Проект по Системи за управление на бази от данни

```
Тема: Електронни тестове
Изготвил: Иван Петров, ф.н. 3MI0700035
```

Съдържание

1. Обхват на модела. Дефиниране на задачата

Базата от данни за електронни тестове, ще съхранява информация за данните в университета нужни за тестовете. Разработената база от данни ще обслужва вътрешната система на университета, т.е. няма да се обслужва уеб приложение. Университета разполага с много факултети, но създаваните тестове не зависят от факултета. Всеки тест се определя еднозначно по идентификационен номер. Тестовете се състоят от множество въпроси, за които се пази информация. Точките, които допринасят, когато са отговорени вярно са различни за всеки въпрос. Всеки въпрос допринася различен брой точки в различните тестове, в които участва. Въпросите се определят еднозначно от идентификационен номер. Въпросите се разделят йерархично в ООП модел на множество типове въпроси, всеки с различна характеристика.

Типовете въпроси, добавени вече в системата, са: Есета, Затворени въпроси и Изчислителни въпроси. Всички те се определят еднозначно с идентификационния номер на въпросите, и имат условие в текстово поле. Есетата имат допълнителни характеристики за максимална и минимална дължина на есето.

В системата се пази информация за отговорите, които еднозначно се определят от идентификационен номер и номер на въпроса за когото отговарят. В отговорите се пази информация за това дали са верен отговор на въпроса, към който сочат, както и стойността на отговора в текстов формат.

В базата данни се съхранява информация за всички студенти от различните факултети на университета. За тях се пази тяхното име и фамилия, факултет и курс. Те се определят еднозначно от десетсимволния си факултетен номер.

За всеки студент се пази информация, на коя дата се явява на определен електронен тест, както и всеки тест, който е взет в съчетание с получената от него оценка.

2. Множества от същности и техните атрибути

- Изпити - номер тест, заглавие, нужни точки за минаване, научна сфера
- Въпроси - номер въпрос, точки (поне една, с точност до един знак след десетичната запетая, конкретния брой зависи от изпита)
    - Есе - условие, минимална дължина на отговор, максимална дължина на отговор
    - Затворен въпрос - условие
    - Изчислителен въпрос - условие
- Отговори - номер отговор, текст
- Студенти - факултетен номер, първо име, фамилия, курс, факултет

3. Домейн на атрибутите

- Изпити - номер тест : цяло положително число, заглавие: низ, минимални точки за минаване : положително цяло число, максимални точки на изпита, научна сфера: низ
- Въпроси - номер въпрос : цяло положително число, точки: цяло число (по-голямо или равно на 1)
    - Есе - условие : низ, минимална дължина на отговор: цяло положително число, максимална дължина на отговор: цяло положително число
    - Затворен въпрос - условие : низ,
    - Изчислителен въпрос - условие: низ
- Отговори - номер отговор: цяло положително число, текст: низ
- Студенти - факултетен номер : низ, първо име : низ, фамилия: низ, курс : цяло положително число, факултет : низ

4. Връзки

- В един Изпит има много Въпроси. Въпросите участват в много Изпити.
- Всеки Изпит има 0 или 1 нужен Изпит.
- Въпросите имат различни точки във всеки изпит
    - Есето е Въпрос
    - Затвореният въпрос е Въпрос
    - Изчислителният въпрос е Въпрос
- Един Въпрос има много Отговори. Един Отговор се асоциира само с един Въпрос.
- Студентите могат да се явят на много Изпити. На един Изпит се явяват много Студенти.
- Студентите могат да вземат много Изпити. Един Изпит може да бъде взет от много Студенти.

