

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
на тему

«Веб-приложение для взаимодействия учителей и родителей в школе»

Студент

Ермолович Дмитрий Сергеевич
Гр. 053504

Руководитель

Ассистент кафедры информатики
Давыдчик А.В.

Минск 2023

ВВЕДЕНИЕ

В современном образовательном пространстве сотрудничество между учителями и родителями играет ключевую роль в обеспечении успешности учебного процесса и развитии учеников. Однако, несмотря на значимость этой взаимодействия, оно часто сталкивается с различными вызовами, связанными с ограниченной доступностью времени, общением и координацией. В этом контексте, разработка и внедрение приложения для взаимодействия учителей и родителей в школе представляет собой актуальную и значимую задачу, которая может улучшить образовательный процесс и усилить вовлеченность всех участников.

Данное приложение может стать эффективным инструментом, способствующим облегчению коммуникации и сотрудничества между учителями и родителями. Оно предоставит возможность учителям оперативно и информативно обмениваться информацией о прогрессе и достижениях учеников, а также давать обратную связь по учебным вопросам. Родителям, в свою очередь, будет предоставлена уникальная возможность более активно включаться в учебный процесс своих детей, следить за их успехами, а также общаться с учителями и администрацией школы.

В данной курсовой работе будут рассмотрены основные аспекты проектирования и разработки такого приложения, а также его потенциальные преимущества и вызовы. Мы также рассмотрим современные технологические решения, которые могут быть использованы для создания данного приложения, и практические шаги, необходимые для его успешной реализации.

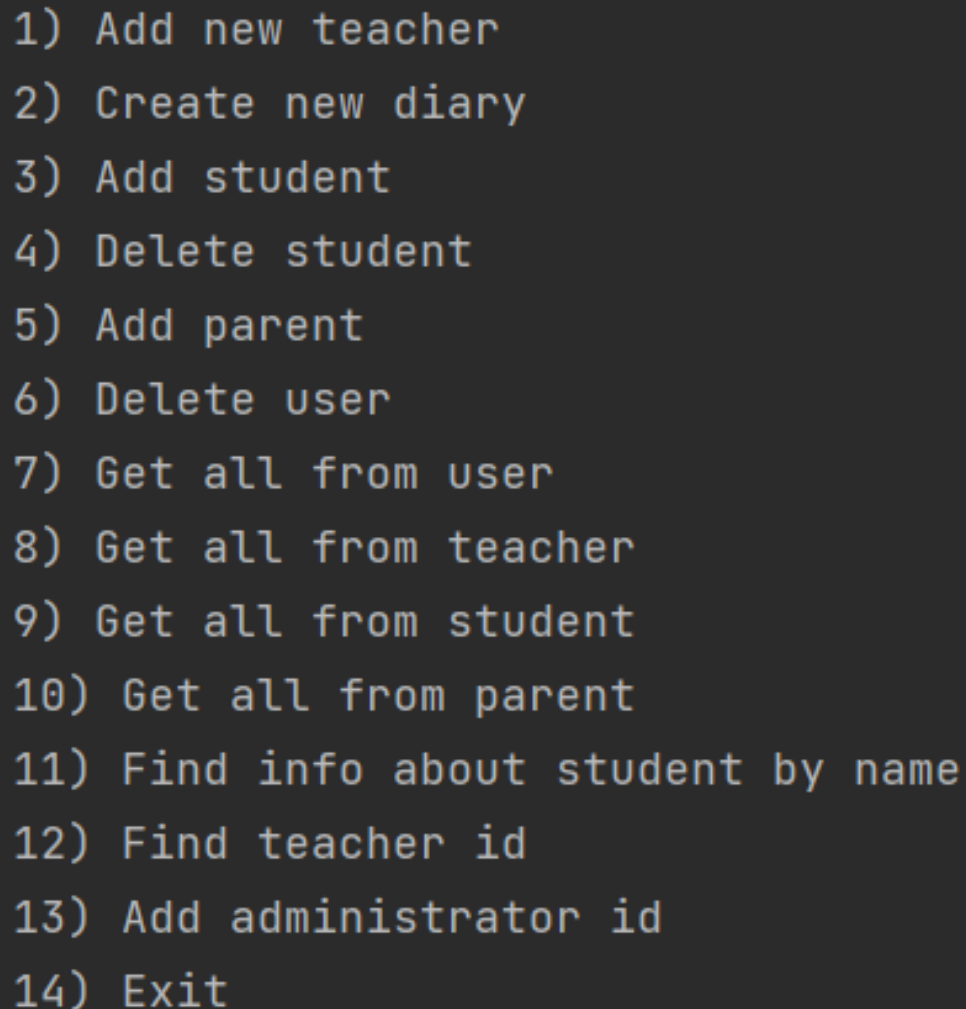
Цель данного исследования - дать вклад в улучшение коммуникации и сотрудничества между учителями и родителями в образовательных учреждениях, что, в свою очередь, содействует повышению качества образования и успешному развитию учеников.

