the MySQL databases. The web interface can be used to create or edit databases and tables etc. The main advantage is the speed one can utilize to check and find mistakes in SQL (Structured Query Language) queries. If there is an intent to use the SQL query, it is a good idea to test it first via this application.

Picture 6: SQL query in PhpMyAdmin



Autor: Jakub Vilímek

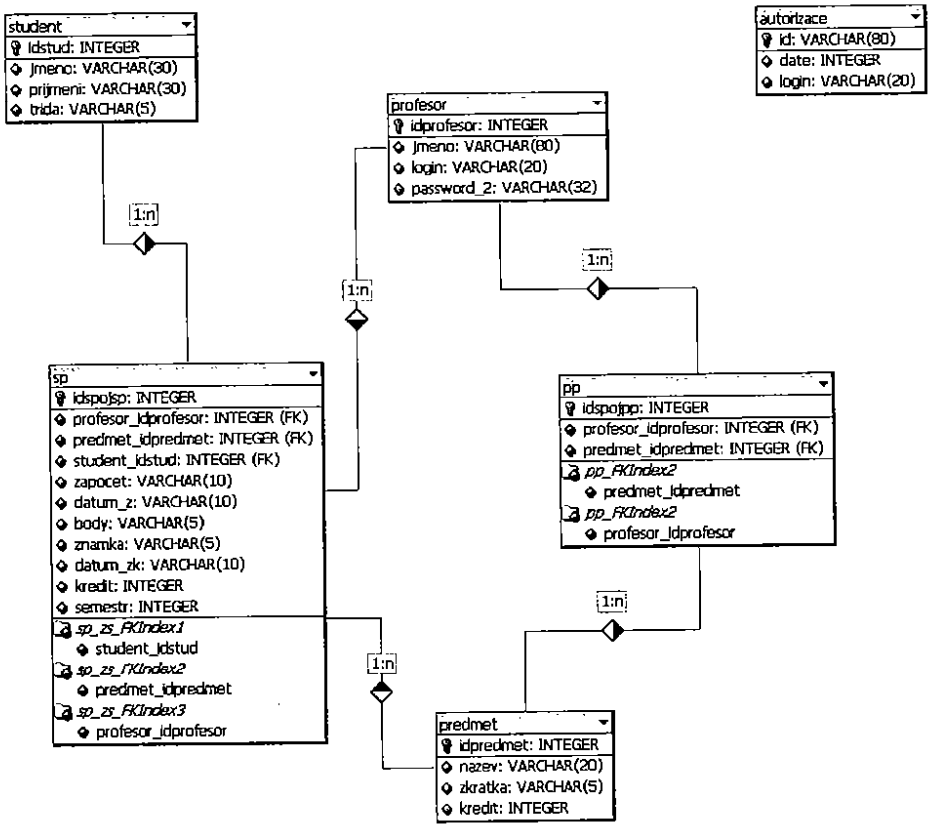
The DBdesigner application was last, not least, helper which supported me during the process of database design. It is also fast and reliable lacking redundant functions and also including useful functions, such as the possibility to connect to MySQL database and synchronize the changes. If there is no chance or will to connect, the possibility to export a SQL file is also present. The application also allows the user to create tables as well as relations among these tables.

2. Grades record keeping database

I decided to create the database using the above mentioned software product, the MySQL server. As stated above, it is very fast, reliable and high quality database system and as such is the perfect tool for the tasks of the project. Administration of MySQL was achieved not from the command line, but via PhpMyAdmin web interface.

2.1. Database design proposal

Picture 7: Database design proposal using Dbdesigner software



Author: Jakub Vilímek

When creating a database, the first step is to create a database design proposal. The bad proposal can ruin the whole project even before it starts. The database is a pillar supporting the whole building of the project and thus, if this pillar (database) collapses or doesn't function, the whole project will collapse.

Before starting, I used the Dbdesigner application to plan the creation of the design proposal. My first consideration was concerning the objects that will be entering the

database as inputs (Students, grades, teachers, classes) These objects should have dedicated tables, because it is virtually impossible to create a table that would contain the information about students as well as classes. The chosen tables must had been filled with fields. For example, the table students contains these fields: jmeno, prijmeni, trida. Each of these fields is defined by the data type (determining what will be stored in each record in the field in question). Another field is so called primary key, which identifies the records unambiquously.

2.1.1. Data types

When designing database, we must specify the data types of the particular fields. Data type is the attribute that determines what kind of data will be valid to be entered into the field. Each field can thus store only the values of one previously specified type. What follows is the list of data types I use in my database.

Numeric data types

These types allow the user to store only numbers. There are two basic types:

- Data type storing only integers
- Data type storing decimals

I am using the INT (Integer) type, which is numeric data type intended to store integer numbers. I also combine the usage of UNSIGNED modifier with this data type, barring me from entering negative numbers, effectively doubling the highest possible number.

The overview of numeric data types (integers only)

Table 1: Numeric data types

Name	Range	Memory requirement
TYNYINT	-128 - 127	1 byte
SMALLINT	-32768 - 32767	2 bytes
MEDIUMINT	-8388608 - 8388607	3 bytes
INT	-2147483648 - 2147483647	4 bytes
BIGINT	-9223372036854775808 - 9223372036854775807	8 bytes

Author: Jakub Vilímek

Text data types

These data types allow storing of text strings, or set of characters. I chose VARCHAR out of the possible range of text string data types to use throughout the project.

Data type VARCHAR is very similar to CHAR data type, but the two bear a quite significant difference. If I used the type CHAR (10) – maximum length of the field 10 characters, its size in the database would be 10 bytes no matter how many characters would actually be entered in the field. Data type VARCHAR, however, defines variable length of the field means that if only 5 characters get stored in the field, the final size would not be 10 bytes, but only 6 bytes instead. The disadvantage of the CHAR fixed length meaning ineffectively used space is leveled by higher processing speed though.

Table 2: Text data types

Name	Maximum size	Memory requirements
CHAR (X)	255 bytes	X bytes
VARCHAR (X)	255 bytes	X + 1 byte

Author: Jakub Vilímek

There obviously are also other text string data types, those, however, don't get used in the project and as such I don't mention them here.

Modifiers

Modifiers are attributes governing the behavior of the table fields.

Modifier **AUTO_INCREMENT**

This modifier gets used for INT fields. The value of such flagged field will increase by 1 after creating of each new record, making sure the values in different records will be different. The value in first record is 1. Thus, AUTO_INCREMENT is a great modifier to produce unique keys.

Modifier **PRIMARY KEY**

A field flagged with this modifier marks each record in the table uniquely. Such field must not contain duplicate values, must be always unique and must contain values that are always different. Primary key disables the modifier NULL.

Modifier **UNIQUE**

Field flagged UNIQUE must not contain duplicate values.

Modifier **BINARY**

Modifier BINARY gets used for data types CHAR and VARCHAR. If the field gets flagged by this modifier, the field will be differentiate between upper and lower case characters.

Modifier **DEFAULT**

This modifier is used to set a default value of the field. If the default value doesn't get changed by user or process, it will stay and will be stored in the field.

Modifiers **NULL** and **NOT NULL**

These specify whether the field can be stored empty (NULL), or whether it must contain a value (NOT NULL)

2.1.2. Relations

After having done the design of the tables, I needed to create the relations between them. Relations link the various tables together. What the actual kinds of relations used?

Relations 1:1

This is the simple relations between two tables, where one primary key value in one table is linked with exactly one primary key value in the other table.

Relations 1:N

This relation is between two tables again, this time one primary key values from one table is linked with several primary key values from the other table. It is also determined in one direction – if there is a 1:N relation between two tables, one of them is main and the other is a table containing more values.

Relation M:N

This is the last relation type. This relation is also between two tables, but each record from both of them can be linked to several records from the other.

2.2. Database creation

I created the database using PhpMyAdmin. There are two options to do so in the software.

1. Using the SQL command
2. PhpMyAdmin is so user-friendly it is possible to create a database with just typing in the name and clicking on one button.

SQL command to create the database :

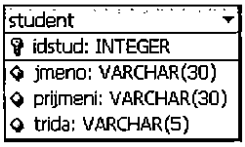
CREATE DATABASE "database_name";

2.3. Creation and description of the tables

Subchapter 2.3. Creation and description of the tables is focusing in detail on tables in the database. It defines the functionality of the table, fields as well as SQL command to create the table in the database.

2.3.1. Table *student*

Picture 8: Design proposal of the table student



Author: Jakub Vilímek

Table is intended to keep track of the students of the school. It contains 4 items. The identifier (primary key) of the table is the field *idstud*

Field *idstud* is data type INTEGER. It is supposed to identify the line, giving each line a unique number, which will not repeat in the table. Thus, we can be sure no two students will have the same identification number.

Another field, *jmeno*, is intended to store the first name of the student. Its data type was set to VARCHAR (30), thus maximum length of the first name entered is 30 characters.

Field *prijmeni* is also data type VARCHAR (30) and holds last names of the students.

