

ГУАП

КАФЕДРА № 43

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

старший преподаватель  
\_\_\_\_\_  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

Н.В. Путилова  
\_\_\_\_\_  
инициалы, фамилия

## ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8

Триггеры. Обеспечение активной целостности данных базы данных

по курсу: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4236

\_\_\_\_\_  
подпись, дата

Л. Мвале  
\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2024

**Цель работы:** Получение умений и навыков по проектированию и созданию триггеров баз данных, включая их использование для поддержания активной ссылочной целостности.

### **Описание задания**

Реализовать для своей базы данных триггеры для всех событий (insert, delete, update) до и после. (6 триггеров) Часть из которых будет обеспечивать ссылочную целостность, остальные могут иметь другое назначение из других предложенных, но не менее 2 различных. - Вычисление/поддержание в актуальном состоянии вычисляемых (производных) атрибутов - логирование (запись) изменений; - проверка корректности проводимых действий.). Вычисляемые поля можно добавить при необходимости.

### **Текст задания Вариант 15**

15. вузы для абитуриента: город, вуз, факультеты, направления, направленности, ЕГЭ которые нужно сдать, дата начала/конца приемной кампании.

(Направление -09.03.04 «Программная инженерия», Направленность — его конкретизация «Разработка программно-информационных систем», именно направленность закреплена за кафедрой и соответственно факультетом)

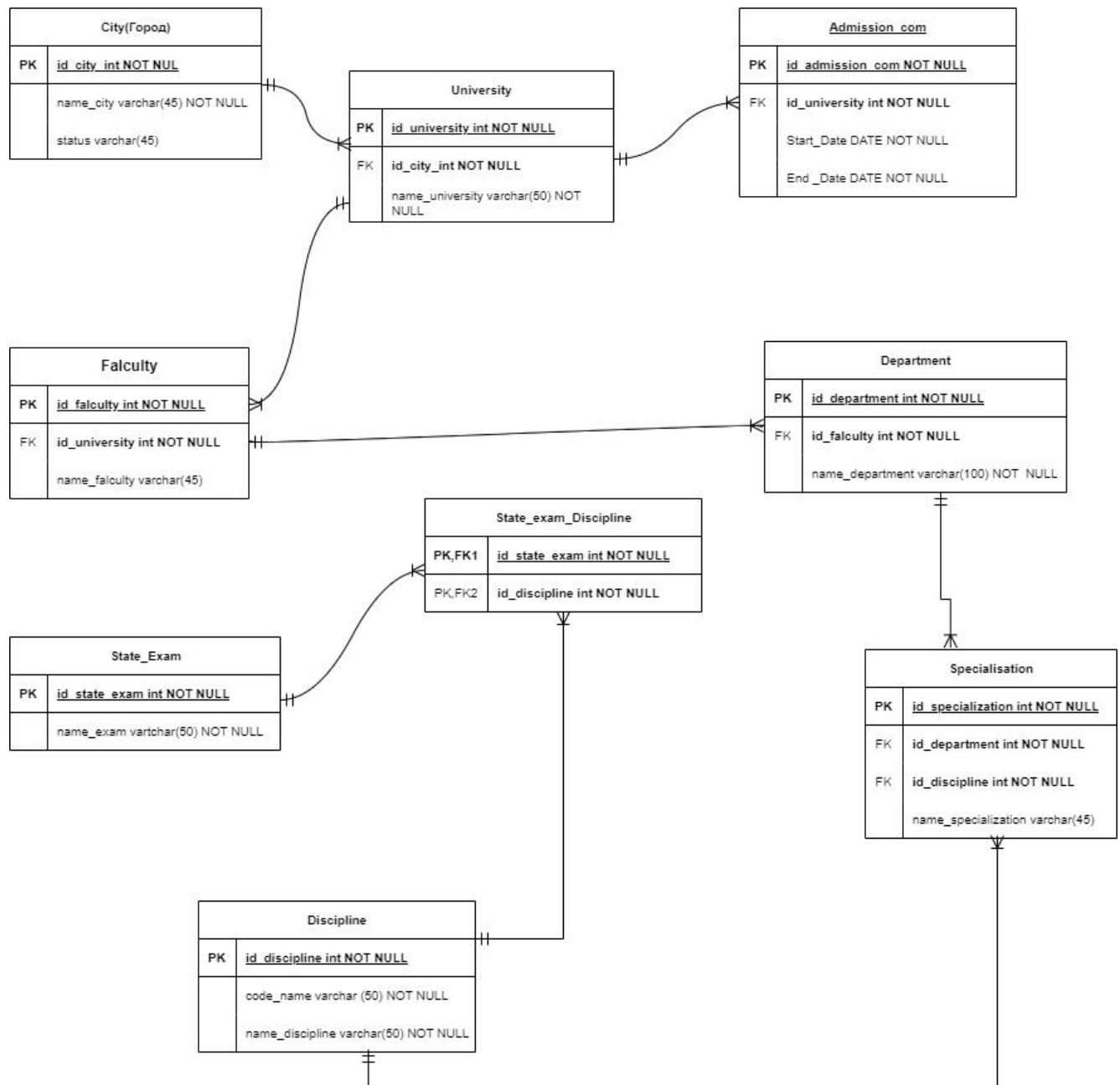
- а. направленности, в которых есть слово «систем», но оно не первое
- б. Кафедра, не принимающая ни на одну направленность
- в. направление, на которое надо сдавать математику и информатику
- г. факультет, принимающий на количество направлений больше среднего

