

СУ “Св. Климент Охридски”  
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

КУРСОВ ПРОЕКТ ПО  
ПРОЕКТИРАНЕ НА ФИЗИЧЕСКО НИВО И РЕАЛИЗАЦИЯ СЪС  
СУБД II

---

**База от данни за система за електронно  
обучение**

---

*Автори:*

Валентина Динкова,

ф.н.71112

Емил Станчев

ф.н.71100

*Ръководители:*

доц. В. Димитров

ас. Р. Горанова

20 май 2010 г.

## Съдържание

<b>1</b>	<b>Описание</b>	<b>2</b>
<b>2</b>	<b>Множества същности</b>	<b>2</b>
<b>3</b>	<b>ER Модел</b>	<b>2</b>
<b>4</b>	<b>Релационен модел</b>	<b>4</b>
<b>5</b>	<b>Ограничения</b>	<b>4</b>
<b>6</b>	<b>Функции</b>	<b>4</b>
<b>7</b>	<b>Тригери</b>	<b>4</b>
<b>8</b>	<b>Функции</b>	<b>6</b>
<b>9</b>	<b>Процедури</b>	<b>6</b>

## 1 Описание

Проектът представлява модел и примерна реализация на база от данни, предназначена за приложение за електронно обучение. В него учителите качват материали за различни курсове, студентите обсъждат във форум различни теми, свързани с курсовете, както и административни въпроси. Учителите могат да качват задания с определен краен срок, за които студентите получават оценка. Студентите могат да дават оценка на преподавателите в даден курс. Всички потребители се идентифицират с парола и email адрес.

## 2 Множества същности

Най-важните множества същности са:

**Course** Курс с име **name** за дадена година. Може да има курсове с еднакви имена в различни години, затова ключът се състои от името и годината на курса. Освен това има опционална парола **password** за записване на курса. Пази се и броят на записаните в курса студенти **numEnrolled**. Всеки курс има един титуляр **titular** и други учители **OtherTeachers**. Всеки курс има категория **Category**. Всеки студент **StudentProfile**, записан чрез **Enrolled** получава оценка за дадения курс **CourseGrade**, която има стойност **value**.

**User** Потребител, който има парола **password**, **email**, и имена **first name**, **last name**. Всеки потребител има поне едно от **StudentProfile** и **TeacherProfile**, които са слаби множества същности.

**Assignment** Задание, което има краен срок **deadline**, максимален брой точки, които дава заданието **max points**, заглавие **title**, описание **description**, уникален номер **number** и дата на създаване **created at**. Авторът на всеки **Assignment** е **TeacherProfile** на някой **User**. Всеки **Assignment** може да има чрез **Attached** прикачени файлове **File**, които имат име **name** и път **path**, който е ключ за файла. Всяко задание принадлежи на даден курс.

**Resource** Ресурс, който принадлежи на даден **Course**, напр. лекция, публикация и др. Ресурсът може да има прикачени файлове.

**ForumThread** Тема в дискусияния форум на даден курс, която има заглавие **title** и тяло **body**. Всяка тема може да има отговори **ForumReply**. Всички теми или отговори имат автори, които са **User**.

**News** Новина относно курс **Course**. Автори на новините са **TeacherProfile**.

**Notification** Известие за настъпило събитие, като например прибавяне или изтриване на **Assignment**, **ForumReply** и др.

## 3 ER Модел

Диаграмата на *Entity Relationship* модела е показана на Фигура 3.

The ER diagram illustrates the relationships between various entities in a course management system. The entities and their attributes are as follows:

- User**: email (PK), first\_name, last\_name, password, NumReplies.
- ForumReply**: created\_at, title, body, num\_likes.
- ForumThread**: created\_at, title, body.
- Assignment**: description, deadline, number (PK), title.
- TeacherProfile**: author (FK to User), title, created\_at.
- StudentProfile**: author (FK to User), value, rating.
- Course**: year (PK), name (PK), password, NumEnrolled, belongs\_to (FK to Course), course\_grade, value.
- File**: name, path, description.
- Resource**: id (PK), date, description.