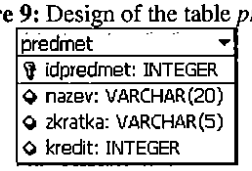
The last field of the table is *trida*. This field (as its czech name implies) serves for entering the class of the student. Because I believe 5 characters are enough for this task, I set the data type to VARCHAR (5).

SQL command to create the table *student*:

```
CREATE TABLE `student` (
`idstud` int(10) unsigned NOT NULL auto_increment,
`jmeno` varchar(30) default NULL,
`prijmeni` varchar(30) default NULL,
`trida` varchar(5) default NULL,
PRIMARY KEY (`idstud`)
)
```

2.3.2. Table *predmet*

Picture 9: Design of the table *predmet*



Author: Jakub Vilímek

This table is intended to keep track of all the classes and is also used in the process of determination, which teacher is allowed to write into which class. Table consists of 4 fields, the unique key being *idpredmet*.

Field *idpredmet* is again the identifier and also serves as the link between this table and table *pp*, where the abovementioned value "vyučující" and this ID get connected. The data type is INTEGER, indeed.

Field *nazev*, data type VARCHAR (20), contains the actual name of the class, and as such it has been limited in length to (in my opinion) sufficient 20 characters.

Field *zkratka* : For the sake of simplicity, each subject gets the abbreviation as well. This is the abbreviation, which appears in various official lists and schedules, for example. The data type of this field is VARCHAR (5)

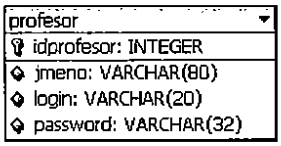
Field *kredit* is aimed at determining the amount of credits the class course "is worth" as the tracking of credits for each of the students was one of the target of the project. Data type : INTEGER

SQL command to create the table *predmet*

```
CREATE TABLE `predmet` (
 `idpredmet` int(10) unsigned NOT NULL auto_increment,
 `nazev` varchar(20) default NULL,
 `zkratka` varchar(5) default NULL,
 `kredit` int(1) NOT NULL default '0',
 PRIMARY KEY (`idpredmet`)
)
```

2.3.3. Table *profesor*

Picture 10: Design of the table *profesor*



Author: Jakub Vilímek

Another track keeping table is called *professor* is supposed to keep track of teachers who are entitled to access the system. This table consists of four fields again.

Field *idprofesor*, the unique key of the table is there to identify the teacher and also to link him or her with taught class in the table *pp*. Data type is INTEGER

Field *jmeno* is keeping track of names (both first and last) of the teacher. It serves as an identifier of whom does the particular login and password belong to for the

administrator. The data type is VARCHAR (80), allowing names of up 80 characters to be entered.

The field *login* serves to store the login name for the system. This field is one of the two fields, which get used when authenticating the user. If the user (teacher or administrator) will try to log into the system, the system will access this MySQL database and will check whether the password entered by the user matches his or her login. If yes, the user's access will be granted, if not, the access into the registered zone will be denied. Data type is VARCHAR (20). Thus *login* can have up to 20 characters.

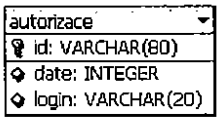
The field *password*'s data type is VARCHAR (32). 32 characters length is needed, because the passwords get stored in the database ciphered using the MD5 coding (see coding MD5). It is used, as stated when explaining the field *login*, to authenticate the users when logging in the system.

SQL command to create the table *profesor*:

```
CREATE TABLE `profesor` (
 `idprofesor` int(5) NOT NULL auto_increment,
 `jmeno` varchar(80) NOT NULL default '',
 `login` varchar(20) NOT NULL default '',
 `password` varchar(32) NOT NULL default '',
 PRIMARY KEY (`idprofesor`),
)
```

2.3.4. Table *authorization*

Picture 11: Design of the table authorization



Author: Jakub Vilímek

This table stands separately in the database and has no link to other tables. It is used to authenticate the user. When the user logs in the registered zone, he is working with secured scripts. The security is kept via session, which get stored in the ID session table and date of entering the ID.

The field *id* is for entering id session, its data type is set to VARCHAR (80). Protected scripts not only save the ID session, but use it for feedback whether the user is still logged in correctly

The field *date*, set to data type INTEGER, where the sessions save the date as the amount of seconds since 1970, as is common in PHP.

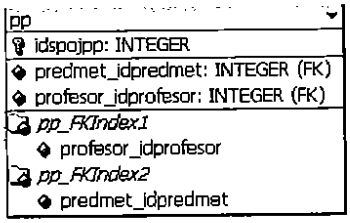
Field *login* is storing the username, which owns the particular session.

SQL command to create the table *authorization*:

```
CREATE TABLE `autorizace` (
  `id` varchar(80) NOT NULL default '',
  `date` int(10) NOT NULL default '0',
  `login` varchar(20) NOT NULL default '',
  PRIMARY KEY  (`id`),
  UNIQUE KEY `id` (`id`)
)
```

2.3.5. Table *pp*

Picture 12: Design of the table *pp*



Author: Jakub Vilímek

The table's task is to combine tables *predmet* and *professor*. Every teacher has his or her classes. The request was to allow the teacher to access and input grades into the classes belonging to him or her. This task is realized via this table, which I used to define the link between teacher and class. The primary key of this table is *idspojpp*.

Idspojpp is the primary key of this table, data type is INTEGER. The field defines the links between the tables *predmet* and *professor*.

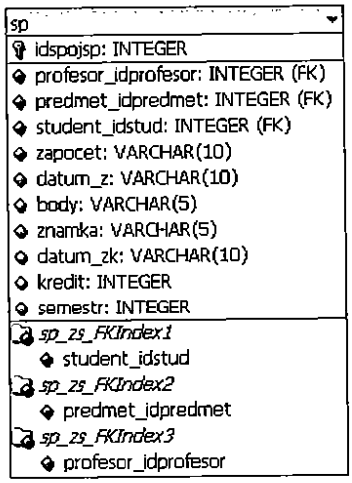
Field *predmet_idpredmet* is the field from the table *predmet*. Data type is INTEGER (just as in the original table). The same is also true for *professor_idprofesor* where the table *professor* stands for the original table.

SQL command to create the table *pp*:

```
CREATE TABLE `pp` (
  `idspojpp` int(10) unsigned NOT NULL auto_increment,
  `predmet_idpredmet` int(10) unsigned NOT NULL default '0',
  `profesor_idprofesor` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY  (`idspojpp`),
  KEY `pp_FKIndex1` (`profesor_idprofesor`),
  KEY `pp_FKIndex2` (`predmet_idpredmet`)
)
```

2.3.6. Table *sp*

Picture 13: Design of the table *sp*



Author: Jakub Vilímek

This table is created to keep track of the actual grades. The information about credits and exams ends up here. The description of the fields follows.

The table obviously does have the primary key, in this case *idspojsp*. Data type of this field is INTEGER. The field has several roles: for example, it is used as record identifier in the process of editing or deletion.

Analogous to previous tables, fields *professor_idprofesor* and *predmet_idpredmet* originate in tables *profesor* and *predmet*, respectively. The data type of both is INTEGER and they store id's of teacher and class respectively.

The field *student_idstud* contains the ID of the student, taken from the table *student*. After choosing the graded student, the database stores the entered ID, which gets later used to recognize and combine the student with his or her grades. Obviously, the data type of this field has to be INTEGER.

The field *zapocet*, data type VARCHAR (12), is the one keeping track of whether the student managed to pass the credit test or not. If he or she did, the teacher enters "ZAPOČTENO"

The field *datum_z* serves for storing the date of passing the credit test. In order to allow entering the date in format 01.01.2006, the field's data type had been set to VARCHAR (10)

I use the field *body* to keep track of the points gained by particular students. The teacher enters the point value gained by the students and such value is then stored in this field. The data type is VARCHAR (5).

The last field of the table is *znamka*. This field is not supposed to be edited, as the actual grade is based on points gained by the student. The table to determine the grade from point count follows:

Table 3: Points to grade table

Grading scale				
Numeric grade	**Letter grade**	**Verbal evaluation**	**English equivalent of the verbal grade**	**Percentage range**
1	A	excelentní	upper - excellent	95 - 100 %
2	B	výborný	lower - excellent	89 – 94 %
2	C	velmi dobrý	very good	83 – 88 %
3	D	dobrý	good	77 – 82 %
3	E	dostatečný	sufficient	71 – 76 %
4	F	nevyhovující	fail	70 % a méně

Source: European polytechnic institute

Just like the previous date field, *datum_zk* stores the date of the exam. And just like the field datum_z, this field's data type is VARCHAR (10) as well.

Field *kredit* files the amount of credits gained by the student. If the student passes the exam or credit test, the credit value of the passed class is in this field. If the student fails the test / exam, value stored is 0. This means the data type has to be INTEGER

Field *semester* defines whether the record about the credit / exam was taken in winter or summer semester. To minimize the room for error, I decided to offer two options, 1 for winter and 2 for summer semester, thus the data type of the field is INTEGER.