1 РОЛИ И ИХ ФУНКЦИИ

Администратор.

Администратор имеет следующие функции. Добавить нового учителя, добавить дневник, добавить студента, удалить студента, добавить родителя, добавить юзера, удалить юзера, выбрать всех юзеров, учителей, студентов, родителей, найти ids по именам, добавить администратора.

На рисунке 1 изображены функции администратора.



```
1) Add new teacher
2) Create new diary
3) Add student
4) Delete student
5) Add parent
6) Delete user
7) Get all from user
8) Get all from teacher
9) Get all from student
10) Get all from parent
11) Find info about student by name
12) Find teacher id
13) Add administrator id
14) Exit
```

Рисунок 1 - Функции администратора

Студент.

Студент имеет следующие функции. Добавить жалобу, получить оценки свои и при желании любого человека, найти ids любого человека

На рисунке 2 изображены функции студента.

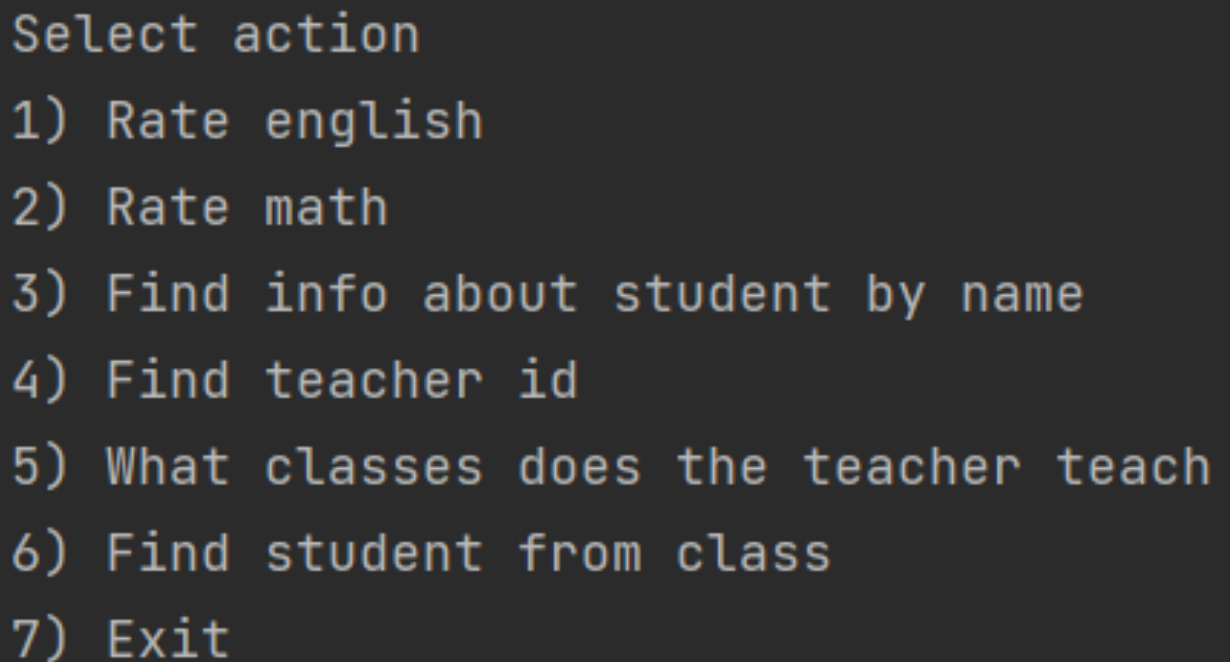
```
Select action
1) Add complain
2) Get english marks
3) Get math marks
4) Find info about student by name
5) Find teacher id
6) Get your english marks
7) Get your math marks
8) Exit
```

Рисунок 2 - Функции студента

Учитель.

