Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра программного обеспечения информационных технологий
Дисциплина: Базы данных (БД)

ОТЧЕТ
по лабораторной работе №7

Выполнил студент: гр. 851006	Верещагин Н.В.
Проверил:	Фадеева Е.Е.

Минск 2021

1. Процедуры:

Удалить все расписание до указанной даты:

```
CREATE PROCEDURE CLEAR_SCHEDULE_TO_DATE(to_date DATETIME)
BEGIN
    DELETE
        FROM
            schedule
        WHERE
            to_date >= schedule.sch_time_end
END;
```

Событие для очистки истории каждый месяц:

```
CREATE EVENT CLEAR_OLD_SCHEDULE
    ON schedule
    EVERY 1 MONTH
    STARTS '2021-01-01 00:00:00'
    DO
    CALL CLEAR_SCHEDULE_TO_DATE(CURRENT_TIME());
```

Удалить всю историю для выбранного пользователя:

```
CREATE PROCEDURE CLEAR_ALL_HISTORY_FOR_USER(to_user_id INTEGER UNSIGNED)
BEGIN
    DELETE
        FROM
            edit_history
        WHERE
            to_user_id = edit_history.use_id
END;
```

Удалить новости до указанной даты:

```
CREATE PROCEDURE CLEAR_NEWS_TO_DATE(to_date DATETIME)
BEGIN
    DELETE
        FROM
            news
        WHERE
            to_date >= news.new_time
END;
```

Установить дату для курса:

```
CREATE PROCEDURE SET_DATES_FOR_COURSE(course_id INTEGER UNSIGNED, date_from DATE,
 date_to DATE)
BEGIN
```

```
    UPDATE
        course
    SET
        course.cou_date_from = date_from,
        course.cou_date_to = date_to
    WHERE
        course.cou_id = course_id
END;
```

2. Функции:

Получить все пары в диапазоне времени:

```
CREATE FUNCTION GET_ALL_CLASSES_BETWEEN_TIME(from_time DATETIME, to_time DATETIME
)
RETURN TABLE
AS
    RETURN
        SELECT
            class.*
        FROM
            class
            INNER JOIN schedule ON class.cla_id = schedule.cla_id
        WHERE
            (from_time BETWEEN schedule.sch_time_start AND schedule.sch_time_end)
 OR
            (to_time BETWEEN schedule.sch_time_start AND schedule.sch_time_end)
```

Получить все пары в указанном месте:

```
CREATE FUNCTION GET_ALL_CLASSES_IN_PLACE(place_name VARCHAR(350))
RETURN TABLE
AS
    RETURN
        SELECT
            class.*
        FROM
            class
            INNER JOIN schedule ON class.cla_id = schedule.cla_id
            INNER JOIN place ON schedule.pla_id = place.pla_id
        WHERE
            place_name = place.pla_name
```

Получить все пары для преподавателя:

```
CREATE FUNCTION GET_ALL_CLASSES_FOR_TEACHER(teacher_id INTEGER UNSIGNED)
RETURN TABLE
AS
```

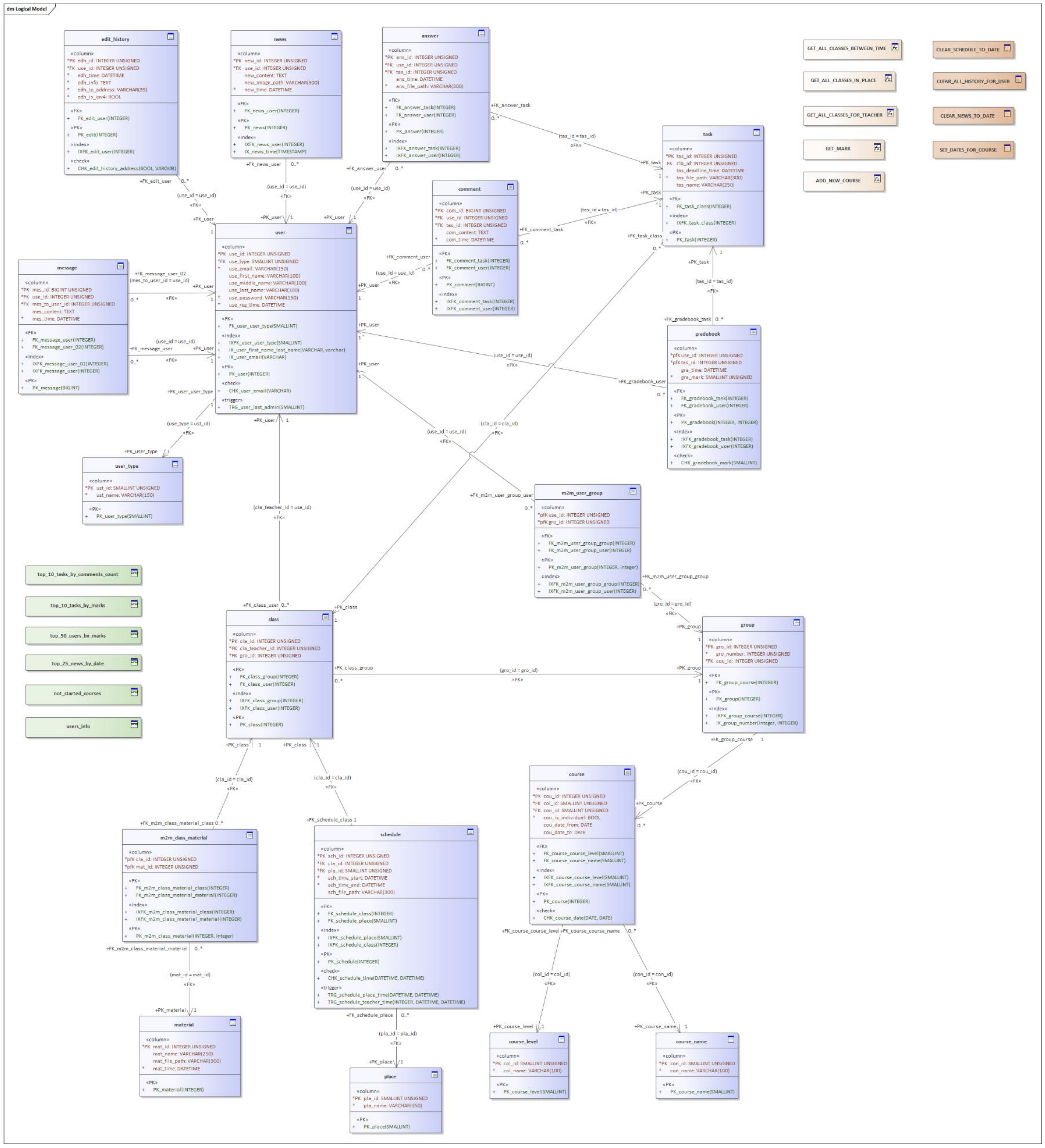
```
    RETURN
        SELECT
            *
        FROM
            class
        WHERE
            teacher_id = class.cla_teacher_id
```