SQL command to create the table *sp*:

```
CREATE TABLE `sp` (
 `idspojsp` int(10) unsigned NOT NULL auto_increment,
 `student_idstud` int(10) unsigned NOT NULL default '0',
 `predmet_idpredmet` int(10) unsigned NOT NULL default '0',
 `zapocet` varchar(12) default NULL,
 `datum_z` varchar(10) default NULL,
 `body` varchar(5) default NULL,
 `znamka` varchar(5) default NULL,
 `datum_zk` varchar(10) default NULL,
 `podpis` varchar(20) NOT NULL default '',
 `kredit` int(1) NOT NULL default '0',
 `semestr` int(1) NOT NULL default '0',
 PRIMARY KEY (`idspojsp`),
 KEY `sp_FKIndex1` (`predmet_idpredmet`),
 KEY `sp_FKIndex2` (`student_idstud`)
)
```

3. Control and administration via the web interface

I designed the database and now I get to another indispensable part of my project. I am talking about the administration of the database using the web interface, as it is almost impossible to create and maintain several account for teachers to access the database and input their grading directly from the command line of MySQL.

My "building material" turned out to be PHP. It proved to be just the perfect product, because of its perfected security features. The actual work, the processing of the scripts, is executed on the server side and thus, if the scripts are run on a well-protected server, there is no way for unauthorized personnel to access the data in the scripts.

Just like I did when describing the database, I would like to focus more on this part, namely on both script organization (directory structure) and detailed analysis of the important parts of the code.

I will now describe the directory structure of the application. I created 3 directories plus root one (including e.g. index.php). The project is divided into secure and insecure zones, I decided to divide the scripts along the same line for the sake of transparency.

For grades reporting, going into insecure zone, I use the directory report. In this directory, there are 6 scripts, tasked with creation of tabulated grades reports for both students and teachers.

The other two directories belong into the secure zone, on the other hand. To get into that zone, the user must log in using his username and password. The names of these directories are *spravce* and *znamky*. The difference is the level of access allowed for various users. The scripts to administrate the tables (the database) are available for users with administrator verification. Teacher's verification allows the user to access the directory znamky, which contains scripts for complex editing and administration of the grades.

The list of scripts in directories:

Root directory contains these scripts:

- index.php
- autorizace.php
- spravce.php
- profesor.php
- connect.php
- at.php

Report directory contains scripts:

- report.php
- report_01.php
- report_02.php
- report_03.php
- report_04.php
- report_05.php

Directory znamky contains scripts:

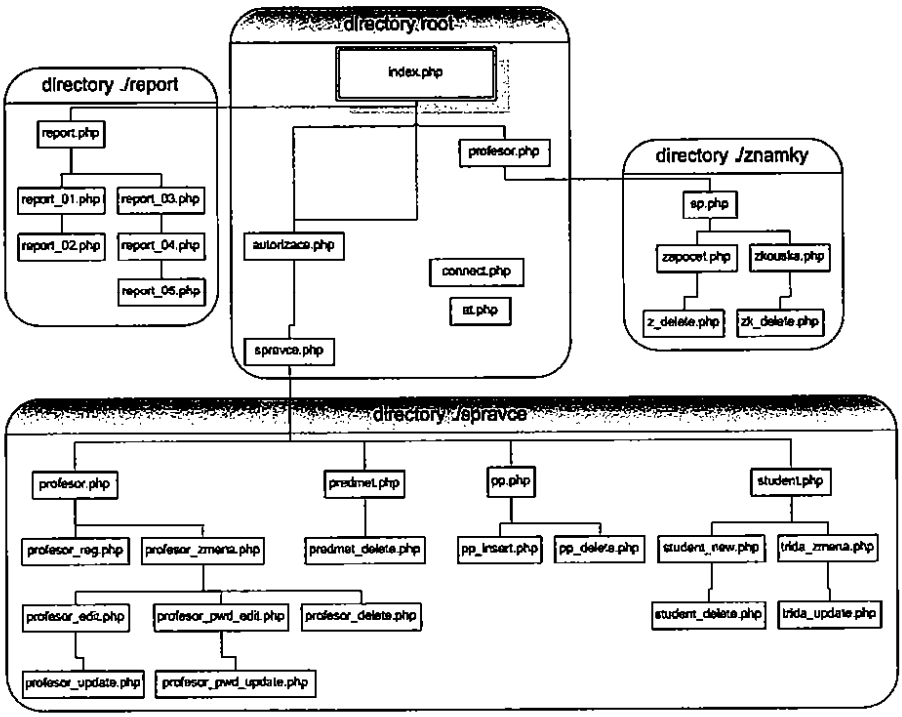
- sp.php
- zapocet.php
- z_delete.php
- zkouska.php
- zk_delete.php

Directory spravce contains scripts:

- profesor.php
- profesor_reg.php
- profesor_zmena.php
- profesor_edit.php
- profesor_update.php
- profesor_pwd_edit.php
- profesor_pwd_update.php
- profesor_delete.php
- predmet.php
- predmet_delete.php
- pp.php
- pp_insert.php
- pp_delete.php
- student.php
- student_new.php
- student_delete.php
- trida_zmena.php
- trida_update.php

Following picture shows the directory structure showing which scripts cooperates with which script. The structure is built along the directory structure.

Picture 14: Direktory structure

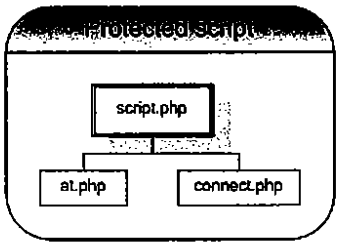


Author: Jakub Vilímek

As you have certainly noticed, the scripts connect.php and at.php are not shown in the picture as having the connection. The truth is opposite – they have so many connections it would be obstructing the clarity of the overview. I will nevertheless describe their connections with "random" scripts.

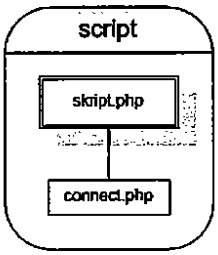
Pictures 15 and 16 show two different scripts. The first one is secured, located in secured zone, while the other is outside of the secured zone, but is connected to the script connect.php and thus is using the MySQL connection to the database.

Picture 15: Protected script



Author: Jakub Vilímek

Picture 16: Script using MySQL connection



Autor: Jakub Vilímek

This concludes the part about listing of the scripts, directory structure and description of both connect.php and at.php scripts.

3.1. Description of the scripts

index.php

The very first script is obviously index.php. In this script there are not any PHP commands, because the whole script is written in HTML. The page is intended to welcome the user in the system and offer them the following options:

- Enter the protected zone – link to script that takes care of the log in the system.
- Grades overview (reporting) – link into unregistered zone, where all the students' grades can be viewed.

connect.php

The script connect.php is inserted into each script using the **include** command. It is a configuration file for the connection to the MySQL database I designed. In order not to have to define the username, password and database for each individual query, we have all the information stored in one script, which, in turn, gets connected to other scripts. Such a

configuration is very advantegous in situations such as moving the whole system to another server, where the login information may be different.

Variables
$db_server defines MySQL database server
$db_login defines the username for the database
$db_password defines password to the database
$db_name defines the name of the database, into which we want to connect
@$spojeni creates the connection with MySQL server
The last variable also determines in which database we will operate.

```
<?
$db_server = "localhost";
$db_login = "root";
$db_password = "mysql";
$db_name = "znamky";
@$spojeni = MySQL_Connect($db_server ,$db_login, $db_password);
@MySQL_Select_DB($db_name);
?>
```

autorizace.php

Script is absolutely essential for the project, because it governs and decides whether the access of particular users is authorized or not

This script will first display a form to enter the information, because previous conditions have not been met before. After inputting the requested information the system checks in the table profesor whether both username an password are entered correctly. If it is so, the name gets set via Session_name, the session then gets started using the session_start and its ID is stored in the variable $sid. This variable is finally saved into the table autorizace together with number of seconds since 1970.

The condition for variables $login and $password to grant access. As we can see, the condition is counting all the records with set username and password. If the result is not equal to 1, the condition is met and the access is not granted, because if number of such records is less then 1 (it is 0) then the user does not exist and if condition result is more than 1, there are several users with same username and password, which is not allowed. In such case (access not granted), the condition will send the user back to login, where the whole process can start anew.

```
If ((IsSet($login)) AND (IsSet($password))):
$p = MD5($password);
$MSQ = MySQL_Query("SELECT * FROM profesor WHERE (login LIKE '$login')
AND (password LIKE '$p')");

If (MySQL_Num_Rows($MSQ) <> 1):
echo "Neautorizovaný přístup";
echo "<br>";
echo "<br>";
echo "<td><a href=\"autorizace.php\">Prosím přihlašte se!</a><td>";
Exit;
```

If the condition is not met (such situation is actually desired), it continues to processing next actions – starting and saving the session as described above.

```
Else:
$SN = "autorizace";
Session_name("$SN");
Session_start();
$sid = Session_id();
$time = Date("U");
$at = Date("U") - 1800;
$lg = $login;

$MSQ = MySQL_Query("INSERT INTO autorizace VALUES ('$sid', $time, '$lg')");
$MSQ = MySQL_Query("DELETE FROM autorizace WHERE time < $at");
Endif;
```

If this action successfully executes and the condition is still not met, the script goes on to next phase, where the protected page gets displayed already.