д. город, в котором есть все укрупненные группы направлений и специальностей(УГСН) (первые 2 цифры номера специальности, т.е у 09.03.04 УГСН=09, а у 02.03.03-02)

е. вуз, с последним по алфавиту названием

ж. направление, на которое не надо сдавать ЕГЭ по математике, но надо по иностранному языку

### Физическую модель базы данных



## Назначение разработанных триггеров текстом:

### 1. Триггер before\_university\_insert

- **Тип:** BEFORE INSERT
- **Назначение:** Этот триггер проверяет корректность данных перед добавлением новой записи в таблицу University. Он проверяет:
  - Поле name\_university не должно быть пустым.
  - Значение id\_city не должно превышать 5.В случае нарушения условий триггер вызывает ошибку, предотвращая вставку некорректных данных. Это гарантирует целостность и корректность информации о университетах.

### 2. Триггер before\_university\_update

- **Тип:** BEFORE UPDATE
- **Назначение:** Этот триггер выполняет валидацию данных перед обновлением записи в таблице University. Он проверяет:
  - Поле name\_university не должно быть пустым.
  - Значение id\_city не должно превышать 5.В случае несоответствия триггер вызывает ошибку, предотвращая обновление данных. Это сохраняет качество и точность информации.

### 3. Триггер before\_university\_delete

- **Тип:** BEFORE DELETE
- **Назначение:** Триггер обеспечивает каскадное удаление связанных записей из других таблиц (Faculty, Admission\_com, Department, Specialization), которые связаны с удаляемым университетом. Это предотвращает оставление "осиротевших" записей в базе данных, поддерживая её согласованность.

### 4. Триггер after\_university\_insert

- **Тип:** AFTER INSERT
- **Назначение:** Этот триггер выполняется после добавления новой записи в таблицу University. Он:

- Вычисляет количество факультетов, связанных с добавленным университетом, используя функцию count\_faculties.
- Добавляет запись в таблицу univ\_log с информацией об ID университета, типе действия (INSERT) и количестве факультетов. Это позволяет отслеживать изменения и актуализировать информацию.

## 5. Триггер after\_university\_update

- **Тип:** AFTER UPDATE
- **Назначение:** Выполняется после обновления записи в таблице University. Он:
  - Вычисляет количество факультетов, связанных с обновлённым университетом.
  - Добавляет запись в таблицу univ\_log с ID университета, типом действия (UPDATE) и количеством факультетов. Это обеспечивает аудит изменений.

## 6. Триггер after\_university\_delete

- **Тип:** AFTER DELETE
- **Назначение:** После удаления записи из таблицы University, триггер добавляет запись в таблицу univ\_log с ID удалённого университета, типом действия (DELETE). Это фиксирует факт удаления для дальнейшего анализа.

## 7. Триггер before\_specialization\_insert

- **Тип:** BEFORE INSERT
- **Назначение:** Проверяет, что поле name\_specialization не пустое перед добавлением записи в таблицу Specialization. При нарушении условия вызывается ошибка, предотвращая добавление некорректных данных.

## 8. Триггер before\_specialization\_update

- **Тип:** BEFORE UPDATE

- **Назначение:** Проверяет, что поле name\_specialization не пустое перед обновлением записи. В случае нарушения триггер предотвращает обновление, поддерживая целостность данных.

## 9. Триггер before\_specialization\_delete

- **Тип:** BEFORE DELETE
- **Назначение:** Удаляет связанные записи из таблицы State\_Exam\_Discipline, которые связаны с удаляемой специализацией. Это предотвращает оставление несогласованных данных.