Получить оценку за задачу:

```
CREATE FUNCTION GET_MARK(user_id INTEGER UNSIGNED, task_id INTEGER UNSIGNED)
RETURNS SMALLINT UNSIGNED
DETERMINISTIC
BEGIN
    RETURN
        SELECT
            gra_mark
        FROM
            gradebook
        WHERE
            user_id = gradebook.use_id AND task_id = gradebook.tas_id
END;
```

Добавить новый курс:

```
CREATE FUNCTION ADD_NEW_COURSE(level_id SMALLINT UNSIGNED, name_id SMALLINT UNSIG
NED, is_individual BOOL)
RETURNS INTEGER UNSIGNED
DETERMINISTIC
BEGIN
    RETURN
        INSERT INTO
            course (col_id, con_id, cou_is_individual)
            OUTPUT INSERTED.cou_id
        VALUES (level_id, name_id, is_individual);
END;
```


dm Logical Model
edit_history
«column»
*PK edh_id: INTEGER UNSIGNED
*FK use_id: INTEGER UNSIGNED
* edh_time: DATETIME
* edh_info: TEXT
* edh_ip_address: VARCHAR(39)
* edh_is_ipv4: BOOL
«FK»
+ FK_edit_user(INTEGER)
«PK»
+ PK_edit(INTEGER)
«index»
+ IXFK_edit_user(INTEGER)
«check»
+ CHK_edit_history_address(BOOL, VARCHAR)
news
«column»
*PK new_id: INTEGER UNSIGNED
*FK use_id: INTEGER UNSIGNED
new_content: TEXT
new_image_path: VARCHAR(300)
* new_time: DATETIME
«FK»
+ FK_news_user(INTEGER)
«PK»
+ PK_news(INTEGER)
«index»
+ IXFK_news_user(INTEGER)
+ IX_news_time(TIMESTAMP)
answer
«column»
*PK ans_id: INTEGER UNSIGNED
*FK use_id: INTEGER UNSIGNED
*FK tas_id: INTEGER UNSIGNED
ans_time: DATETIME
* ans_file_path: VARCHAR(300)
«FK»
+ FK_answer_task(INTEGER)
+ FK_answer_user(INTEGER)
«PK»
+ PK_answer(INTEGER)
«index»
+ IXFK_answer_task(INTEGER)
+ IXFK_answer_user(INTEGER)
GET_ALL_CLASSES_BETWEEN_TIME
GET_ALL_CLASSES_IN_PLACE
GET_ALL_CLASSES_FOR_TEACHER
GET_MARK
ADD_NEW_COURSE
CLEAR_SCHEDULE_TO_DATE
CLEAR_ALL_HISTORY_FOR_USER
CLEAR_NEWS_TO_DATE
SET_DATES_FOR_COURSE
task
«column»
*PK tas_id: INTEGER UNSIGNED
FK cla_id: INTEGER UNSIGNED
tas_deadline_time: DATETIME
* tas_file_path: VARCHAR(300)
tas_name: VARCHAR(250)
«FK»
+ FK_task_class(INTEGER)
«index»
+ IXFK_task_class(INTEGER)
«PK»
+ PK_task(INTEGER)
comment
«column»
*PK com_id: BIGINT UNSIGNED
*FK use_id: INTEGER UNSIGNED
*FK tas_id: INTEGER UNSIGNED
com_content: TEXT
* com_time: DATETIME
«FK»
+ FK_comment_task(INTEGER)
+ FK_comment_user(INTEGER)
«PK»
+ PK_comment(BIGINT)
«index»
+ IXFK_comment_task(INTEGER)
+ IXFK_comment_user(INTEGER)
user
«column»
*PK use_id: INTEGER UNSIGNED
*FK use_type: SMALLINT UNSIGNED
* use_email: VARCHAR(150)
use_first_name: VARCHAR(100)
use_middle_name: VARCHAR(100)
use_last_name: VARCHAR(100)
* use_password: VARCHAR(150)
* use_reg_time: DATETIME
«FK»
+ FK_user_user_type(SMALLINT)
«index»
+ IXFK_user_user_type(SMALLINT)
+ IX_user_first_name_last_name(VARCHAR, varchar)
+ IX_user_email(VARCHAR)
«PK»
+ PK_user(INTEGER)
«check»
+ CHK_user_email(VARCHAR)
«trigger»
+ TRG_user_last_admin(SMALLINT)
message
«column»
*PK mes_id: BIGINT UNSIGNED
*FK use_id: INTEGER UNSIGNED
*FK mes_to_user_id: INTEGER UNSIGNED
mes_content: TEXT
* mes_time: DATETIME
«FK»
+ FK_message_user(INTEGER)
+ FK_message_user_02(INTEGER)
«index»
+ IXFK_message_user_02(INTEGER)
+ IXFK_message_user(INTEGER)
«PK»
+ PK_message(BIGINT)
+FK_message_user_02
(mes_to_user_id = use_id)
+FK_message_user
(use_id = use_id)