Поставить оценку по английскому и математике только студентам у которого он преподаёт, найти студента и учителя, у каких учителей преподаёт, вывести всех студентов класса.

На рисунке 3 изображены функции учителя.



```
Select action
1) Rate english
2) Rate math
3) Find info about student by name
4) Find teacher id
5) What classes does the teacher teach
6) Find student from class
7) Exit
```

Рисунок 3 - Функции учителя

Родители.

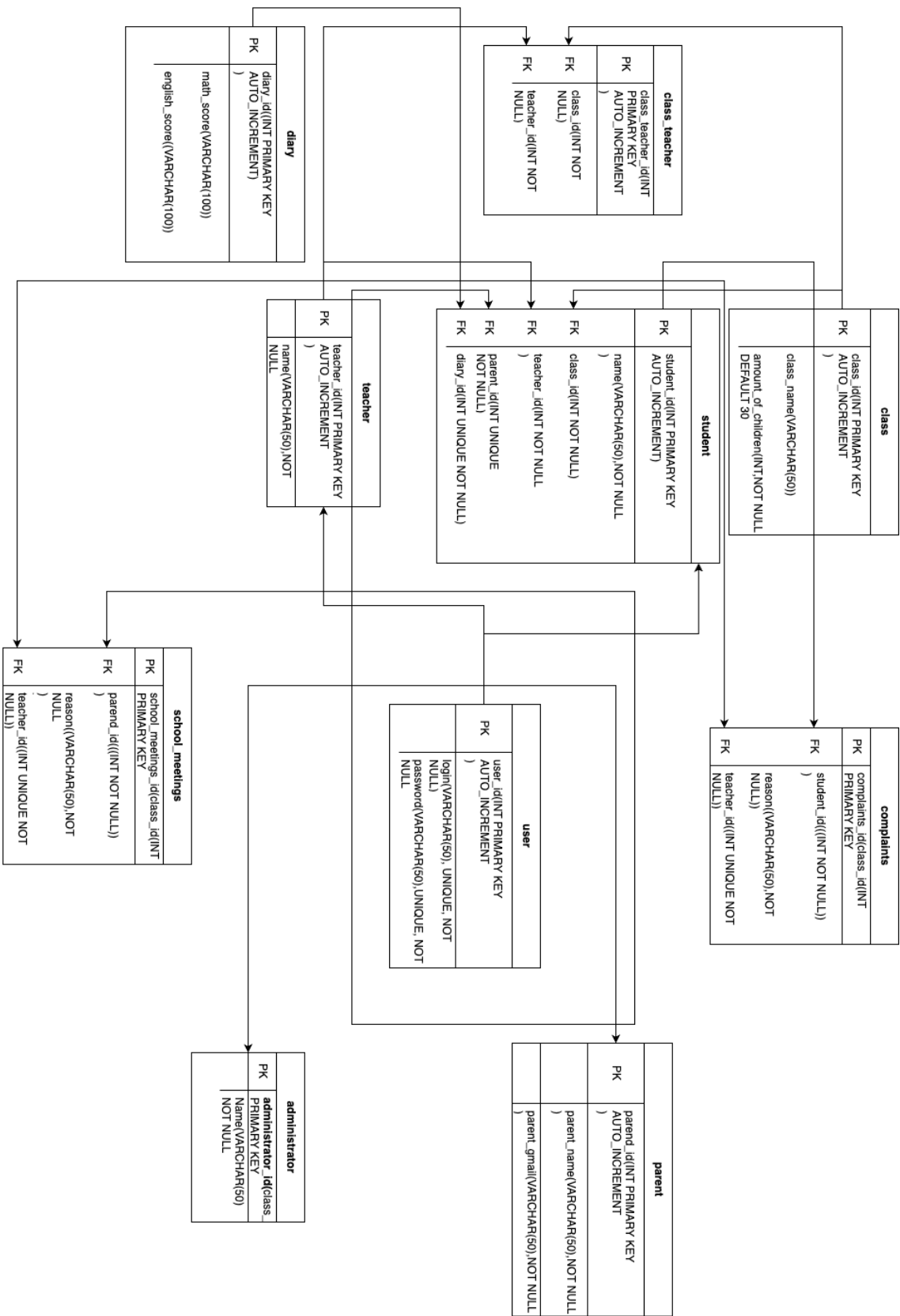
Добавить встречу, получить оценки твоего ребенка или любого, найти учителей и студентов

