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

 Ассистент кафедры информатики

Руководитель 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, добавит родитедя, добавить юзера, удалить юзера, выбрать всех юзеров, учителей, студентов, родителей, найти ids по именам, добавить администратора.

На рисунке 1 изображены функции администратора.

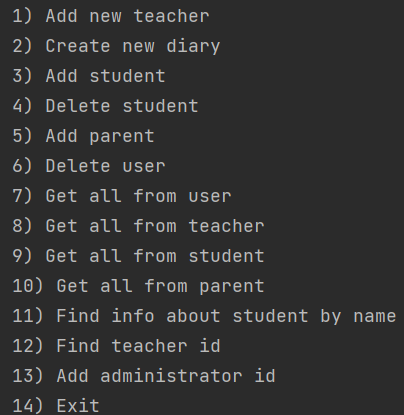


Рисунок 1 - Функции администратора

Студент.

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

На рисунке 2 изображены функции студентa.

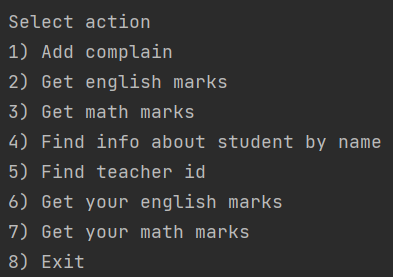


Рисунок 2 - Функции студентa

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.

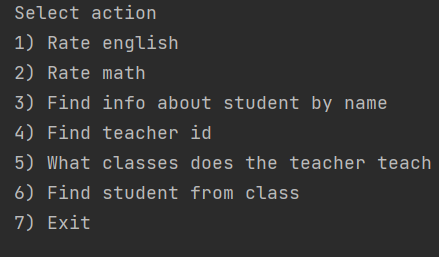


Рисунок 3 - Функции учителя

Родители.

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

На рисунке 4 изображены функции родителя.

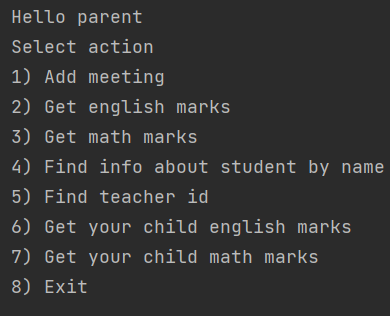
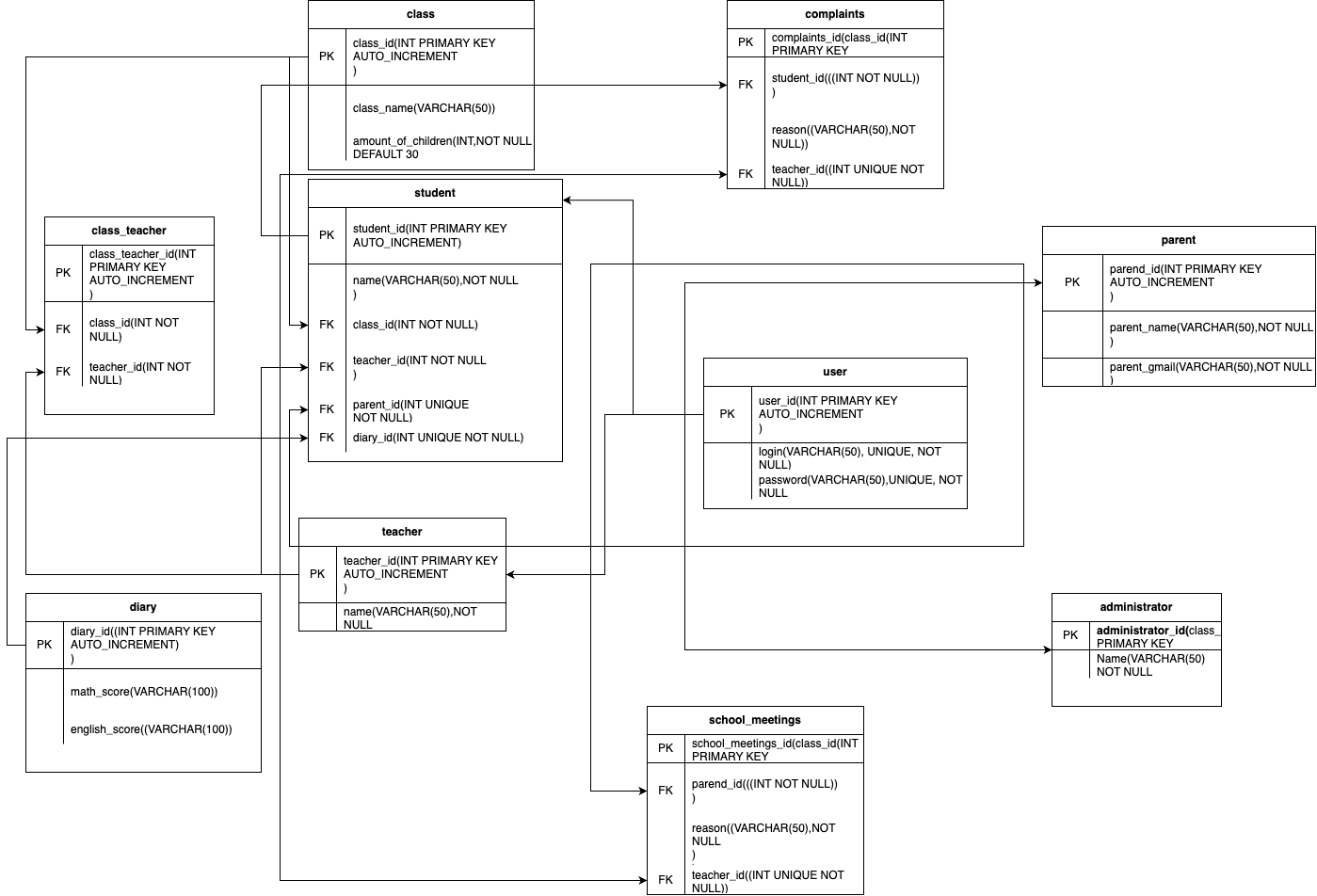


Рисунок 4 - Функции родителя

**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 учаться 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 учаться 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 Найдем 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 преподы

SELECT teacher.name,class\_id

FROM

teacher LEFT JOIN class\_teacher

ON teacher.teacher\_id = class\_teacher.teacher\_id

8 Каждому преподу 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\_name,"\_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(NEW.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(NEW.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;