5. Ключове

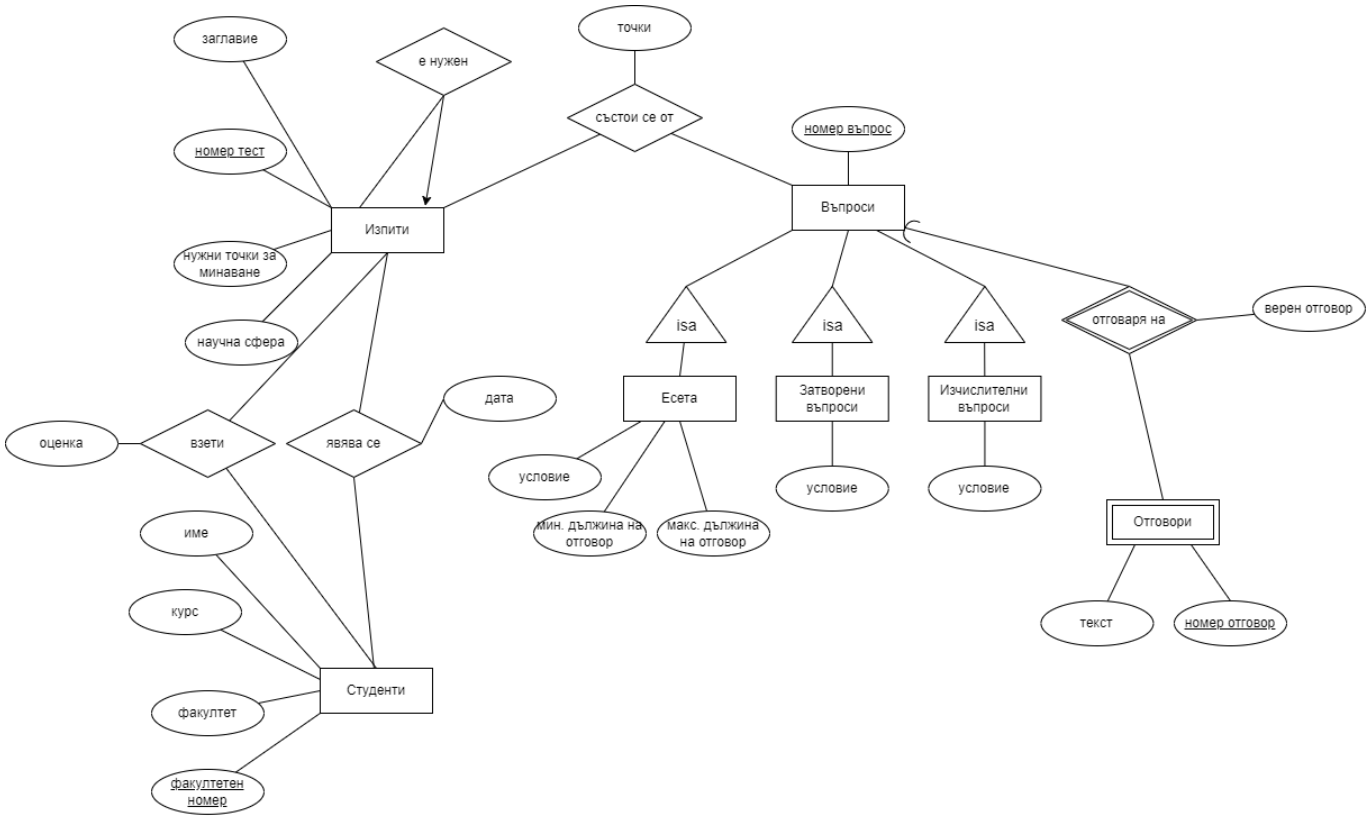
- Изпити - номер тест : еднозначно определя изпита
- Въпроси - номер въпрос : еднозначно определя въпроса
- Отговори - номер отговор : еднозначно определя въпроса

- Студенти - факултетен номер: еднозначно определя студента

6. Правила и проверки

- За изпитите - минимални и максимални точки: точките трябва да са неотрицателни и максималните точки трябва да са повече или равни на минималните
- За въпросите - точки: се прави проверка дали са неотрицателни
- За есетата - минималната дължина и максималната дължина: минималната дължина се прави проверка дали е по-малка или равна от максималната дължина
- За студентите - курс: се прави проверка да е повече от 0
- За взетите изпити - оценка: се прави проверка оценката да е по-голяма или равна на 2