This part of the code decides whether the user is administrator or regular teacher. If he or she is an admin, to be exact, his or her login is one of an administrator, he or she will get transferred automatically to script spravce.php. If the login does not match the one of an administrator meaning the user is a regular teacher, the transfer is targeted to the script profesor.php

```
<? if ($login=="spravce"): ?>
  <meta http-equiv="refresh" content="0; url=spravce.php">
 <?
                exit;
 endif;
 ?>

 <? if ($login<>"spravce"): ?>
 <meta http-equiv="refresh" content="0; url=profesor.php">
  <?
                exit;
 endif;
 ?>
```

The last condition included in this script is the condition of a logout. If we want to log the user out of the system, we must distribute this link into all the scripts:

```
<a href="autorizace.php?lo=true">
```

The result of this link is the execution of a condition which deletes the record in the table autorizace and thus logout of the system.

The form for entering username and password:

```
<form action="autorizace.php" method="post">
<input type="Text" name="login">
<br>
<input type="Password" name="password">
<br>
<input type="Submit" value="Log On">
</form>
```

So, the authorization seems to have been solved now, but that is true only for the main page. In order not to have only one page secured but all of them instead, we must add a code to the top of each script. To save me the efforts of copying and pasting or typing of the same portion of the code, I saved it in one script, called at.php, which is added using the command **include** to the top of each script I intend to protect from being accessed without proper authorization.

at.php

As mentioned above, the script at.php is another verification script. Just like connect.php, at.php gets added to the top of any other script using the **include** command. It protects all the other scripts from unauthorized access.

```
<?
include "connect.php";
$MC = $spojeni;
$MS = MySQL_Select_DB;

$SN = "autorizace";
Session_name("$SN");
Session_start();
$sid = Session_id();
$date = Date("U");
$ad = Date("U") - 300;

$dotaz = "SELECT * FROM autorizace WHERE (id = '$sid') AND (date >= $ad)";
    $MSQ = MySQL_Query($dotaz);
    $query = mysql_query($dotaz);
    $radky = mysql_fetch_array($query);

If (MySQL_Num_Rows($MSQ) <> 1):
echo "Neautorizovaný přístup";
Exit;
Else:
$MSQ = MySQL_Query("UPDATE autorizace SET date = $date WHERE id = '$sid'");
Endif;
MySQL_Close($MC);
?>
```

The way the script is running is very similar to the script autorizace.php. Setting of the session_name, followed by execution of the session by the command session_start and saving the id session into the variable $sid. As the next step, the script determines, whether the id exists in the autorizace table. If there is no other script, we terminate using the command **exit**, if there is one, the time of the last action in the table gets actualized.

spravce.php

If we log in as system administrator, we get to the special area of the project for administrators. In this area, there is the administration of the tables student, predmet, profesor and pp. The page is written by the HTML, because it contains only the links to scripts in the spravce directory. The most significant functional difference against the previous scripts is the protection to deny access to the scripts for non-admin users.

The code will be entered in the beginning of each script intended for administrators only.

```
<?
include "connect.php";
include "at.php";
$lg = $radky['login'];
?>
<? if ($lg=="spravce"): ?>
```

As is clearly visible, the decision is based on the condition login = spravce. If it is met, everything, what is mentioned between the start and end lines of the condition will get displayed. The code is always at the end of the script intended for login = spravce.

```
<?
 exit;
 endif;
?>
<? if ($lg<>"spravce"): ?>
<html>
<head>
 <meta http-equiv="refresh" content="3; url=autorizace.php?lo=true">
</head>
<body>
Vstoupil jste do zóny, která vám není zpřístupněna. Budete automaticky odhlášen.
</body>
</html>
<?
                exit;
 endif;
?>
```

3.1.1. Scripts for user admin (spravce)

Scripts belonging to the administration of students category:

- student.php
- student_new.php
- student_delete.php
- trida_zmena.php
- trida_update.php

student.php

If we select the option Administrace studentů, we get to the script student.php. This script is not important so far, we just enter the classroom we want to be inputting and will get through to another script (student_new.php). I set up $trt as the variable of the classroom

There is another interesting option in this page, the one to the script trida_zmena.php. This link is named Hromadná změna třídy. This script allows the user to choose any classroom and re-assign all the students to a different classroom. The original intent of this script was advancement over the grades.

student_new.php

This script inserts and edits records and allows the creation of a new record as well. The functions INSERT and UPDATE are encountered right at the start next to conditions, that cannot be met during the first execution. These conditions will get fulfilled when the form with variables gets filled and sent.

The condition to insert a new student. If variable insert equals the letter "a", the SQL command INSERT gets executed, therefore the new student is inserted. The variable insert is sent from the form of new student entry already.

```
<?
include "../connect.php";
$MC = $spojeni;
$MS = MySQL_Select_DB;
If ($insert == "a") {

mysql_query("INSERT INTO `student` ( `jmeno`, `prijmeni` , `trida` )
                    VALUES ('$jmeno', '$prijmeni', '$trt');"
   ,$spojeni) or die("Nepodařilo se vložit");
   }
```

This condition serves for changing records, several of them to be exact. To realize this function, there is a cycle. Because we can (and probably will) require changing several records, only one SQL query would not be sufficient, thus we are using a cycle which is keeps working as long as there are unchanged records.

```
if (!empty($tlac_zmena)) {
    $pocet_zazn=count($id);
    for ($i=0;$i<=$pocet_zazn-1;$i++) {
       $dotaz = "UPDATE student SET jmeno = \"$jmeno[$i]\" , prijmeni =
\"$prijmeni[$i]\", trida = \"$trida[$i]\" WHERE idstud = $id[$i]";
       mysql_query($dotaz,$spojeni)
          or die("<center><b>Nepodařilo se upravit</b></center>");
          }
 }
?>
```

Form for entering the new student. The last <input> specifies the value needed to recognize the need to create new records in the condition intended to do so. The value is set to "a", the same as has been specified in the condition. The tag <input> has been set as hidden, meaning this input field is kept from being seen and thus edited. The classroom is the same as chosen in the previous script (student.php) and inserted into the <input> tag value: <? echo($trt)?>. In order not to lose the trt value, it is inserted into the link when sending the form.

```
<form action="student_new.php?trt=<? echo $trt;?>" method="post">
<font color = "darkblue"><center><h3>Formulář pro vložení nového
studenta</h3><center></font>
<table align="center">
<tr>

<tr>
<td>Jméno : </td><td><input type="text" name="jmeno" size="20" value=""></td>
</tr>
<td>Příjmení : </td><td><input type="text" name="prijmeni" size="20" value=""></td>
</tr>
<td>Třída : </td><td><input type="text" name="trt" size="5" value="<?
echo($trt)?>"></td>
<td><input type="hidden" name="insert" size="1" value="a"></td>
</tr>

<tr>
<td colspan="2" align="center"><input type="submit" value="Odešli formulář"></td>
</tr>
</table>
</form>
```

Another form is the actual selection of the data from the database and subsequent possible editing of those. To do so, we are using the part of this code:

```
<form action="student_new.php?trt=<? echo $trt;?>" enctype="multipart/form-data"
method="post" />
 <table border=1>
 <tr bgcolor=#d3d3d3 align=center
valign=middle><td></td><td>Jméno</td><td>Příjmení</td><td>Třída</td></tr>
 <?
```

```
$vysledek=mysql_query("SELECT *
FROM student
WHERE trida = '$trt'
ORDER  BY student.idstud",$spojeni)
    or die("Nepodařilo se vykonat příkaz");

for ($i=0;$i<=mysql_num_rows($vysledek)-1;$i++) {
   echo "<tr align=center valign=middle>
      <td>
      <input type=\"hidden\" name=\"id[]\"
         value=\"".mysql_result($vysledek,$i,"idstud")."\" ></td>
      <td>
      <input size=\"10\" type=\"text\" name=\"jmeno[]\"
         value=\"".mysql_result($vysledek,$i,"jmeno")."\" ></td>
      <td><input size=\"8\" type=\"text\" name=\"prijmeni[]\"
         value=\"".mysql_result($vysledek,$i,"prijmeni")."\" ></td>
      <td><input size=\"8\" type=\"text\" name=\"trida[]\"
         value=\"".mysql_result($vysledek,$i,"trida")."\" ></td>";

   echo                                                                    "<td><a
href=\"student_delete.php?id=".mysql_result($vysledek,$i,"idstud")."&trt=$trt\">x</a></t
d>";
  }
 ?>
 </table>
 <table>
 <tr><td   colspan="6"><input   name="tlac_zmena"   type="submit"   value="Změna"
/></td></tr>
 </table>
 </form>
```

The first thing we encounter in the code is the SQL query. It extracts all the data from the table student meeting the condition of having the same value in the field trida as we had chosen in the beginning. This selection is defined via WHERE and is being used for the sake of clarity and transparency, after all, listing all possible students could get quite messy.