gradebook
«column»
*pfK use_id: INTEGER UNSIGNED
*pfK tas_id: INTEGER UNSIGNED
gra_time: DATETIME
* gra_mark: SMALLINT UNSIGNED
«FK»
+ FK_gradebook_task(INTEGER)
+ FK_gradebook_user(INTEGER)
«PK»
+ PK_gradebook(INTEGER, INTEGER)
«index»
+ IXFK_gradebook_task(INTEGER)
+ IXFK_gradebook_user(INTEGER)
«check»
+ CHK_gradebook_mark(SMALLINT)
user_type
«column»
*PK ust_id: SMALLINT UNSIGNED
* ust_name: VARCHAR(150)
«PK»
+ PK_user_type(SMALLINT)
(use_type = ust_id)
(cla_teacher_id = use_id)
m2m_user_group
«column»
*pfK use_id: INTEGER UNSIGNED
*pfK gro_id: INTEGER UNSIGNED
«FK»
+ FK_m2m_user_group_group(INTEGER)
+ FK_m2m_user_group_user(INTEGER)
«PK»
+ PK_m2m_user_group(INTEGER, integer)
«index»
+ IXFK_m2m_user_group_group(INTEGER)
+ IXFK_m2m_user_group_user(INTEGER)
top_10_tasks_by_comments_count
top_10_tasks_by_marks
top_50_users_by_marks
top_25_news_by_date
not_started_courses
users_info
class
«column»
*PK cla_id: INTEGER UNSIGNED
*FK cla_teacher_id: INTEGER UNSIGNED
*FK gro_id: INTEGER UNSIGNED
«FK»
+ FK_class_group(INTEGER)
+ FK_class_user(INTEGER)
«index»
+ IXFK_class_group(INTEGER)
+ IXFK_class_user(INTEGER)
«PK»
+ PK_class(INTEGER)
(gro_id = gro_id)
group
«column»
*PK gro_id: INTEGER UNSIGNED
* gro_number: INTEGER UNSIGNED
*FK cou_id: INTEGER UNSIGNED
«FK»
+ FK_group_course(INTEGER)
«PK»
+ PK_group(INTEGER)
«index»
+ IXFK_group_course(INTEGER)
+ IX_group_number(integer, INTEGER)
(cou_id = cou_id)
course
«column»
*PK cou_id: INTEGER UNSIGNED
*FK col_id: SMALLINT UNSIGNED
*FK con_id: SMALLINT UNSIGNED
* cou_is_individual: BOOL
cou_date_from: DATE
cou_date_to: DATE
«FK»
+ FK_course_course_level(SMALLINT)
+ FK_course_course_name(SMALLINT)
«index»
+ IXFK_course_course_level(SMALLINT)
+ IXFK_course_course_name(SMALLINT)
«PK»
+ PK_course(INTEGER)
«check»
+ CHK_course_date(DATE, DATE)
(cla_id = cla_id)
m2m_class_material
«column»
*pfK cla_id: INTEGER UNSIGNED
*pfK mat_id: INTEGER UNSIGNED
«FK»
+ FK_m2m_class_material_class(INTEGER)
+ FK_m2m_class_material_material(INTEGER)
«index»
+ IXFK_m2m_class_material_class(INTEGER)
+ IXFK_m2m_class_material_material(INTEGER)
«PK»
+ PK_m2m_class_material(INTEGER, Integer)
schedule
«column»
*PK sch_id: INTEGER UNSIGNED
*FK cla_id: INTEGER UNSIGNED
*FK pla_id: SMALLINT UNSIGNED
* sch_time_start: DATETIME
* sch_time_end: DATETIME
sch_file_path: VARCHAR(300)
«FK»
+ FK_schedule_class(INTEGER)
+ FK_schedule_place(SMALLINT)
«index»
+ IXFK_schedule_place(SMALLINT)
+ IXFK_schedule_class(INTEGER)
«PK»
+ PK_schedule(INTEGER)
«check»
+ CHK_schedule_time(DATETIME, DATETIME)
«trigger»
+ TRG_schedule_place_time(DATETIME, DATETIME)
+ TRG_schedule_teacher_time(INTEGER, DATETIME, DATETIME)
(mat_id = mat_id)
(pla_id = pla_id)
(col_id = col_id)
(con_id = con_id)
material
«column»
*PK mat_id: INTEGER UNSIGNED
mat_name: VARCHAR(250)
mat_file_path: VARCHAR(300)
* mat_time: DATETIME
«PK»
+ PK_material(INTEGER)
place
«column»
*PK pla_id: SMALLINT UNSIGNED
* pla_name: VARCHAR(350)
«PK»
+ PK_place(SMALLINT)
course_level
«column»
*PK col_id: SMALLINT UNSIGNED
* col_name: VARCHAR(100)
«PK»
+ PK_course_level(SMALLINT)
course_name
«column»
*PK con_id: SMALLINT UNSIGNED
* con_name: VARCHAR(100)
«PK»
+ PK_course_name(SMALLINT)