7. E/R модел на данни



8. Релационен модел на данни

> Students(fn : char(10), fname : varchar(50), lname : varchar(50), faculty : string(100), course : smallint)

PK: (fn, unique, not null)

CK: (course > 0)

> Exams(nexam : int, title : varchar(255), minpoints : smallint, maxpoints : smallint, science_field : varchar(255), needed_nexam : int)

PK: (nexam, unique, not null)

FK: (needed_nexam)

CK: (minpoints > 0)

CK: (maxpoints>=minpoints)

> TakesExams(fn : char(10), nexam : int, date : date)

PK: (fn, nexam, unique, not null)

FK: (fn, nexam)

> TakenExams(fn : char(10), nexam : int, grade : decimal)

PK: (fn, nexam, unique, not null)

FK: (fn, nexam)

CK: (grade >= 2)

> ConsistsOf(nexam : int, nquestion : int, points : decimal)

PK: (nexam, nquestion , unique, not null)

FK: (nexam, nquestion)

CK: (points > 0)

> Answers(nanswer : int, text : varchar(100), nquestion : int, is_right_answer : char(1))

Отговорите са слаби множества

PK: (nanswer, text, nquestion, unique, not null)

FK: (nquestion)

CK: (is_right_answer - ‘0’, ‘1’)

Окончателен вариант

> Questions(nquestion : int)

PK:(nquestion, unique, not null)

> Essays(nquestion : int, condition : varchar(255), min_length : int, max_length : int)

PK:(nquestion, unique, not null)

FK:(nquestion)

CK: (min_length > 0)

CK: (max_length >= min_length)

> ClosedQuestions(nquestion : int, condition : varchar(255))

PK:(nquestion, unique, not null)

FK:(nquestion)

> CalculationQuestions(nquestion : int, condition : varchar(255))

PK:(nquestion, unique, not null)

FK:(nquestion)

NULL-подход

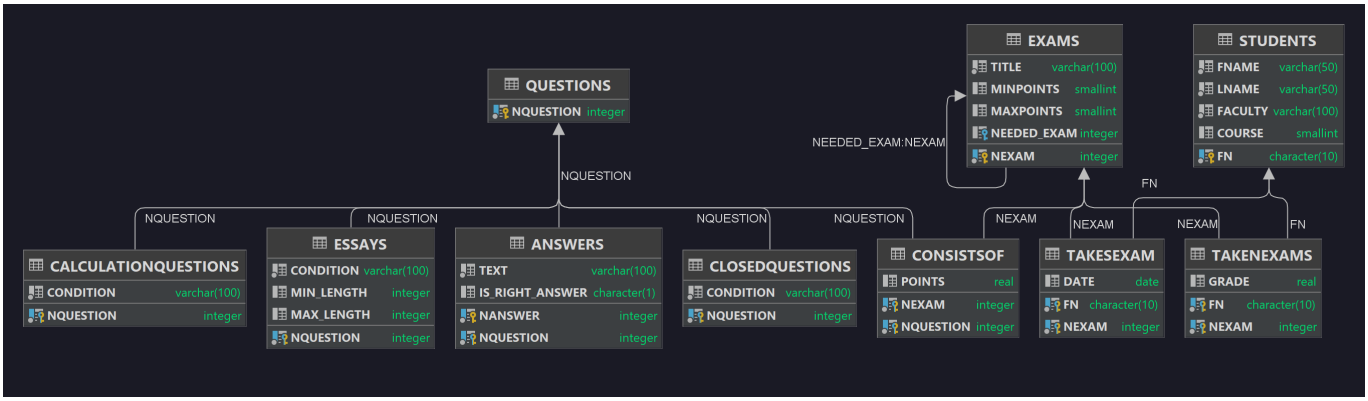
> Questions(nquestion : int, condition : varchar(255), min_length : int, max_length : int)

PK:(nquestion, unique, not null)

CK: (min_length > 0)

CK: (max_length >= min_length)