What should there be done to display the query, though? There is a cycle beginning after the SQL query, which tackles this task – it executes the defined setting for one line of the desired table for the number of times equal to the forecasted number of lines given by the query.

You might notice there is a link in the cycle. This link deletes a line in the table. Because this link is included in the cycle, it gets displayed for each line. This is achieved using the id of the line.

Student_delete.php

If we want to delete a certain record in the script student_new.php, all it takes is a single click on the X link, which will take us to the script student_delete.php and carries the necessary information in its url to successfully remove the record in question. Thus for deletion of the records, this student_delete.php script included shall be used.

```
<?php
include "../connect.php";

$MC = $spojeni;
$MS = MySQL_Select_DB;

 $dotaz = "DELETE FROM `student` WHERE idstud = $id";

 $vysl = mysql_query($dotaz);

 echo("<center><b>Data byla úspěšně vymazána</b></center>");

?>
```

There is a DELETE command in this code, deleting the records based on condition idstud=$id, where id is the variable transferred via the URL as the information about the record to be deleted.

Let's go back to the script student.php, where there is, as stated above, the link to mass change the classroom. It will take us to the script trida_zmena.php

trida_zmena.php

There are only two listboxes to be found on this page, which contain all the classroom of the EPI Kunovice. The first listbox bears the name of trt_old variable – this is the one to govern the selection query. The other listbox bears the name for trt_new, which is the variable to be placed behind the previous classroom.

The beginning of the listbox with the name trt_old

```
<td>Zvolte aktuální třídu: </td><td><select name='trt_old' size="1">
<option value="1EI">1EI - EPI</option>
<option value="1EII">1EII - EPI</option>
<option value="1EP">1EP - EPI</option>
<option value="1EPi">1EPi - EPI</option>
```

The beginning of the listbox with the name trt_new

```
<td>Zvolte budoucí třídu: </td><td><select name='trt_new' size="1">
<option value="1EI">1EI - EPI</option>
<option value="1EII">1EII - EPI</option>
<option value="1EP">1EP - EPI</option>
<option value="1EPi">1EPi - EPI</option>
```

After the selection of the classroom, we can continue by pressing the button to the next script to process the information, namely trida_update.php

trida_update.php

The code processes the information I send from the script trida_zmena.php and implements the requested changes in the MySQL database in appropriate table.

```
<?php
include "../connect.php";

$MC = $spojeni;
$MS = MySQL_Select_DB;

 $dotaz = "UPDATE student SET trida = \"$trt_new\"  WHERE trida = \"$trt_old\" ";

 $vysl = mysql_query($dotaz);
 echo("<center><b>Data byla úspěšně upravena</b></center>");
?>
```

I used the SQL command update with the condition trida = \"$trt_old\". This condition is supposed to select the records that have the value trida equal to the value set in the first listbox of the script trida_zmena.php – current classroom. The classroom value of these records has to be set to the value from the second trida_zmena.php listbox. For such a task, the SQL command SET = \"$trt_new\". All the selected records will be changed into the classroom we chose.

Scripts belonging to the class administration category

- predmet.php
- predmet_delete.php

The next in line after students administration, the classes administration comes up (In the script spravce.php) The scripts for the classes administration are basically the same, varying only in descriptions and SQL queries syntaxes.

predmet.php

At the beginning of the script there are conditions again for the execution of entering the record or change thereof. As stated above, the difference is only in the SQL queries

SQL query for the table predmet enters new records from the form.

```
mysql_query("INSERT INTO `predmet` ( `nazev`, `zkratka`, `kredit` )
                    VALUES ('$nazev', '$zkratka', '$kredit');"
   ,$spojeni) or die("Nepodařilo se vložit");
```

Another SQL query there is serves as record-changing one.

```
if (!empty($tlac_zmena)) {
   $pocet_zazn=count($id);
   for ($i=0;$i<=$pocet_zazn-1;$i++) {
      $dotaz = "UPDATE predmet SET nazev = \"$nazev[$i]\", zkratka = \"$zkratka[$i]\",
kredit = \"$kredit[$i]\"  WHERE idpredmet = $id[$i]";
```

```
    mysql_query($dotaz,$spojeni)
      or die("<center><b>Nepodařilo se upravit</b></center>");

  }
}
```

The forms for these queries are the same as the ones for the student_new.php script, so I deem it pointless to show them again. The only difference is the data we require, as is obvious from looking at the SQL queries. Just like the script student_new, this one is equally equipped with the deletion of the record. This is achieved using the link to a script called predmet_delete.php

```
<a href=\"predmet_delete.php?id=".mysql_result($vysledek,$i,"idpredmet")."\">x</a>
```

predmet_delete.php

SQL command for deletion of the record in the table:

```
<?php
include "../connect.php";

$MC = $spojeni;
$MS = MySQL_Select_DB;

 $dotaz = "DELETE FROM `predmet` WHERE idpredmet = $id";

 $vysl = mysql_query($dotaz);

 echo("<center><b>Data byla úspěšně vymazána</b></center>");

?>
```

Scripts belonging into the category classes administration:

- profesor.php
- profesor_reg.php
- profesor_zmena.php
- profesor_edit.php
- profesor_update.php
- profesor_pwd_edit.php
- profesor_pwd_update.php
- profesor_delete.php
- pp.php
- pp_insert.php
- pp_delete.php

The last administration part to go is the user administration. This part is made up by entering a new user (done via profesor_reg.php), change of the users' details (there are several scripts for this task – profesor_zmena.php, profesor_edit.php, profesor_update.php, profesor_pwd_edir.php and profesor_pwd_update.php). The last action we have to do is the connection between teacher and class. It needs to be created in order to determine

which teacher is allowed to input grades to which class. The scripts used in the process are pp.php, pp_insert.php and pp_delete.php.

The link "Administrace uživatelů" will take us to the script profesor.php. Don't confuse this script with the script profesor.php located in the root directory of the project. This profesor.php is located in the directory spravce. In the script there are links to subgroups I have already mentioned above. Thus I start with entering a new user.

profesor_reg.php

There is a condition for entering a new user right at the beginning of the scrip. On the first execution, the condition will obviously be not met, because we didn't provide it with needed information. This condition is implemented to guard the existence of particular user. If the user exists, the condition will stop, if not, it will create a new user into the table profesor with the data from the form. There is a notable information about this script and that is ciphering of the password using MD5.

```
<?
include "../connect.php";
$MC = $spojeni;
$MS = MySQL_Select_DB;

If (($jmeno != "") AND ($login != "") AND ($password != "")):
$MSQ = MySQL_Query("SELECT * FROM profesor WHERE login LIKE '$login'");
If (MySQL_Num_Rows($MSQ) > 0):
$login = "";
$error = "<h4>Login již existuje!!!</h4>";
$f = "true";
Else:
$p = MD5($password);
$MSQ = MySQL_Query("INSERT INTO profesor VALUES (NULL, '$jmeno', '$login',
'$p')");
$m = "true";
$x = StrLen($password);
Endif;

Elseif (IsSet($send)):
$error = "<h4>Chybí povinné údaje!!!</h4>";
Endif;
?>
```

The form to enter the user data is the same as other forms, thus I don't think it is necessary to be shown here, but there is one more script here allowing the printing of the login data for the user in case of correct entering and processing of the data. As I have noted, if the condition of correct entering is met, the variable m gets set to "true" and printing can begin.

```
<?Elseif (IsSet($m)):?>
Uživatel byl úspěšně zaregistrován:<br><br>
<b>Jméno</b>: <?echo $jmeno;?><br>
<b>Přihlašovací jméno</b>: <?echo $login;?><br>
```

```
<b>Heslo</b>: <? echo $password;?><br>
<?Endif;?>
```

I have analyzed the script profesor_req.php. The next up should be the profesor_zmena.php profesor_zmena.php, profesor_edit.php, profesor_update.php, profesor_pwd_edir.php and profesor_pwd_update.php, but those are very similar to the ones I had gone through already, thus I don't see a reason to further describe them.

Let's move on to the scripts pp.php, pp_insert.php and pp_delete.php. These are also very similar to other scripts, though there two listboxes showing SQL query from the tables profesor and predmet. I have already written that table pp serves to link together the teacher and class.

pp.php

Here I would like to demonstrate the first listbox, in which I set up a list of all the users from the table profesor in order to select the teacher.

```
<select name='profesor' size="1">
<?php
include "../connect.php";
$MC = $spojeni;
$MS = MySQL_Select_DB;

 $dotaz = "SELECT idprofesor, jmeno
 FROM profesor";
 $query = mysql_query($dotaz);
for ($i=0;$i<mysql_num_rows($query);$i++){
 $radky=mysql_fetch_array($query);

?>

<option value="<?echo $radky['idprofesor'];?>">
 <?php echo $radky['jmeno'];?>
 </option>
<?
 }
?>
</select>
```

The SQL query will select us some records we need to display afterwards. Primary key of the table, idprofesor is always set up as the value, though in order to understand who is the current teacher, the names have to be assigned.