На рисунке 4 изображены функции родителя.

```
Hello parent
Select action
1) Add meeting
2) Get english marks
3) Get math marks
4) Find info about student by name
5) Find teacher id
6) Get your child english marks
7) Get your child math marks
8) Exit
```

Рисунок 4 - Функции родителя

Физическая диаграмма (будет дополнена)



Запросы

1 У кого ведет Svetlana Alexandrovna + Запросы на выборку из нескольких таблиц

```
SELECT amount_of_children,class_name,name
FROM class_teacher
    INNER JOIN class
        ON class.class_id = class_teacher.class_id
    INNER JOIN teacher
        ON teacher.teacher_id = class_teacher.teacher_id
WHERE
    name = "Svetlana Alexandrovna"
```

2 Какие преподаватели ведут у 10А класса

```
SELECT amount_of_children,class_name,name
FROM class_teacher
    INNER JOIN class
        ON class.class_id = class_teacher.class_id
    INNER JOIN teacher
        ON teacher.teacher_id = class_teacher.teacher_id
WHERE
    class_name = "10A"
```

3 Люди которые учатся в одном классе

```
SELECT name, class_name, amount_of_children,behavior
FROM student
    INNER JOIN class
        ON class.class_id = student.class_id
WHERE student.class_id = 1
```

4 Люди которые учатся в классе 1 и поведение больше чем два

```
SELECT name, class_name, amount_of_children,behavior
FROM student
    INNER JOIN class
        ON class.class_id = student.class_id
WHERE student.class_id = 1 AND behavior>2
```


5 Найдём среднее, максимальное и минимальное поведение по классам

```
SELECT class_name, MAX(behavior) AS max_behaviour, MIN(behavior) AS  
min_behaviour, AVG(behavior) AS avg_behaviour  
FROM student  
INNER JOIN class  
ON class.class_id = student.class_id  
GROUP BY class_name
```

6 Найти максимальное поведение по классам

```
SELECT class_name, MAX(behavior) AS Максимальное_поведение FROM  
student  
INNER JOIN class ON student.class_id = class.class_id  
GROUP BY class_name  
HAVING MAX(behavior) < 5  
ORDER BY MAX(behavior) DESC
```

7 Вывести все классы у которых ведут преподады

```
SELECT teacher.name, class_id  
FROM  
teacher LEFT JOIN class_teacher  
ON teacher.teacher_id = class_teacher.teacher_id
```

8 Каждому преподау поставить все возможные классы

```
SELECT teacher.name, class.class_name  
FROM  
teacher CROSS JOIN class
```

9 Вывести минимальными с минимальными оценками поведения

```
SELECT name, behavior  
FROM student  
WHERE behavior = (SELECT MIN(behavior) FROM student)
```

10 Нормировать поведение

```
SELECT name, behavior, (behavior - (SELECT AVG(behavior) FROM  
student))/(SELECT STD(behavior) FROM student) AS norma  
FROM student
```

11 Вывести все классы, среднее поведение которых больше 3

```
SELECT class.class_name
FROM student
INNER JOIN class ON student.class_id = class.class_id
WHERE class.class_id IN (
  SELECT class_id
  FROM student
    GROUP BY class_id
    HAVING AVG(behavior)>3
)
LIMIT 1
```

12 Оценка поведения студентов

```
SELECT name, behavior,
CASE
  WHEN behavior>3 THEN 'good'
  WHEN behavior<2 THEN 'bad'
  ELSE 'medium'
END AS Beh
FROM student
```

Триггеры и процедуры

1

```
DELIMITER //
CREATE PROCEDURE add_user(IN login_1 VARCHAR(50))
BEGIN
  DECLARE password_1 VARCHAR(50);
  SET password_1 = "123456";
  INSERT INTO users(login,password) VALUES (login_1,password_1);
END //
```