9. Схема на базата от данни



10. Функции

```
CREATE OR REPLACE FUNCTION FUNC_HAS_PASSED_EXAM(student_fn CHAR(10), exam_id INT)
    RETURNS SMALLINT
    LANGUAGE SQL
BEGIN
    DECLARE grade REAL;

    SELECT GRADE
    INTO grade
    FROM TAKENEXAMS
    WHERE FN = student_fn
      AND NEXAM = exam_id;

    RETURN CASE WHEN grade > 2 THEN 1 ELSE 0 end;
END;

VALUES (FN3MI0700035.FUNC_HAS_PASSED_EXAM( STUDENT_FN 'AB456CD789',  EXAM_ID 8)); -- 0
VALUES (FN3MI0700035.FUNC_HAS_PASSED_EXAM( STUDENT_FN 'AB456CD789',  EXAM_ID 11)); -- 1
```

Функция за проверка дали студент е минал даден изпит

```
CREATE OR REPLACE FUNCTION FUNC_AVG_GRADE_STUDENT(student_fn CHAR(10))
RETURNS REAL
BEGIN
    DECLARE avg_grade REAL;

    SELECT AVG(GRADE) INTO avg_grade
    FROM TAKENEXAMS
    WHERE FN = student_fn;

    RETURN avg_grade;
END;

VALUES(FN3MI0700035.FUNC_AVG_GRADE_STUDENT( STUDENT_FN 'AB456CD789')); -- 5
```

Функция за вземане средния успех на студент

11. Тригери

```
CREATE OR REPLACE TRIGGER TRIG_VALIDATE_RETAKE
    BEFORE INSERT
    ON TAKESEXAM
    REFERENCING NEW AS N
    FOR EACH ROW
    WHEN (EXISTS (SELECT *
                  FROM TAKENEXAMS
                  WHERE FN = N.FN
                    AND NEXAM = N.NEXAM))
BEGIN
    SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Student has already taken this exam';
end;
```

Тригер, който проверява дали студента си е взел изпита, преди вмъкване в таблицата за явяване на изпита

```
CREATE OR REPLACE TRIGGER TRIG_FIX_EXAM_POINTS
    BEFORE UPDATE
    ON EXAMS
    REFERENCING NEW AS N
    FOR EACH ROW
BEGIN
    DECLARE EXAM_POINTS REAL;
    CALL FN3MI0700035.PROC_CALC_EXAM_POINTS( EXAM_ID N.NEXAM,  EXAM_POINTS EXAM_POINTS);
    IF (EXAM_POINTS != N.MAXPOINTS)
    THEN
        SET N.MAXPOINTS = EXAM_POINTS;
    end if;
end;
```

Тригер преди обновяване на Изпит с правилния брой максимален брой точки

12. Изгледи

```sql
CREATE VIEW V_TAKENEXAM_STUD_INFO
AS
SELECT S.FNAME, S.LNAME, E.TITLE, E.SCIENCE_FIELD, TE.GRADE
FROM STUDENTS S,
     EXAMS E,
     TAKENEXAMS TE
WHERE TE.NEXAM = E.NEXAM
  AND TE.FN = S.FN;

SELECT *
FROM V_TAKENEXAM_STUD_INFO;
```

	FNAME	LNAME	TITLE	SCIENCE_FIELD	GRADE
1	Даниел	Маринов	Подобрения в ежедневието ни	Философия	4
2	Мария	Николова	Подобрения в ежедневието ни	Философия	3
3	Виктория	Симеонова	Математика за всеки	Математика	4
4	Виктория	Симеонова	Подобрения в ежедневието ни	Философия	6
5	Димитър	Димитров	Математика за всеки	Математика	3.5

```sql
DROP VIEW V_STUDENTS_ALL;
CREATE VIEW V_STUDENTS_ALL
AS
SELECT FN, FNAME, LNAME, FACULTY, COURSE
FROM STUDENTS
WHERE COURSE = '3';

SELECT *
FROM V_STUDENTS_ALL;

INSERT INTO V_STUDENTS_ALL(FN, FNAME, LNAME, FACULTY, COURSE)
VALUES ('FHG3TóKL9P', 'Гошо', 'Петров', 'Факултет по Глобално стопанство', 3);

SELECT *
FROM V_STUDENTS_ALL;
```