The other listbox is principally same as the previous one, but backed by different SQL query to selecting from the table predmet. This time I chose to display two values, namely the class abbreviation and full name

```
<option value="<?echo $radky['idpredmet'];?>">
 <?php echo $radky['zkratka'];?>
  -
 <?php echo $radky['nazev'];?>
 </option>
```

This script – and it was the first to do so – used the SQL query from several tables. In the editing table, there is a query based on tables predmet, profesor and pp. As you can see, I linked this query with tables predmet and profesor, because I want to display users' names and full class names instead of abstract numbers from the class ID.

```
 $dotaz = "SELECT pp.idspojpp, profesor.jmeno, predmet.nazev
FROM profesor, predmet, pp
WHERE predmet.idpredmet = pp.predmet_idpredmet and profesor.idprofesor =
pp.profesor_idprofesor
ORDER BY profesor.jmeno";
```

Then come two scripts, very similar to those previously explained. These scripts are pp_insert and pp_delete, which contain SQL commands insert and delete respectively.

This could be it for the spravce login scripts and now I will get to th scripts used by teachers' login

3.1.2. Scripts for user teacher (profesor)

Scripts belonging to the grades category

- sp.php
- zapocet.php
- z_delete.php
- zkouska.php
- zk_delete.php

How did it all get started? So we login as the teacher user. This means we will be using different login than spravce. As I mentioned in the beginning, there is a condition forwarding us to the script professor.php

profesor.php

There is a sort of selection questionnaire for the teacher in the script profesor.php. There first listbox allows the teacher to select the class he or she wants to input the grade to. Here I would like to spend some time explaining the input of values for further processing. As there was a request for record keeping of the credit values of the classes as well, I needed to solve in an easy way sending and processing this information. The solution is in the field value. The two bits of information are separated by "_" It is supposed to determine how to divide the information later, as they are about to be sent as one value.

```
<select name='prt' size="1">
<?php
include "connect.php";
$MC = $spojeni;
$MS = MySQL_Select_DB;

 $dotaz = "SELECT pp.predmet_idpredmet, predmet.nazev, predmet.zkratka,
predmet.kredit, profesor.jmeno
FROM pp, predmet, profesor
WHERE pp.predmet_idpredmet = predmet.idpredmet
AND pp.profesor_idprofesor = profesor.idprofesor
AND profesor.login = '$lg'";

 $query = mysql_query($dotaz);
 for ($i=0;$i<mysql_num_rows($query);$i++){
 $radky=mysql_fetch_array($query);
?>
<option value="<?echo $radky['predmet_idpredmet'];?>_<?echo $radky['kredit'];?>">
 <?php echo $radky['zkratka'];?>
 -
 <?php echo $radky['nazev'];?>
 </option>
<?
 }
?>
</select>
```

Yet another selection there is, chooses the classroom to be graded. This selection is followed by the choice of requested semester.

```
<td>Semestr: </td><td><select name='smr' size="1">
<option value="1">Zimní semestr</option>
<option value="2">Letní semestr</option>
</select>
```

The last choice here is the option whether the class is completed by a credit test or an exam.

```
<td>Zápočet : </td><td><input type="radio" name="bod" value="z"></td>
</tr>
<tr>
<td>Zápočet, Zkouška : </td><td><input type="radio" name="bod" value="zk"></td>
```

After sending the form, we get to the script sp.php, which in turn, will take us to either script zkouska.php or zapocet.php based on selection made in the listbox.

```
 <? if ($bod=="z"): ?>
  <meta http-equiv="refresh" content="0; url=zapocet.php?pret=<? echo $prt;?>&smr=<?
echo $smr;?>&trt=<? echo $trt;?>">
 <?exit;
```

```
 endif;
 ?>

 <? if ($bod=="zk"): ?>
 <meta http-equiv="refresh" content="0; url=zkouska.php?pret=<? echo $prt;?>&smr=<?
echo $smr;?>&trt=<? echo $trt;?>">
  <?exit;
 endif;
 ?>
```

The two scripts (zapocet.php and zkouska.php) are the same, only zapocet.php has some removed fields that contain data relevant only for exams. That's why I will describe only the full input script, zkouska.php

zkouska.php

I am sure you remember how I sent two values in one in the script professor.php (the one for teachers). There is one php function right at the beginning splitting these two values apart again. The function is explode, which with divider set to the underscore mark ("_"). The variable $pret gets thus separated back into two needed variables, $prt (class id) and $kredit (credit value of the class)

```
<?
$explo = explode("_", $pret);
$prt = &$explo[0];
$kredit =&$explo[1];
?>
```

There is a condition's turn after the splitting of the values. The first one to go is the determination whether the student has already the record for the particular class. The condition is listed below :

```
If ($insert == "a") {

$test = MySQL_Query("SELECT * FROM sp WHERE student_idstud = '$jmeno' AND
predmet_idpredmet = '$prt'");
If (MySQL_Num_Rows($test) <> 0):
echo ("<center><h3>Student je již zadán</h3></center>");
```

If the condition is not met, it is good, because this a negative condition. Unmet condition means there is not any records for the special student-class combination. Unmet condition triggers execution of its else part.

```
Else:
If ($body>=95):
$b = "A";
$k = "$kredit";
endif;
If ($body<=94 and $body>=89):
$b = "B";
$k = "$kredit";
```

```
endif;
If ($body<=88 and $body>=83):
$b = "C";
$k = "$kredit";
endif;
If ($body<=82 and $body>=77):
$b = "D";
$k = "$kredit";
endif;
If ($body<=76 and $body>=70):
$b = "E";
$k = "$kredit";
endif;
If ($body<70):
$b = "F";
$k = "0";
endif;
If (empty($body)):
$b = " ";
$k = "0";
endif;
mysql_query("INSERT INTO `sp` ( `student_idstud` , `predmet_idpredmet` , `zapocet` ,
`datum_z` , `body` , `znamka` , `datum_zk` , `podpis` , `kredit`, `semestr`)
                    VALUES ('$jmeno', '$prt' , '$zapocet' , '$datum_z' , '$body' , '$b' ,
'$datum_zk', '$lg', '$k', '$smr');"
   ,$spojeni) or die("Nepodařilo se vložit");
Endif;    }
```

The condition is enriched by the conditions supposed to calculate the grades out of points from tests. I defined several ranges which assign different grades to various point values. It is obviously not only the matter of grades, but of credit values as well. If the student fails the class, the credits don't get added to hi or her total. If the student scores at least 70 points, the credit value is added to the total. The condition is also a part of UPDATE of the records, only with different variables input.

We continue with the SQL command, which will enter a new student into grades record keeping. It is abvious it enters the data into the table sp.

```
mysql_query("INSERT INTO `sp` ( `student_idstud` , `predmet_idpredmet` , `zapocet` ,
`datum_z` , `body` , `znamka` , `datum_zk` , `podpis` , `kredit`, `semestr`)
                    VALUES ('$jmeno', '$prt' , '$zapocet' , '$datum_z' , '$body' , '$b' ,
'$datum_zk', '$lg', '$k', '$smr');"
   ,$spojeni) or die("Nepodařilo se vložit");
    }
```

Then the command to update the records comes. Command update is used in the same way as in previous instances, thus I won't explain it anymore. Next up are the two forms, for creation of a new record and for editing of all the records selected by the criteria chosen in the script professor.php

I don't think I have to describe the form for creation, as it is exactly the same as for example in students' administration. It contains one listbox with SQL query and <input> tags, into which we enter values. The SQL command selects records from the table student with classroom being the criteria. Thus onl the current classroom gets displayed in the listbox instead of all the students.

The second form, which is supposed to update records already entered into sp table. The SQL query is quite complicated and is combining data gained from 4 different tables: student (first name, last name of the student), predmet, sp (data about credit tests and exams) and finally profesor, (each grade is flagged to enable the chance to track it back to the teacher, who actually entered it into the database). The table is then created by the cycle for, which I described earlier.

```
<?
 $vysledek=mysql_query("SELECT sp.idspojsp, student.jmeno, student.prijmeni,
predmet.nazev, sp.zapocet, sp.datum_z, sp.body, sp.znamka, sp.datum_zk, profesor.jmeno,
sp.kredit
  FROM student, predmet, sp, profesor
  WHERE sp.student_idstud = student.idstud AND sp.predmet_idpredmet =
predmet.idpredmet AND sp.podpis = profesor.login AND sp.predmet_idpredmet = $prt
AND sp.semestr = $smr
  ORDER BY student.prijmeni",$spojeni)
    or die("Nepodařilo se vykonat příkaz");

 for ($i=0;$i<=mysql_num_rows($vysledek)-1;$i++) {
    echo "<tr align=center valign=middle>
      <td>
      <input type=\"hidden\" name=\"id[]\"
         value=\"".mysql_result($vysledek,$i,"sp.idspojsp")."\" ></td>
      <td><label>".mysql_result($vysledek,$i,"student.jmeno")."</label></td>
      <td><label>".mysql_result($vysledek,$i,"student.prijmeni")."</label></td>
      <td>
      <input size=\"10\" type=\"text\" name=\"zapocet[]\"
         value=\"".mysql_result($vysledek,$i,"sp.zapocet")."\" ></td>
      <td><input size=\"8\" type=\"text\" name=\"datum_z[]\"
         value=\"".mysql_result($vysledek,$i,"sp.datum_z")."\" ></td>
      <td><input size=\"1\" type=\"text\" name=\"body[]\"
         value=\"".mysql_result($vysledek,$i,"sp.body")."\" ></td>
      <td><label>".mysql_result($vysledek,$i,"sp.znamka")."</label></td>
      <td><input size=\"8\" type=\"text\" name=\"datum_zk[]\"
      value=\"".mysql_result($vysledek,$i,"sp.datum_zk")."\" ></td>
      <td><label>".mysql_result($vysledek,$i,"profesor.jmeno")."</label></td>";

    echo "<td><a
href=\"zk_delete.php?id=".mysql_result($vysledek,$i,"sp.idspojsp")."&pret=$pret&trt=$trt
&smr=$smr\">x</a></td>";


 }
 ?>
```

There are two scripts left, z_delete and zk_delete. They contain the deletion SQL command to remove records from the tables sp_zs and sp_ls. The target script gets recognized – z_delete is for credit script (zapocet.php), zk_delete for exam script (zkouska.php)

That is it for the scripts from secure zone. Now I want to analyze scripts located in the unsecured folder report. These scripts are accessed via the link Přehled známek found in the index script.