Код:
```
/* ---------------------------------------------------- */
/*  Generated by Enterprise Architect Version 15.0      */
/*  Created On : 26-Apr-2021 6:08:35 PM                 */
/*  DBMS       : MySql                      */
/* ---------------------------------------------------- */

SET FOREIGN_KEY_CHECKS=0
;
/* Drop Views */

DROP VIEW IF EXISTS `not_started_courses` CASCADE
;

DROP VIEW IF EXISTS `top_10_tasks_by_comments_count` CASCADE
;

DROP VIEW IF EXISTS `top_10_tasks_by_marks` CASCADE
;

DROP VIEW IF EXISTS `top_25_news_by_date` CASCADE
;

DROP VIEW IF EXISTS `top_50_users_by_marks` CASCADE
;

DROP VIEW IF EXISTS `users_info` CASCADE
;

/* Drop Tables */

DROP TABLE IF EXISTS `answer` CASCADE
;

DROP TABLE IF EXISTS `class` CASCADE
;

DROP TABLE IF EXISTS `comment` CASCADE
;

DROP TABLE IF EXISTS `course` CASCADE
;

DROP TABLE IF EXISTS `course_level` CASCADE
;
```

```sql
DROP TABLE IF EXISTS `course_name` CASCADE
;

DROP TABLE IF EXISTS `edit_history` CASCADE
;

DROP TABLE IF EXISTS `gradebook` CASCADE
;

DROP TABLE IF EXISTS `group` CASCADE
;

DROP TABLE IF EXISTS `m2m_class_material` CASCADE
;

DROP TABLE IF EXISTS `m2m_user_group` CASCADE
;

DROP TABLE IF EXISTS `material` CASCADE
;

DROP TABLE IF EXISTS `message` CASCADE
;

DROP TABLE IF EXISTS `news` CASCADE
;

DROP TABLE IF EXISTS `place` CASCADE
;

DROP TABLE IF EXISTS `schedule` CASCADE
;

DROP TABLE IF EXISTS `task` CASCADE
;

DROP TABLE IF EXISTS `user` CASCADE
;

DROP TABLE IF EXISTS `user_type` CASCADE
;

/* Drop Functions */
```

```
DROP FUNCTION IF EXISTS `ADD_NEW_COURSE`
;

DROP FUNCTION IF EXISTS `GET_ALL_CLASSES_BETWEEN_TIME`
;

DROP FUNCTION IF EXISTS `GET_ALL_CLASSES_FOR_TEACHER`
;

DROP FUNCTION IF EXISTS `GET_ALL_CLASSES_IN_PLACE`
;

DROP FUNCTION IF EXISTS `GET_MARK`
;

/* Create Functions */

DELIMITER //
CREATE FUNCTION ADD_NEW_COURSE(level_id SMALLINT UNSIGNED, name_id SMALLINT UNSIG
NED, is_individual BOOL)
RETURNS INTEGER UNSIGNED
DETERMINISTIC
BEGIN
    RETURN
        INSERT INTO
            course (col_id, con_id, cou_is_individual)
            OUTPUT INSERTED.cou_id
        VALUES (level_id, name_id, is_individual);
END;
//
DELIMITER ;

;

DELIMITER //
CREATE FUNCTION GET_ALL_CLASSES_BETWEEN_TIME(from_time DATETIME, to_time DATETIME
)
RETURN TABLE
AS
    RETURN
        SELECT
            class.*
        FROM
            class
            INNER JOIN schedule ON class.cla_id = schedule.cla_id
        WHERE
            (from_time BETWEEN schedule.sch_time_start AND schedule.sch_time_end)
 OR
            (to_time BETWEEN schedule.sch_time_start AND schedule.sch_time_end)
//
DELIMITER ;

;

DELIMITER //
CREATE FUNCTION GET_ALL_CLASSES_FOR_TEACHER(teacher_id INTEGER UNSIGNED)
RETURN TABLE
AS
    RETURN
        SELECT
            *
        FROM
            class
        WHERE
            teacher_id = class.cla_teacher_id
//
DELIMITER ;

;

DELIMITER //
CREATE FUNCTION GET_ALL_CLASSES_IN_PLACE(place_name VARCHAR(350))
RETURN TABLE
AS
    RETURN
        SELECT
            class.*
        FROM
            class
            INNER JOIN schedule ON class.cla_id = schedule.cla_id
            INNER JOIN place ON schedule.pla_id = place.pla_id
        WHERE
            place_name = place.pla_name
//
DELIMITER ;

;

DELIMITER //
CREATE FUNCTION GET_MARK(user_id INTEGER UNSIGNED, task_id INTEGER UNSIGNED)
```

```
RETURNS SMALLINT UNSIGNED
DETERMINISTIC
BEGIN
    RETURN
        SELECT
            gra_mark
        FROM
            gradebook
        WHERE
            user_id = gradebook.use_id AND task_id = gradebook.tas_id
END;
//
DELIMITER ;

;

/* Create Tables */

CREATE TABLE `answer`
(
    `ans_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `use_id` INT UNSIGNED NOT NULL,
    `tas_id` INT UNSIGNED NOT NULL,
    `ans_time` DATETIME NULL,
    `ans_file_path` VARCHAR(300) NOT NULL COMMENT 'The path to the file with the
answer',
    CONSTRAINT `PK_answer` PRIMARY KEY (`ans_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `class`