	FN	FNAME	LNAME	FACULTY	COURSE
1	AB123CD456	Александра	Петрова	Икономически университет	3
2	XY789Z0123	Георги	Милев	Факултет по право	3
3	EF012GH345	Димитър	Димитров	Факултет по информатика	3
4	FHG3TóKL9P	Гошо	Петров	Факултет по Глобално стопанство	3

```sql
DROP VIEW V_STUDENTS_ALL_WITH_CK;
CREATE VIEW V_STUDENTS_ALL_WITH_CK
AS
SELECT FN, FNAME, LNAME, FACULTY, COURSE
FROM STUDENTS
WHERE COURSE = '4'
WITH CHECK OPTION;

SELECT *
FROM V_STUDENTS_ALL_WITH_CK;

DELETE
FROM V_STUDENTS_ALL_WITH_CK
WHERE FN = 'M5N8D3K7T9';

INSERT INTO V_STUDENTS_ALL_WITH_CK(FN, FNAME, LNAME, FACULTY, COURSE)
VALUES ('M5N8D3K7T9', 'Калина', 'Марипосова', 'Факултет по кулинарни изкуства', 4); -- OK
```

	FN	FNAME	LNAME	FACULTY	COURSE
1	PQ567RS890	Даниел	Маринов	Изкуствен интелект и роботика	4
2	AB456CD789	Виктория	Симеонова	Факултет по икономика	4
3	M5N8D3K7T9	Калина	Марипосова	Факултет по кулинарни изкуства	4

```sql
INSERT INTO V_STUDENTS_ALL_WITH_CK(FN, FNAME, LNAME, FACULTY, COURSE)
VALUES ('H9F2J5L6P8', 'Йоан', 'Айрянов', 'Богословски факултет', 2); -- NOT OK
```

13. Процедури

```sql
CREATE OR REPLACE PROCEDURE PROC_CALC_EXAM_POINTS(IN EXAM_ID INT, OUT EXAM_POINTS REAL)
BEGIN
    SET EXAM_POINTS = (SELECT SUM(POINTS)
                       FROM CONSISTSOF
                       WHERE NEXAM = EXAM_ID);
end;

BEGIN
    DECLARE EXAM_POINTS REAL;
    CALL FN3MI0700035.PROC_CALC_EXAM_POINTS( EXAM_ID 8, EXAM_POINTS EXAM_POINTS);
    CALL DBMS_OUTPUT.PUT_LINE(DEC(EXAM_POINTS, 8, 2)); -- 212.00
end;
```

Процедура за връщане на общ брой точки на изпит

```
CREATE OR REPLACE PROCEDURE PROC_EXAM_AVG_GRADE(IN exam_id INTEGER, OUT avg_grade REAL)
BEGIN
    DECLARE DONE SMALLINT DEFAULT 0;
    DECLARE total_grade REAL DEFAULT 0;
    DECLARE student_count INT DEFAULT 0;
    DECLARE grade REAL;

    DECLARE C CURSOR FOR
        SELECT GRADE FROM TAKENEXAMS WHERE NEXAM = exam_id;

    DECLARE CONTINUE HANDLER FOR NOT FOUND
        BEGIN
            SET DONE = 1;
        end;
    OPEN C;

    READ_LOOP:
    LOOP
        FETCH C INTO grade;
        IF DONE = 1 THEN
            LEAVE READ_LOOP;
        END IF;

        SET total_grade = total_grade + grade;
        SET student_count = student_count + 1;
    END LOOP;

    CLOSE C;

    SET avg_grade = CASE WHEN student_count > 0 THEN total_grade / student_count ELSE -1 END;
END;
```