3.1.3. Scripts overview of grades (reports)

Scripts that belong to the category overview of the grades (reports)

- report.php
- report_01.php
- report_02.php
- report_03.php
- report_04.php
- report_05.php

The area of grades' overview is unsecured one, and so anyone can access it without any login. Reports can be changed anytime on request thus if anyone asks for a new report, the scripts can be added and/or changed. I will mention here though interesting/important parts of the code from already existing scripts.

The first script is report.php. This script is basically just an option menu allowing the user to choose either in a simple way based on student or in a more complex way with class and classroom as the criteria.

Scripts report_01.php and report_02.php are designated for selection according to classroom and class. In the report_01.php script, there is a form containing a listbox with class selection, tag <input>, which results in entering the value classroom and another listbox to choose the semester at the end, after sending the data to script report_02.php, where the SQL command creates the table with data using cycle. Here I would only like to show the SQL command to select the data for the report "výběr podle třídy a předmětu" :

```
$dotaz = "SELECT sp.idspojsp, student.jmeno, student.prijmeni, predmet.nazev, sp.zapocet, sp.datum_z, sp.body, sp.znamka, sp.datum_zk, profesor.jmeno
FROM student, predmet, sp, profesor
WHERE sp.student_idstud = student.idstud AND sp.predmet_idpredmet = predmet.idpredmet AND sp.podpis = profesor.login AND idpredmet = $prt AND sp.semestr = $smr
ORDER BY student.prijmeni";
```

I will continue with the scripts report_03.php, report_04.php and report_05.php. These scripts mediate the selection of data of particular student. Thus the student comes to the computer, types in the address of this project and can view the complete set of grades for the semester.

Report_03.php selects student's classroom as the variable and sends it to script report_04.php, which immediately incorporates it into the SQL command intended to select the students from the desired classroom. Further we must input the semester we chose to display. The data gets sent to another script, namely report_05.php, the last script of this report. In this script, there is a SQL command to select already entered criteria from previous scripts. In this part of the code, we can see the SQL command of the script_05.php and creation of the table using the cycle for.

```
<?php
echo(„<center>“);
include „../connect.php“;
$MC = $spojeni;
$MS = MySQL_Select_DB;

 $dotaz = „SELECT sp.idspojsp, student.jmeno, student.prijmeni, predmet.nazev,
sp.zapocet, sp.datum_z, sp.body, sp.znamka, sp.datum_zk, sp.kredit, profesor.jmeno
 FROM student, predmet, sp, profesor
 WHERE sp.student_idstud = student.idstud AND sp.predmet_idpredmet =
predmet.idpredmet AND sp.podpis = profesor.login AND sp.student_idstud = $jmeno
AND sp.semestr = $smr
 GROUP BY sp.idspojsp“;

 $vysl = mysql_query($dotaz);
 $pocet_radku = mysql_num_rows($vysl);
 $pocet_sloupcu = mysql_num_fields($vysl);

 echo („<table border=2><caption>“.$_POST["databaze"].“</caption>“);
 echo („<tr bgcolor=#d3d3d3 align=center valign=middle >
<td>Jméno</td><td>Příjmení</td><td>Předmět</td><td>Zápočet</td><td>Datum
Z</td><td>Zkouška %</td><td>Zkouška známka</td><td>Datum
ZK</td><td>Kredit</td><td>Profesor</td></tr>“);
 $countFirstRow = 0;
 for($i=0; $i<$pocet_radku; $i++)
 {


  $pole = mysql_fetch_array($vysl);
  echo(„<tr align=center valign= middle>“);
  $countFirtsRow += $pole[9];

  for($j=1; $j<$pocet_sloupcu; $j++)
  {
   echo („<td>$pole[$j]</td>“);


    }
 }
 echo („</table>“);

echo(„</center>“);
?>
```

We can also notice the variable $countFirstRow, which is supposed to count the student's credit total. Thus the report represents the complex overview of student's exams, grades and credits. In order to get the listing of the credit values, the following code is used:

```
<? echo ("<center><h3>Součet kreditů: $countFirtsRow</h3></center>"); ?>
```

The chapter control (administration) via the web interface, I went through the important parts of the project's code. All the scripts that are in the application are mentioned here. From the directory structure all the way to detailed description of the scripts. Separated along the administrator, user and public grades overview.

The chapter is intended as a sort of manual for the future programmers, who will work with project to get a better grasp of the functions and tasks of various parts. Such manual will allow these programmers to work efficiently, because it will inform him or her how to handle the part of the code and what needs to be changed to tailor the project to particular need.

I certainly hope that this will be an useful manual and will serve the intended purpose well.

4. Project Record keeping of the grades at the EPI and its comparison with other such project at other institutions

The next part of my paper is the comparison between my project and other similar projects used at other schools. I obviously focused on the grading record keeping systems only, as some of the schools have a lot more complex software equipment with many other tasks.

Thus, I addressed a few schools with a request to send me some information about their grades keeping systems, or forward my information to person or persons, who are doing the administration for them.

I was interested at how many schools the old fashioned index entry was the only means to keep track of the results. Though I assumed it is impossible to do so without the use of modern equipment, I was surprisingly proved wrong!

Finally I chose the following projects for the comparison:

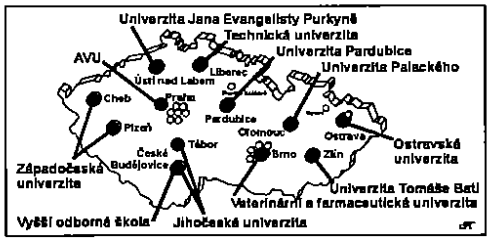
- IS/STAG – Západočeská univerzita v Plzni
- Projekt NETžákajda – this system is currently still in testing phase.

There is another system though I chose not to include in the comparison at the end. I mean the system "bakaláři". This system is used by the private high school of information technology. I don't include it in the comparison mainly because it is a commercial paid project and that is such a significant difference I think the comparison would not be fair. The project is run on Microsoft system – meaning MSSQL as the database system, IIS as the web server and asp.net is used for the web pages

4.1. IS/STAG

IS/STAG or Informační Systém STudijní AGendy (Information System of STudy Agenda) is a very sophisticated system developed by many people. The project has been created by the Center of information management and technology of the West Bohemian University in Plzeň. The origins of the system can be traced back to 1991, which is quite some time for the development. The time has paid back and the project is very successful even on international stage. Among others, I can mentioned the 3. place award from the Eunis Elite Award event in 2001, organized by the Association for information systems at the European Universities (EUNIS)

Picture 17: Coverage map of IS/STAG



Source: stag.zcu.cz

The project is of high quality that is confirmed by several schools outside of ZČU that are using it.

The System consists of these modules:

- Complex file keeping of the study (personal data as well as the process of the study)
- Entering and editing of syllabi for the courses;
- Entering and editing of study programs;
- Entering and editing of study plans;
- Entering and editing of the schedule;
- Pre-signup of the students;
- Grades input (done not by some central body, but by different departments throughout the university instead);
- Publishing and sign-up for the exam dates
- Preparation of the sign-up files
- A number of various print reports (ranging from particular prints concerning one student, course or schedule, up to starting sheet, matrix or medical insurance statements);
- Data exchange between STAG and dormitory accommodation system;
- Module to manage the process to accept new students;
- Module Alumni;
- module Evaluation (collecting and processing data from students' surveys)

The whole project is based on database system Oracle, contrary to my, SQL based project. This is the first cleave between the two – the communication is possible only remotely, via locally installed clients. The access is also provided by the web interface, though.

Comparison:

IS/STAG project has been created by many experts, while I worked on my project (with the exception of a few expertise consultations) alone. Just by saying so, the first argument why the projects are only hardly comparable has been brought up. But if wanted to keep the comparison going anyway, we woul hit another obstacle to do so – the database "heart" of the project is different. As stated above, my project is based on MySQL, which is a totally different system than Oracle.