(
    `cla_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `cla_teacher_id` INT UNSIGNED NOT NULL,
    `gro_id` INT UNSIGNED NOT NULL,
    CONSTRAINT `PK_class` PRIMARY KEY (`cla_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `comment`
(
    `com_id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `use_id` INT UNSIGNED NOT NULL,
    `tas_id` INT UNSIGNED NOT NULL,
    `com_content` TEXT NULL,
    `com_time` DATETIME NOT NULL,
    CONSTRAINT `PK_comment` PRIMARY KEY (`com_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `course`
(
    `cou_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `col_id` SMALLINT UNSIGNED NOT NULL,
    `con_id` SMALLINT UNSIGNED NOT NULL,
    `cou_is_individual` BOOL NOT NULL COMMENT 'Flag that indicates whether the co
urse is individual or group',
    `cou_date_from` DATE NULL,
    `cou_date_to` DATE NULL,
    CONSTRAINT `PK_course` PRIMARY KEY (`cou_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `course_level`
(
    `col_id` SMALLINT UNSIGNED NOT NULL,
    `col_name` VARCHAR(100) NOT NULL,
    CONSTRAINT `PK_course_level` PRIMARY KEY (`col_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `course_name`
(
    `con_id` SMALLINT UNSIGNED NOT NULL,
    `con_name` VARCHAR(100) NOT NULL,
    CONSTRAINT `PK_course_name` PRIMARY KEY (`con_id` ASC)
)
COLLATE utf8_general_ci

;
```

```sql
CREATE TABLE `edit_history`
(
    `edh_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `use_id` INT UNSIGNED NOT NULL,
    `edh_time` DATETIME NOT NULL,
    `edh_info` TEXT NOT NULL COMMENT 'Change information is stored in JSON format
',
    `edh_ip_address` VARCHAR(39) NOT NULL COMMENT 'It stores either an ipv4 addre
ss or an ipv6 address',
    `edh_is_ipv4` BOOL NOT NULL COMMENT 'A flag that indicates whether the addres
s is stored in ipv4 format or in ipv6 format',
    CONSTRAINT `PK_edit` PRIMARY KEY (`edh_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `gradebook`
(
    `use_id` INT UNSIGNED NOT NULL,
    `tas_id` INT UNSIGNED NOT NULL,
    `gra_time` DATETIME NULL,
    `gra_mark` SMALLINT UNSIGNED NOT NULL,
    CONSTRAINT `PK_gradebook` PRIMARY KEY (`use_id` ASC, `tas_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `group`
(
    `gro_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `gro_number` INT UNSIGNED NOT NULL,
    `cou_id` INT UNSIGNED NOT NULL,
    CONSTRAINT `PK_group` PRIMARY KEY (`gro_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `m2m_class_material`
(
    `cla_id` INT UNSIGNED NOT NULL,
    `mat_id` INT UNSIGNED NOT NULL,
    CONSTRAINT `PK_m2m_class_material` PRIMARY KEY (`cla_id` ASC, `mat_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `m2m_user_group`
(
    `use_id` INT UNSIGNED NOT NULL,
    `gro_id` INT UNSIGNED NOT NULL,
    CONSTRAINT `PK_m2m_user_group` PRIMARY KEY (`gro_id` ASC, `use_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `material`
(
    `mat_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `mat_name` VARCHAR(250) NULL,
    `mat_file_path` VARCHAR(300) NULL COMMENT 'File path with material',
    `mat_time` DATETIME NOT NULL,
    CONSTRAINT `PK_material` PRIMARY KEY (`mat_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `message`
(
    `mes_id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `use_id` INT UNSIGNED NOT NULL,
    `mes_to_user_id` INT UNSIGNED NOT NULL COMMENT 'The id of the user to whom th
e message is intended is stored here',
    `mes_content` TEXT NULL,
    `mes_time` DATETIME NOT NULL,
    CONSTRAINT `PK_message` PRIMARY KEY (`mes_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `news`
(
    `new_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `use_id` INT UNSIGNED NOT NULL,
    `new_content` TEXT NULL,
    `new_image_path` VARCHAR(300) NULL COMMENT 'The path to the picture',
    `new_time` DATETIME NOT NULL,
    CONSTRAINT `PK_news` PRIMARY KEY (`new_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `place`
(