Процедура за изчисляване на средна оценка на изпит

```
CREATE OR REPLACE PROCEDURE PROC_UPDATE_EXAM_POINTS()
BEGIN
    DECLARE exam_id INT;
    DECLARE rows_updated INT;
    DECLARE SQLSTATE CHAR (5) DEFAULT '00000';

    DECLARE C CURSOR FOR
        SELECT NEXAM FROM EXAMS;

    OPEN C;

    WHILE SQLSTATE = '00000'
        DO
            UPDATE CONSISTSOF
            SET POINTS = POINTS + 10
            WHERE NEXAM = exam_id;

            SET rows_updated = (SELECT COUNT(*)
                                FROM CONSISTSOF
                                WHERE NEXAM = exam_id);

            UPDATE EXAMS
            SET MAXPOINTS = MAXPOINTS + (rows_updated * 10)
            WHERE NEXAM = exam_id;

            FETCH C INTO exam_id;
        end while;

    CLOSE C;
END;
```

Процедура за увеличаване точките на отговорите и съответно изпита

```
CREATE OR REPLACE PROCEDURE PROC_ADD_STUDENT(IN fac CHAR(10), IN fname VARCHAR(50), IN lname VARCHAR(50),
                                             IN faculty VARCHAR(100), IN course SMALLINT)
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
        BEGIN
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'Грешка при създаване на студент';
        END;

    INSERT INTO STUDENTS (FN, FNAME, LNAME, FACULTY, COURSE)
    VALUES ( FN fac, FNAME fname, LNAME lname, FACULTY faculty, COURSE course);
END;
```

Процедура за добавяне на студент

14. Приложение за достъп до базата

```
public static void main(String[] args) {
    DB2Manager db2 = new DB2Manager();

    String statement = " INSERT INTO FN3MI0700035.STUDENTS (FN, FNAME, LNAME, FACULTY, COURSE) " +
            "VALUES ('BGFD4324FW', 'Мартин', 'Самуилов', 'Богословски', 3)";
    db2.insert(statement);
    System.out.println();
    statement = "SELECT FNAME, LNAME FROM FN3MI0700035.STUDENTS";
    db2.select(statement);
    System.out.println();
    statement = """
            DELETE FROM FN3MI0700035.STUDENTS
            WHERE FN= 'BGFD4324FW'
            """;
    db2.delete(statement);
    statement = "SELECT * FROM FN3MI0700035.STUDENTS";
    db2.select(statement);
    System.out.println();
    db2.disconnect();
}
```

```
DB2 driver is loaded successfully
DB2 Database Connected
Rows changed: 1

FN          FNAME       LNAME       FACULTY
AB123CD456  Александра  Петрова     Икономически университет
DE789FG012  Иван        Костадинов  Технически университет
HI345JK678  Елена       Иванова     Медицински университет
LM901NO234  Петър       Георгиев    Инженерен университет
PQ567RS890  Даниел      Маринов     Изкуствен интелект и роботика
TU123VW456  Мария       Николова    Филологически факултет
XY789Z0123  Георги      Милев       Факултет по право
AB456CD789  Виктория    Симеонова   Факултет по икономика
EF012GH345  Димитър     Димитров    Факултет по информатика
FHG3T6KL9P  Гошо        Петров      Факултет по Глобално стопанство
M5N8D3K7T9  Калина      Марипосова  Факултет по кулинарни изкуства
BGFD4324FW  Мартин      Самуилов    Богословски

Rows changed: 1
FN          FNAME       LNAME       FACULTY
AB123CD456  Александра  Петрова     Икономически университет
DE789FG012  Иван        Костадинов  Технически университет
HI345JK678  Елена       Иванова     Медицински университет
LM901NO234  Петър       Георгиев    Инженерен университет
PQ567RS890  Даниел      Маринов     Изкуствен интелект и роботика
TU123VW456  Мария       Николова    Филологически факултет
XY789Z0123  Георги      Милев       Факултет по право
AB456CD789  Виктория    Симеонова   Факултет по икономика
EF012GH345  Димитър     Димитров    Факултет по информатика
FHG3T6KL9P  Гошо        Петров      Факултет по Глобално стопанство
M5N8D3K7T9  Калина      Марипосова  Факултет по кулинарни изкуства
```