The only comparison we can do is from the functional point of view. Even this comparison, has to be done between my project and the respective module of IS/STAG. But from this point of view, I think I can proudly say that both systems are doing their job very well.

4.2. NETžákajda

Project NETžákajda is not developed by a school, but by a certain programmer named Pavel Procházka, who aims his project on improving the communication between schools and families. This also determines the suitability of the project mainly to grade schools and high schools.

This, however, doesn't obstruct the comparison between my project and this one. The comparison should be between the systems as solutions, the actual implementation is irrelevant.

What doeas the system do?

- Improvement of the communication between the school and the family.
- Contains modules for messaging, input and control of the grades or electronic excuse module.

The system recognizes three different types of users:

1. pupil
2. teacher
3. administrator

The system also differentiates between regular and homeroom teachers allowing to be able to determine who is allowed to accept electronic excuse slips. Such acceptance is obviously allowed by the homeroom teacher only. Also it is only the homeroom teacher, who is allowed to do minor administrating tasks in his class – adding pupils, blocking existing ones or changing their passwords.

Administrator is allowed to add the teacher type users and has access to all the messages in the system.

All is saved in MySQL, indeed, the login is validated by the username and password from the database, checked by php script.

Comparison:

The project is built with the same logic as my application, with MySQL being the database system and PHP. The module for inputting the grades is virtually identical with mine, because the final input of the grade is based on class and course selection.

There is one objection I have to the project. The administrator has unlimited access to all the messages in the system. I don not think this is a good solution, as not even the admin shoul have unlimited power of the system, especially over messages he or she did not create.

I would suggest a solution to the author maybe adjusting the access rights so, that the administrator would not be allowed to read the messages, but he or she would be able to manage the messages based on some kind of abstract ID code of the message.

Apart from that, the project is very well done and the similarity with my project is significant.

5. Testing of the project, benefits for the submitter

5.1. Project testing

In order to finish the project, two tests had to be done.

The first one was done by me personally before I could submit the system for testing by the user. Thus this test focused on finding whether all the links, scripts tables and other parts work as they are supposed to. After I tested even the last SQL command, I could upload the system to the internet web server.

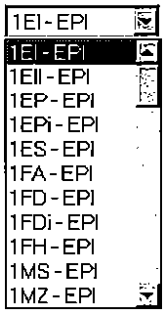
For the user testing I chose the internet server WEBZDARMA, which supports PHP as well as offering 5MB disk space for the SQL database. The project was ready for betatesting.

The betatest was carried out by Mr. ing. Hynek Dušek, who was not only a tester, but a sub,itter of the project as well. Thus, the testing was done by the client himself.The advantage of such an arrangement is in client's ability to express the needs and ask for adjustments when something is according to submitted plan, but would actually need a different solution. While testing, Mr. Dušek discovered several flaws which were solved on the spot.

Here I will describe just one issue we addressed during the betatest.

Originally, the selection of the classroom was done by a listbox. This also meant that to select a classroom, the user had to go through an extensive list of classes looking for the correct one. Mr. Dušek asked for solution of this issue using some kind of quick input field by the user.

Picture 18: Listbox for classroom selection



Author: Jakub Vilímek

Picture 19: Field for classroom quick entry

Author: Jakub Vilímek

Today, in the time of computers and digital information technology, it is almost impossible to unleash a new system upon the user without any kind of documentation (manual). I decided to solve this issue by using the possibilities of modern multimedia computers and prepared a complex tutorial video (included on the CD), which shows an unexperienced user how to navigate through the system.

5.2. Benefits for the submitter

Modernization of the current way of keeping the record about grades is very beneficial for the school. It is even so large for particular teachers I believe they will be glad to have gotten rid of the old way.

So, what are the main benefits, for the users?

Application is not local, it is uploaded on the server. The data can be accessed using regular web browser, allowing the access from the school as well as from home.

The information security is on higher level. I explained this statement in the introduction. The database is located on the server, which has a higher level of security, compared to the old record keeping system, much more vulnerable to the attack

Nicer and more transparent graphical presentation supports better orientation throughout the system than the screen full of cells and formulas. The graphic interface should make the system more user-friendly.

One of the main goals and benefits of the project was simplified processing of the grades. I think this requirement has been fulfilled, because the the time required to enter individual grade is much shorter than before.

These arguments support strongly the statement about significant benefiting of the school after transferring from the old system to the new one.

CONCLUSION

Currently, the society is going through turbulent times, when there are rapid changes going on in pretty much every area of life. The driving force of this development are computers. They influence everything mankind does, be it genetic engineering, space exploration programs, mobile technologies development or seemingly orthodox issues as schooling or culture.

It is no wonder, then, that the databases are no exception to this trend. The databases are not anymore just cardboard boxes filled with paper, but they are changing to be specialized and expert programs, which are and will be able to handle the requirements of common users, who will desire more safety, accessibility, transparency and effectiveness, when ordering the data processing applications.

There certainly isn't any need to talk about Verne's fantastic visions, when imagining interactive interconnected database systems that will inform us upon request verbally or graphically of any information we wish for.

Even more visible trend is the one of so called "active databases". These systems will not only collect and process information, but will actively predict the future needs of their users based on the data.

We can illustrate such idea by a future refrigerator, which not only monitors the stock of groceries, but will also actively order the supplies via internet-like network. Such a vision can obviously be used effectively not only concerning refrigerators, but number of other areas of human activity as well.

It is absolutely true that my EPI project is very far from such sophisticated and complex systems, but I am proud of the fact my work contributed also to the advancement of modern technology.

RESUME

The task of my bachelor's project was to replace and modernize the current grade files keeping system the basis of which lay in excel file. I decided to use the database system MySQL, script programming language PHP and webserver Apache to achieve this task (see chapter programming resources)

Though I encountered several major and minor obstacles (be it low practical experience with the used software, problems with structure designing, server configuration and even seemingly mundane problems with layout or grammar)

Even though it may sound like a cliché, I can frankly say that the more difficult to overcome the obstacles were, the better was and is my feeling from finally overcoming them. My pride for a well done work knowing the project is also helpful and useful and gave me many good experiences, is tremendous.

So, what are the benefits of this graduation project? Compared to the current system, it offers a totally different approach to data processing and allows the users to take advantage of the modern technology to collect, store, process and evaluate data. The advantages of my project can be briefly summarized in following points:

- better quality of access to the information via both internet and intranet
- higher level of security of the stored information
- more transparent graphical layout
- processing of the grades made simpler

In consideration to these statements, I think I can rate this project as successful

LIST OF USED ABBREVIATIONS:

MySQL	My Struktured Query Language
SQL	Struktured Query Language
PHP	Hypertext Preprocesor
HTML	Hypertext Markup Language
FTP	File Transfer Protocol
OOP	Object-oriented programming
EPI	European Polytechnical Institute
CSS	Cascading Style Sheets

LIST OF USED LITERATURE:

- WILLIAMS, Hugh, LANE, David. *PHP a MySQL : Vytváříme webové databázové aplikace*. David Krásenský. 2002. vyd. Hornocholupická 22, 143 00 Praha 4 : Computer press, 2002. 530 s. ISBN 80-7226-760-4.

- THOMSON, Laura, GILMORE, William J., WELLING, Luke. *PHP a MySQL : rozvoj webových aplikací*. 2002. vyd. Pod Křížem 15, 184 00 Praha 8 : SoftPress s.r.o., 2002. 692 s., 26. ISBN 80-86497-20-8.

- STANÍČEK, Petr. *CSS Kaskádové styly : Kompletní průvodce*. 2003. vyd. Nám. 28. dubna 48, 635 00 Brno : Computer Press, a.s., 2003. 178 s. ISBN 80-7226-872-4.

- *MySQL* [online]. MySQL. MySQL AB., 1995-2006 [cit. 2006-03-18]. Dostupný z WWW: <http://www.mysql.com/>.

- *PHP* [online]. The PHP group, 2001-2006 , last updated 20-3-2006 [cit. 2006-03-29]. Dostupný z WWW: <http://www.php.net/>.

- ROZSYPAL, Petr. *Php o php : PHP a první konktakt s databází* [online]. 2005 [cit. 2006-04-16]. Dostupný z WWW: <http://www.php-o-php.jicinet.cz/prvni_kontakt.php>.

APPENDIX:

Appendix 1: Classification table in Microsoft Exel

Klasifikační tabulka třídy :									3EPi
akademický rok 2005/2006									
Předmět : JAA									
Zkoušející : Ing. Jaroslav Kavka									
Semestr : zimní									
P. č.	Příjmení	Jméno	Zápočet			Zkouška			
			počet bodů	abecední stupeň	datum	počet bodů	abecední stupeň	datum	
1	Handl	Martin							
2	Janováč	Dušan							
3	Nunhardt	Milan							
4	Pelc	Michal							
5	Peller	Tomáš							
6	Sloboda	Peter							
7	Svída	Radim							
8	Unčík	Miroslav							
9	Vilímek	Jakub							
10	Zýbal	Karel							

Dne :

podpis