    `pla_id` SMALLINT UNSIGNED NOT NULL,
    `pla_name` VARCHAR(350) NOT NULL COMMENT 'Name of the venue',
    CONSTRAINT `PK_place` PRIMARY KEY (`pla_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `schedule`
(
    `sch_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `cla_id` INT UNSIGNED NOT NULL,
    `pla_id` SMALLINT UNSIGNED NOT NULL,
    `sch_time_start` DATETIME NOT NULL,
    `sch_time_end` DATETIME NOT NULL,
    `sch_file_path` VARCHAR(300) NULL COMMENT 'Path to a file with additional inf
ormation for the lesson',
    CONSTRAINT `PK_schedule` PRIMARY KEY (`sch_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `task`
(
    `tas_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `cla_id` INT UNSIGNED NULL,
    `tas_deadline_time` DATETIME NULL COMMENT 'Time until which you can send a re
sponse to the assignment',
    `tas_file_path` VARCHAR(300) NOT NULL COMMENT 'The path to the task file',
    `tas_name` VARCHAR(250) NULL,
    CONSTRAINT `PK_task` PRIMARY KEY (`tas_id` ASC)
)
```

```
COLLATE utf8_general_ci

;

CREATE TABLE `user`
(
    `use_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `use_type` SMALLINT UNSIGNED NOT NULL,
    `use_email` VARCHAR(150) NOT NULL,
    `use_first_name` VARCHAR(100) NULL,
    `use_middle_name` VARCHAR(100) NULL,
    `use_last_name` VARCHAR(100) NULL,
    `use_password` VARCHAR(150) NOT NULL,
    `use_reg_time` DATETIME NOT NULL,
    CONSTRAINT `PK_user` PRIMARY KEY (`use_id` ASC)
)
COLLATE utf8_general_ci

;

CREATE TABLE `user_type`
(
    `ust_id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    `ust_name` VARCHAR(150) NOT NULL,
    CONSTRAINT `PK_user_type` PRIMARY KEY (`ust_id` ASC)
)
COLLATE utf8_general_ci

;

/* Create Primary Keys, Indexes, Uniques, Checks */

ALTER TABLE `answer`
 ADD INDEX `IXFK_answer_task` (`tas_id` ASC)
;

ALTER TABLE `answer`
 ADD INDEX `IXFK_answer_user` (`use_id` ASC)
;

ALTER TABLE `class`
 ADD INDEX `IXFK_class_group` (`gro_id` ASC)
;

ALTER TABLE `class`
```

```
 ADD INDEX `IXFK_class_user` (`cla_teacher_id` ASC)
;

ALTER TABLE `comment`
 ADD INDEX `IXFK_comment_task` (`tas_id` ASC)
;

ALTER TABLE `comment`
 ADD INDEX `IXFK_comment_user` (`use_id` ASC)
;

ALTER TABLE `course`
 ADD CONSTRAINT `CHK_course_date` CHECK (cou_date_from <= cou_date_to)
;

ALTER TABLE `course`
 ADD INDEX `IXFK_course_course_level` (`col_id` ASC)
;

ALTER TABLE `course`
 ADD INDEX `IXFK_course_course_name` (`con_id` ASC)
;

ALTER TABLE `edit_history`
 ADD CONSTRAINT `CHK_edit_history_address` CHECK ((edh_is_ipv4 AND IS_IPV4(edh_ip
_address)) OR
(NOT edh_is_ipv4 AND IS_IPV6(edh_ip_address))
)
;

ALTER TABLE `edit_history`
 ADD INDEX `IXFK_edit_user` (`use_id` ASC)
;

ALTER TABLE `gradebook`
 ADD CONSTRAINT `CHK_gradebook_mark` CHECK (gra_mark <= 10)
;

ALTER TABLE `gradebook`
 ADD INDEX `IXFK_gradebook_task` (`tas_id` ASC)
;

ALTER TABLE `gradebook`
 ADD INDEX `IXFK_gradebook_user` (`use_id` ASC)
;
```

```
ALTER TABLE `group`
 ADD INDEX `IXFK_group_course` (`cou_id` ASC)
;

ALTER TABLE `group`
 ADD INDEX `IX_group_number` (`gro_number` ASC, `cou_id` ASC)
;

ALTER TABLE `m2m_class_material`
 ADD INDEX `IXFK_m2m_class_material_class` (`cla_id` ASC)
;

ALTER TABLE `m2m_class_material`
 ADD INDEX `IXFK_m2m_class_material_material` (`mat_id` ASC)
;

ALTER TABLE `m2m_user_group`
 ADD INDEX `IXFK_m2m_user_group_group` (`gro_id` ASC)
;

ALTER TABLE `m2m_user_group`
 ADD INDEX `IXFK_m2m_user_group_user` (`use_id` ASC)
;

ALTER TABLE `message`
 ADD INDEX `IXFK_message_user_02` (`mes_to_user_id` ASC)
;

ALTER TABLE `message`
 ADD INDEX `IXFK_message_user` (`use_id` ASC)
;

ALTER TABLE `news`
 ADD INDEX `IXFK_news_user` (`use_id` ASC)
;

ALTER TABLE `news`
 ADD INDEX `IX_news_time` (`new_time` ASC)
;

ALTER TABLE `schedule`
 ADD CONSTRAINT `CHK_schedule_time` CHECK (sch_time_start <= sch_time_end)
;
```

```
ALTER TABLE `schedule`
 ADD INDEX `IXFK_schedule_place` (`pla_id` ASC)
;

ALTER TABLE `schedule`
 ADD INDEX `IXFK_schedule_class` (`cla_id` ASC)