2

```
DELIMITER //
CREATE TRIGGER app_trig
BEFORE INSERT ON administrator
FOR EACH ROW
```

```
BEGIN
    SET NEW.name = "DZIMAZ";
END //
```

```
3 ADMIN
DELIMITER //
CREATE PROCEDURE add_parent(IN gmail_ VARCHAR(50),IN name_
VARCHAR(50))
BEGIN
    INSERT INTO parent(parent_gmail,parent_name)
    VALUES (gmail_,name_);
END //
```

```
CALL add_parent("Hello@gmail.com", "Dver")
```

```
5
DELIMITER //
CREATE PROCEDURE add_teacher(IN name_ VARCHAR(50))
BEGIN
    INSERT INTO teacher(name)
    VALUES (name_);
END //
CALL add_teacher("Anisimov")
```

```
6
DELIMITER //
CREATE PROCEDURE create_diary()
BEGIN
    INSERT INTO diary(english_score,math_score)
    VALUES ("","");
END //
CALL create_diary()
```

```
7
DELIMITER //
CREATE PROCEDURE add_math_score(IN id_ INT,IgetN math_score_
VARCHAR(50))
```

```
BEGIN
UPDATE diary
SET math_score = CONCAT(math_score," ",math_score_)
WHERE diary.diary_id = id_;
END //
```

```
CALL add_math_score(11,"10")
CALL add_math_score(11,"9");
```

```
8
DELIMITER //
CREATE PROCEDURE add_user(IN login_ VARCHAR(50),IN password_
VARCHAR(50),IN role_id_ INT,IN your_personal_data_id_ INT)
BEGIN
INSERT INTO users(login,password,role_id,your_personal_data_id)
VALUES
(login_,password_,role_id_,your_personal_data_id_);
END //
CALL add_user("Mikola@gmail.com","1234",1,0)
```

```
9
DELIMITER //
CREATE PROCEDURE add_english_score(IN id_ INT,IN english_score_
VARCHAR(50))
BEGIN
UPDATE diary
SET english_score = CONCAT(english_score," ",english_score_)
WHERE diary.diary_id = id_;
END //
CALL add_english_score(11,"9")
```

```
10
DELIMITER //
CREATE PROCEDURE find_student_id(IN name_ VARCHAR(50))
BEGIN
    SELECT student.student_id,student.name FROM student
    WHERE name = name_;
```

END//

11

DELIMITER //

```
CREATE PROCEDURE add_student(IN name_ VARCHAR(50),IN class_id_ INT,
teacher_id_ INT, IN parent_id_ INT,IN diary_id_ INT, IN behavior_ INT)
BEGIN
INSERT INTO student(name, class_id, teacher_id, parent_id, diary_id,behavior)
VALUES(name_, class_id_, teacher_id_, parent_id_, diary_id_,behavior_);
END //
CALL add_student("Xor",2,1,3,11,5)
```

12

```
INSERT INTO school_meetings(parent_id,reason,teacher_id)
VALUES(1,"Bad marks",1)
```

DELIMITER //

```
CREATE PROCEDURE add_meeting(IN parent_id_ INT,IN reason_
VARCHAR(50), IN teacher_id_ INT)
BEGIN
INSERT INTO school_meetings(parent_id,reason,teacher_id)
VALUES(parent_id_,reason_,teacher_id_);
END //
CALL add_meeting(2,"Simple",2)
```

13

DELIMITER //

```
CREATE PROCEDURE get_user_data(IN login_ VARCHAR(50),OUT role_id_
INT, OUT password_ VARCHAR(50), OUT your_personal_data_id_
VARCHAR(50))
BEGIN
SELECT role_id, password,your_personal_data_id
INTO role_id_,password_,your_personal_data_id_
FROM users
WHERE users.login = login_;
END //
```

14

```

DELIMITER //
CREATE PROCEDURE add_complain(IN student_id_ INT,IN reason_
VARCHAR(50), IN teacher_id_ INT)
BEGIN
    INSERT INTO complaints(reason,student_id,teacher_id)
    VALUES(reason_,student_id_,teacher_id_);
END //
CALL add_complain(3,"Simple_2",3)

```

```

15
DELIMITER //
CREATE PROCEDURE find_teacher_id(IN name_ VARCHAR(50))
BEGIN
    SELECT teacher.teacher_id,teacher.name FROM teacher
    WHERE teacher.name = name_;
END

```

```

16
DELIMITER //
CREATE PROCEDURE delete_student(IN student_id_ INT)
BEGIN
    DELETE FROM student
    WHERE student_id = student_id_;
END

```

```

17
DELIMITER //
CREATE TRIGGER insert_into_users_parent
AFTER INSERT ON parent
FOR EACH ROW
BEGIN
    INSERT INTO users(login,password,role_id,your_personal_data_id)
    VALUES
    (CONCAT(NEW.parent_id,"_", "4",NEW.parent_gmail),CONCAT(NEW.parent_n
ame,"_4_",NEW.parent_id),4,NEW.parent_id);
END