The relationships and their cardinalities are:

- has\_points** (User to points): 1:M.
- author** (User to ForumReply): 1:M.
- author** (User to ForumThread): 1:M.
- author** (User to Assignment): 1:M.
- author** (User to TeacherProfile): 1:M.
- author** (User to StudentProfile): 1:M.
- has** (TeacherProfile to Assignment): 1:M.
- has** (StudentProfile to Assignment): 1:M.
- belongs\_to** (Course to Assignment): 1:M.
- belongs\_to** (Course to File): 1:M.
- belongs\_to** (Course to Resource): 1:M.
- attached** (Assignment to File): 1:M.
- attached** (File to Resource): 1:M.
- about** (Assignment to File): 1:M.
- about** (File to Resource): 1:M.
- about** (Assignment to Resource): 1:M.
- about** (File to Assignment): 1:M.
- about** (Resource to Assignment): 1:M.
- about** (Resource to File): 1:M.
- about** (Resource to Course): 1:M.
- about** (Course to Assignment): 1:M.
- about** (Course to File): 1:M.
- about** (Course to Resource): 1:M.
- about** (Course to Course): 1:M.

Specialization is used for TeacherProfile and StudentProfile, which are specialized views of the User entity.

## 4 Релационен модел

При преобразуването на *Entity Relationship* модела в релационен модел всички *много-към-едно* връзки са преобразувани в две релации вместо в три, като е използван *foreign key* в релацията за множеството, което стои от страната *много*. Множествата същности **File** и **Notification** са преобразувани в релации съответно *AssignmentFile*, *ResourceFile* и *CourseNotificaiton*, *AssignmentNotification* и *ForumReplyNotification*. Това е направено, за да може да се създаде *foreign key* от страна на **File** и **Notification**. Всички ключове са преобразувани до *минимални* такива.

Диаграмата на релационния модел е показана на Фигура 4.

## 5 Ограничения

Всички подходящи *foreign key* ограничения са наложени. Всички атрибути като `numEnrolled`, `num_likes` и т.н. са ограничени до неотрицателни стойности. Стойността на `year` атрибута на `Course` е ограничена до число между 1000 и 9999. За всеки `ForumReply` е наложено ограничението той да е създаден по-късно от родителя си. Всички пароли се изисква да са с дължина най-малко 5. За релацията `SpecialityLookup` се проверява дали `fn_from`  $\leq$  `fn_to`.

## 6 Функции

`get_speciality(fn INT) RETURNS VARCHAR(255)` Връща името на специалността на даден студент (по факултетен номер `fn`), използвайки релацията `SpecialityLookup`, в която са дефинирани интервали от факултетни номера и съответстващи имена на специалности.

`count_speciality_students(списък VARCHAR(255)) RETURNS INT` Връща броя на студентите от дадена специалност.

`teacher_mean_rating(teacher_email VARCHAR(255)) RETURNS DOUBLE` Връща средния рейтинг на даден учител - сумата от рейтингите от всички курсове, разделена на броя на всички рейтинги.

`all_course_teachers(n VARCHAR(255), y INT) RETURNS TABLE(User_email VARCHAR(255))` Връща релация, включваща всички учители за даден курс - титуляра и другите учители, ако има такива. `n` и `y` са името и годината на курса.