;

DELIMITER //
CREATE TRIGGER 'TRG_schedule_place_time' BEFORE INSERT ON schedule FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT
            *
        FROM
            schedule
        WHERE
            schedule.pla_id = NEW.pla_id AND
            (
                NEW.sch_time_start BETWEEN schedule.sch_time_start AND schedule.s
ch_time_end
            ) OR (
                NEW.sch_time_end BETWEEN schedule.sch_time_start AND schedule.sch
_time_end
            )
    )
    THEN
        SIGNAL SQLSTATE '45002'
        SET MESSAGE_TEXT = 'You cannot schedule another activity during this time
.'
        MYSQL_ERRNO = 1002
    END IF;
END;
//
DELIMITER ;
;

DELIMITER //
CREATE TRIGGER 'TRG_schedule_teacher_time' BEFORE INSERT ON schedule FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT
            *
        FROM
            schedule
            INNER JOIN class ON class.cla_id = schedule.cla_id
        WHERE
            class.cla_teacher_id = NEW.cla_teacher_id AND
            (
                NEW.sch_time_start BETWEEN schedule.sch_time_start AND schedule.s
ch_time_end
            ) OR (
                NEW.sch_time_end BETWEEN schedule.sch_time_start AND schedule.sch
_time_end
            )
    )
    THEN
        SIGNAL SQLSTATE '45001'
        SET MESSAGE_TEXT = 'You cannot assign several lessons to the same instruc
tor at the same time.'
        MYSQL_ERRNO = 1001
    END IF;
END;
//
DELIMITER ;
;

ALTER TABLE `task`
 ADD INDEX `IXFK_task_class` (`cla_id` ASC)
;

ALTER TABLE `user`
 ADD CONSTRAINT `CHK_user_email` CHECK (use_email LIKE '%_@__%.__%')
;

ALTER TABLE `user`
 ADD INDEX `IXFK_user_user_type` (`use_type` ASC)
;

ALTER TABLE `user`
 ADD INDEX `IX_user_first_name_last_name` (`use_first_name` ASC, `use_last_name`
ASC)
;

ALTER TABLE `user`
 ADD INDEX `IX_user_email` (`use_email` ASC)
;

DELIMITER //
CREATE TRIGGER 'TRG_user_last_admin' BEFORE DELETE ON user FOR EACH ROW
```

```
BEGIN
    IF (
        SELECT
            *
        FROM
            user_type
            INNER JOIN user ON user.ust_type = user_type.ust_id
        WHERE
            (user_type.ust_name = 'admin') AND
            (user.ust_type = user_type.ust_id)
        < 2
    )
    THEN
        SIGNAL SQLSTATE '45003'
        SET MESSAGE_TEXT = 'You cannot remove the last admin.'
        MYSQL_ERRNO = 1003
    END IF;
END;
//
DELIMITER ;
;

/* Create Foreign Key Constraints */

ALTER TABLE `answer`
 ADD CONSTRAINT `FK_answer_task`
    FOREIGN KEY (`tas_id`) REFERENCES `task` (`tas_id`) ON DELETE Restrict ON UPD
ATE Restrict
;

ALTER TABLE `answer`
 ADD CONSTRAINT `FK_answer_user`
    FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict ON UPD
ATE Restrict
;

ALTER TABLE `class`
 ADD CONSTRAINT `FK_class_group`
    FOREIGN KEY (`gro_id`) REFERENCES `group` (`gro_id`) ON DELETE Restrict ON UP
DATE Restrict
;

ALTER TABLE `class`
 ADD CONSTRAINT `FK_class_user`
```

```
    FOREIGN KEY (`cla_teacher_id`) REFERENCES `user` (`use_id`) ON DELETE Restric
t ON UPDATE Restrict
;

ALTER TABLE `comment`
 ADD CONSTRAINT `FK_comment_task`
    FOREIGN KEY (`tas_id`) REFERENCES `task` (`tas_id`) ON DELETE Restrict ON UPD
ATE Restrict
;

ALTER TABLE `comment`
 ADD CONSTRAINT `FK_comment_user`
    FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict ON UPD
ATE Restrict
;

ALTER TABLE `course`
 ADD CONSTRAINT `FK_course_course_level`
    FOREIGN KEY (`col_id`) REFERENCES `course_level` (`col_id`) ON DELETE Restric
t ON UPDATE Restrict
;

ALTER TABLE `course`
 ADD CONSTRAINT `FK_course_course_name`
    FOREIGN KEY (`con_id`) REFERENCES `course_name` (`con_id`) ON DELETE Restrict
 ON UPDATE Restrict
;

ALTER TABLE `edit_history`
 ADD CONSTRAINT `FK_edit_user`
    FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Cascade ON UPDA
TE Cascade
;

ALTER TABLE `gradebook`
 ADD CONSTRAINT `FK_gradebook_task`
    FOREIGN KEY (`tas_id`) REFERENCES `task` (`tas_id`) ON DELETE Restrict ON UPD
ATE Restrict
;

ALTER TABLE `gradebook`