```

18

```
DELIMITER //
CREATE TRIGGER bahavior_check
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    IF NEW.behavior>5 THEN SET NEW.behavior = 5; END IF;
    IF NEW.behavior<0 THEN SET NEW.behavior = 0; END IF;
END
CALL add_student("Polyna",2,1,3,12,12)
CALL add_student("Victoria",2,1,3,13,-2)
```

19

```
DELIMITER //
CREATE TRIGGER check_first_letter
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    SET NEW.name =
CONCAT(UPPER(SUBSTRING(NEW.name,1,1)),SUBSTRING(NEW.name,2));
END
CALL add_student("angela",2,1,3,14,-2)
```

20

```
DELIMITER //
CREATE TRIGGER update_acad
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
    UPDATE academic_performance
    SET score = score+NEW.behavior
    WHERE academic_performance.academic_performance_id = 1;
END
CALL add_student("veniamin",2,1,3,15,2)
```

21

```
DELIMITER //
```

```
CREATE TRIGGER insert_into_users_teacher
AFTER INSERT ON teacher
FOR EACH ROW
BEGIN
INSERT INTO users(login,password,role_id,your_personal_data_id)
VALUES
(CONCAT(NEW.name,"_",NEW.teacher_id,"_",3,"@gmail.com"),CONCAT(NE
W.name,"_",NEW.teacher_id),3,NEW.teacher_id);
END
```

```
CALL add_teacher("Chekanova")
```

22

```
DELIMITER //
CREATE TRIGGER insert_into_users_student
AFTER INSERT ON student
FOR EACH ROW
BEGIN
INSERT INTO users(login,password,role_id,your_personal_data_id)
VALUES
(CONCAT(NEW.name,"_",NEW.student_id,"_",1,"@gmail.com"),CONCAT(NE
W.name,"_",NEW.student_id),1,NEW.student_id);
END
```


DDL скрипты

```
CREATE DATABASE new_data_basa_1
```

```
CREATE TABLE teacher(  
    teacher_id INT PRIMARY KEY AUTO_INCREMENT,  
    name VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE class(  
    class_id INT PRIMARY KEY AUTO_INCREMENT,  
    class_name VARCHAR(50),  
    amount_of_children INT NOT NULL DEFAULT 30  
);
```

```
CREATE TABLE class_teacher(  
    class_teacher_id INT PRIMARY KEY AUTO_INCREMENT,  
    class_id INT NOT NULL,  
    teacher_id INT NOT NULL,  
    FOREIGN KEY (class_id) REFERENCES class (class_id),  
    FOREIGN KEY (teacher_id) REFERENCES teacher (teacher_id) );
```

```
INSERT INTO teacher(name)  
VALUES  
    ("Irina Alexandrovna"),  
    ("Svetlana Alexandrovna"),  
    ("Georgi Semenovich"),  
    ("Elena Alexandrovna");
```

```
INSERT INTO class(amount_of_children,class_name)  
VALUES  
    (28,"10A"),  
    (27,"10B"),  
    (21,"5V"),  
    (19,"8A");
```

```
INSERT INTO class_teacher(class_id,teacher_id)  
VALUES
```