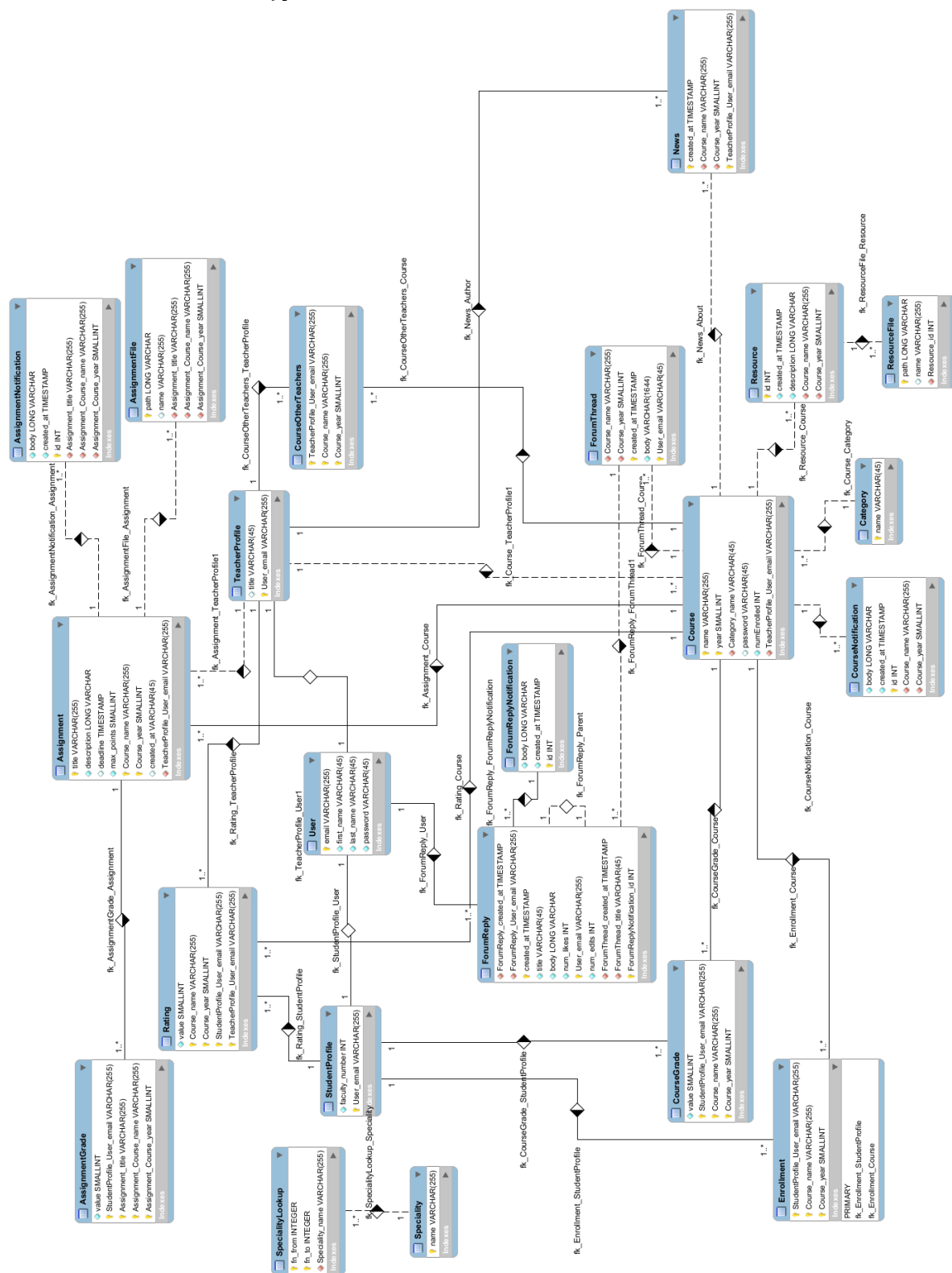
`too_old(t TIMESTAMP) RETURNS INT` Проверява дали времето `t` не определя асоцииран с него запис кат "прекалено стар", т.е. по-стар от 5 дни. Връща 0 или 1.

## 7 Тригери

*SQL* тригерите, дефинирани в проекта, са:

`tr_new_reply_notify AFTER INSERT ON ForumReply`

5



`tr_new_assignment_notify` AFTER INSERT ON Assignment

`tr_deleted_assignment_notify` AFTER DELETE ON ForumReply

Тези тригери създават известия за съответните събития.

`tr_enrollment_new_count` AFTER INSERT ON Enrollment

`tr_enrollment_delete_count` AFTER DELETE ON Enrollment

Тези тригери увеличават или намаляват `numEnrolled` на съответния курс, когато в него се запише или отпише студент.

## 8 Функции

*SQL* функциите, дефинирани в проекта, са:

`get_speciality(fn INT) RETURNS VARCHAR(255)` Връща името на специалността на даден студент (по факултетен номер `fn`), използвайки релацията `SpecialityLookup`, в която са дефинирани интервали от факултетни номера и съответстващи имена на специалности.

`count_speciality_students(spec VARCHAR(255)) RETURNS INT` Връща броя на студентите от дадена специалност.

`teacher_mean_rating(teacher_email VARCHAR(255)) RETURNS DOUBLE` Връща средния рейтинг на даден учител - сумата от рейтингите от всички курсове, разделена на броя на всички рейтинги.

`all_course_teachers(n VARCHAR(255), y INT) RETURNS TABLE(User_email VARCHAR(255))`  
Връща релация, включваща всички учители за даден курс - титуляра и другите учители, ако има такива. `n` и `y` са името и годината на курса.

`too_old(t TIMESTAMP) RETURNS INT` Проверява дали времето `t` не определя асоцииран с него запис кат "прекалено стар", т.е. по-стар от 5 дни. Връща 0 или 1.

## 9 Процедури

*SQL* процедурите, дефинирани в проекта, са:

`cleanup_old_notifications` Изтрива всички "твърде стари" известия според функцията `too_old`.

`urgent_assignment_notifications` Проверява за задания, за които остава по-малко от 1 ден до крайния срок и създава известия за тях.