 ADD CONSTRAINT `FK_gradebook_user`
    FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict ON UPD
ATE Restrict
;
```

```sql
ALTER TABLE `group`
 ADD CONSTRAINT `FK_group_course`
    FOREIGN KEY (`cou_id`) REFERENCES `course` (`cou_id`) ON DELETE Restrict ON U
PDATE Restrict
;

ALTER TABLE `m2m_class_material`
 ADD CONSTRAINT `FK_m2m_class_material_class`
    FOREIGN KEY (`cla_id`) REFERENCES `class` (`cla_id`) ON DELETE Cascade ON UPD
ATE Cascade
;

ALTER TABLE `m2m_class_material`
 ADD CONSTRAINT `FK_m2m_class_material_material`
    FOREIGN KEY (`mat_id`) REFERENCES `material` (`mat_id`) ON DELETE Cascade ON
UPDATE Cascade
;

ALTER TABLE `m2m_user_group`
 ADD CONSTRAINT `FK_m2m_user_group_group`
    FOREIGN KEY (`gro_id`) REFERENCES `group` (`gro_id`) ON DELETE Cascade ON UPD
ATE Cascade
;

ALTER TABLE `m2m_user_group`
 ADD CONSTRAINT `FK_m2m_user_group_user`
    FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Cascade ON UPDA
TE Cascade
;

ALTER TABLE `message`
 ADD CONSTRAINT `FK_message_user`
    FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict ON UPD
ATE Restrict
;

ALTER TABLE `message`
 ADD CONSTRAINT `FK_message_user_02`
    FOREIGN KEY (`mes_to_user_id`) REFERENCES `user` (`use_id`) ON DELETE Restric
t ON UPDATE Restrict
;

ALTER TABLE `news`
 ADD CONSTRAINT `FK_news_user`
```

```
    FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict ON UPD
ATE Restrict
;

ALTER TABLE `schedule`
 ADD CONSTRAINT `FK_schedule_class`
    FOREIGN KEY (`cla_id`) REFERENCES `class` (`cla_id`) ON DELETE Cascade ON UPD
ATE Cascade
;

ALTER TABLE `schedule`
 ADD CONSTRAINT `FK_schedule_place`
    FOREIGN KEY (`pla_id`) REFERENCES `place` (`pla_id`) ON DELETE Restrict ON UP
DATE Cascade
;

ALTER TABLE `task`
 ADD CONSTRAINT `FK_task_class`
    FOREIGN KEY (`cla_id`) REFERENCES `class` (`cla_id`) ON DELETE Restrict ON UP
DATE Restrict
;

ALTER TABLE `user`
 ADD CONSTRAINT `FK_user_user_type`
    FOREIGN KEY (`use_type`) REFERENCES `user_type` (`ust_id`) ON DELETE Restrict
 ON UPDATE Restrict
;

SET FOREIGN_KEY_CHECKS=1
;
/* Create Views */

CREATE OR REPLACE VIEW `not_started_courses` AS
SELECT
    course_name.con_name AS name,
    course_level.col_name AS language_level,
    course.cou_is_individual AS is_individual,
    course.cou_date_from AS date_from,
    course.cou_date_to AS date_to
FROM
    course
    INNER JOIN course_level AS col ON con.col_id = course.col_id
    INNER JOIN course_name AS con ON con.con_id = course.con_id
WHERE
    date_from > NOW()
```

```
ORDER BY
    DATE_FORMAT(date_from, '%m%d') DESC;
;


CREATE OR REPLACE VIEW `top_10_tasks_by_comments_count` AS
SELECT
    task.*,
    COUNT(comment.com_id) AS comments_count
FROM
    task
    INNER JOIN comment ON comment.tas_id = task.tas_id
GROUP BY task.tas_id
ORDER BY comments_count
LIMIT 10;
;


CREATE OR REPLACE VIEW `top_10_tasks_by_marks` AS
SELECT
    task.*,
    SUM(gradebook.gra_mark) AS marks_sum
FROM
    task
    INNER JOIN gradebook ON gradebook.tas_id = task.tas_id
GROUP BY task.tas_id
ORDER BY marks_sum
LIMIT 10;
;


CREATE OR REPLACE VIEW `top_25_news_by_date` AS
SELECT * FROM news ORDER BY DATE_FORMAT(new_time, '%m%d%H%i%s');
;


CREATE OR REPLACE VIEW `top_50_users_by_marks` AS
SELECT
    user.*,
    AVG(gradebook.gra_mark) AS average_mark
FROM
    user
    INNER JOIN gradebook ON gradebook.tas_id = user.tas_id
GROUP BY user.tas_id
ORDER BY average_mark
LIMIT 50;
```

```sql
;


CREATE OR REPLACE VIEW `users_info` AS
SELECT
    user.*,
    COUNT(message.use_id) AS messages_count,
    COUNT(answer.use_id) AS answers_count,
    COUNT(comment.use_id) AS comments_count,
    AVG(gradebook.gra_mark) AS average_mark
FROM
    user
    INNER JOIN message ON message.use_id = user.use_id
    INNER JOIN answer ON answer.use_id = user.use_id
    INNER JOIN comment ON comment.use_id = user.use_id
    INNER JOIN gradebook ON gradebook.use_id = user.use_id
GROUP BY user.use_id
ORDER BY average_mark;
;
```