(1,1),
(1,2),
(1,3),
(2,2),
(3,1),
(4,1);

```
CREATE TABLE parent (  
    parent_id INT PRIMARY KEY AUTO_INCREMENT,  
    parent_name VARCHAR(50) NOT NULL,  
    parent_gmail VARCHAR(50) NOT NULL  
);
```

```
INSERT INTO parent (parent_name,parent_gmail)  
VALUES  
    ("Dima Ermolovich","dima123@gmail.com"),  
    ("Alex Sneshko","alex007@gmail.com"),  
    ("Tima Dydich","timoxa432@gmail.com"),  
    ("Stes","big_stas22@gmail.com");
```

```
CREATE TABLE diary(  
    diary_id INT PRIMARY KEY AUTO_INCREMENT,  
    math_score VARCHAR(100),  
    english_score VARCHAR(100)  
);
```

```
INSERT INTO diary(math_score,english_score)  
VALUES  
    ("10 9 10","7 7 7"),  
    ("6 2 8","9 7 7 8"),  
    ("5 3 2 1","4 3 2"),  
    ("9 6 8 7 8","9 9"),  
    ("1 6 8","6 3 8");
```

```
CREATE TABLE student(  
    student_id INT PRIMARY KEY AUTO_INCREMENT,  
    name VARCHAR(50) NOT NULL,  
    class_id INT,  
    teacher_id INT,
```

```
parent_id INT,  
diary_id INT UNIQUE,  
FOREIGN KEY (class_id) REFERENCES class (class_id),  
FOREIGN KEY (teacher_id) REFERENCES teacher (teacher_id),  
FOREIGN KEY (parent_id) REFERENCES parent (parent_id),  
FOREIGN KEY (diary_id) REFERENCES diary (diary_id)  
);
```

```
INSERT INTO student(name,class_id,teacher_id,diary_id,parent_id)  
VALUES  
("a",1,1,1,1),  
("b",1,1,2,1),  
("c",1,1,3,1);
```

```
CREATE TABLE complaints(  
complaints_id INT PRIMARY KEY AUTO_INCREMENT,  
reason VARCHAR(50) NOT NULL,  
student_id INT,  
teacher_id INT,  
FOREIGN KEY (teacher_id) REFERENCES teacher (teacher_id),  
FOREIGN KEY (student_id) REFERENCES student (student_id)  
);
```

```
CREATE TABLE school_meetings(  
school_meetings_id INT PRIMARY KEY AUTO_INCREMENT,  
reason VARCHAR(50) NOT NULL,  
parent_id INT,  
teacher_id INT,  
FOREIGN KEY (teacher_id) REFERENCES teacher (teacher_id),  
FOREIGN KEY (parent_id) REFERENCES parent (parent_id)  
);
```

```
INSERT INTO complaints(reason,student_id,teacher_id)  
VALUES  
("she yelled at me",1,1),  
("she is bad",2,2);
```

```
CREATE TABLE role
(
role_id INT PRIMARY KEY AUTO_INCREMENT,
role_name VARCHAR(50) NOT NULL UNIQUE
);
```

```
CREATE TABLE users
(
users_id INT PRIMARY KEY AUTO_INCREMENT,
login VARCHAR(50) NOT NULL UNIQUE,
password VARCHAR(50) NOT NULL,
your_personal_data_id INT NOT NULL,
role_id INT,
FOREIGN KEY(role_id) REFERENCES role (role_id) ON DELETE RESTRICT
);
```

```
INSERT INTO role(role_name)
VALUES
("admin"),
("student"),
("teacher"),
("parent");
```

```
INSERT INTO users(login,password,role_id,your_personal_data_id)
VALUES
("dima@gmail.com","123456",1,0);
```

```
INSERT INTO parent(parent_name,parent_gmail)
VALUES ("Dima","qadsa@gmail");
```

```
INSERT INTO school_meetings(parent_id,reason,teacher_id)
VALUES
(1,"Marks",1);
```

```
INSERT INTO diary(math_score,english_score)
VALUES
```

```
("1 3 5","7 7 7"),  
("7 4 1","5 7 5 8"),  
("3 3 6 2","4 5 2"),  
("1 6 7","8 8"),  
("4 2 1","3 2 8");
```

```
INSERT INTO student(name,class_id,teacher_id,diary_id,parent_id)  
VALUES  
("Egor",1,2,5,1),  
("Anton",3,3,6,2),  
("Nikolay",3,3,7,3),  
("Misha",3,3,8,1),  
("Nikita",2,1,9,2),  
("Ilya",2,1,10,3);
```

```
ALTER TABLE student  
ADD behavior INT;
```

```
ALTER TABLE student  
ADD CONSTRAINT diary_id FOREIGN KEY (diary_id)  
REFERENCES diary (diary_id)  
ON DELETE RESTRICT;
```

```
ALTER TABLE student  
ADD CONSTRAINT class_id FOREIGN KEY (class_id)  
REFERENCES class (class_id)  
ON DELETE SET NULL;
```

```
ALTER TABLE student  
ADD CONSTRAINT parent_id FOREIGN KEY (parent_id)  
REFERENCES parent (parent_id)  
ON DELETE SET NULL;
```

```
ALTER TABLE parent CHANGE parent_name parent_name VARCHAR(30) NOT  
NULL;
```

```
ALTER TABLE complaints  
ADD CONSTRAINT student_id FOREIGN KEY (student_id)  
REFERENCES student (student_id)  
ON DELETE SET NULL;
```