## 10. Триггер after\_specialization\_insert

- **Тип:** AFTER INSERT
- **Назначение:** После добавления новой записи в таблицу Specialization, триггер:
  - Считает общее количество специализаций в таблице.
  - Добавляет запись в таблицу specializ\_log с ID специализации, типом действия (INSERT) и количеством специализаций. Это помогает вести учёт и анализ изменений.

## 11. Триггер after\_specialization\_update

- **Тип:** AFTER UPDATE
- **Назначение:** Выполняется после обновления записи в таблице Specialization. Триггер:
  - Считает общее количество специализаций.
  - Добавляет запись в таблицу specializ\_log с ID специализации, типом действия (UPDATE) и количеством специализаций. Это поддерживает актуальность логов.

## 12. Триггер after\_specialization\_delete

- **Тип:** AFTER DELETE
- **Назначение:** После удаления записи из таблицы Specialization, триггер:
  - Считает общее количество оставшихся специализаций.

- Добавляет запись в таблицу `specializ_log` с ID удалённой специализации, типом действия (`DELETE`) и количеством оставшихся специализаций.  
Это фиксирует изменения для анализа.

## Скрипт для создания триггеров:

```
DELIMITER $$

-- Создаем функцию для подсчета количества факультетов в университете
CREATE FUNCTION count_faculti(p_university_id INT)
RETURNS INT -- Возвращаемое значение типа INT
reads sql data
BEGIN
    DECLARE faculty_count INT; -- Переменная для хранения количества
    факультетов
```

```
    -- Выполняем подсчет факультетов для указанного университета
    SELECT COUNT(*)
    INTO faculty_count
    FROM Faculty f
    JOIN University u ON f.id_university = u.id_university
    WHERE u.id_university = p_university_id;
```

```
    -- Возвращаем количество факультетов
    RETURN faculty_count;
END$$
```

```
DELIMITER ;
```

```
-- University Table Triggers
drop table if exists univ_log;
CREATE TABLE IF NOT EXISTS univ_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    university_id INT NOT NULL,
    action VARCHAR(10) NOT NULL,
    changetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
-- 1. Before Inserting
DELIMITER $$
```

```
CREATE TRIGGER before_university_insert
BEFORE INSERT ON University
FOR EACH ROW
BEGIN
    IF NEW.name_university = '' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'University name cannot be empty.';
    END IF;
    IF NEW.id_city > 5 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Invalid city ID';
    END IF;
END$$
```

```
DELIMITER ;
```

```
-- 2. Before Updating
DELIMITER $$
```



```

CREATE TRIGGER before_university_update
BEFORE UPDATE ON University
FOR EACH ROW
BEGIN
    IF NEW.name_university = '' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'University name cannot be empty.';
    END IF;
    IF NEW.id_city > 5 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Invalid city ID';
    END IF;
END$$

```

```

DELIMITER ;

```

```

DROP TRIGGER IF EXISTS before_university_delete;
-- 3. Before Deleting
DELIMITER $$

```

```

CREATE TRIGGER before_university_delete
BEFORE DELETE ON University
FOR EACH ROW
BEGIN
    -- Cascade delete related Faculty records
    DELETE FROM Faculty WHERE id_university = OLD.id_university;

```

```

    -- Cascade delete related Admission_com records
    DELETE FROM Admission_com WHERE id_university = OLD.id_university;

```

```

    -- Cascade delete related Department records
    DELETE FROM Department WHERE id_faculty IN (
        SELECT id_faculty FROM Faculty WHERE id_university =
OLD.id_university
    );

```

```

    -- Cascade delete related Specialization records
    DELETE FROM Specialization WHERE id_department IN (
        SELECT id_department FROM Department WHERE id_faculty IN (
            SELECT id_faculty FROM Faculty WHERE id_university =
OLD.id_university
        )
    );

```

```

END$$

```

```

DELIMITER ;

```

```

-- 4. After Inserting
DROP TRIGGER IF EXISTS after_university_insert;
DELIMITER $$

```

```

CREATE TRIGGER after_university_insert
AFTER INSERT ON University

```

```
FOR EACH ROW
BEGIN

    INSERT INTO univ_log (university_id, action, changetime)
    VALUES (NEW.id_university, 'INSERT', NOW());
END$$
```

```
DELIMITER ;
```

```
-- 5. After Update
DELIMITER $$
```

```
CREATE TRIGGER after_university_update
AFTER UPDATE ON Falculty
FOR EACH ROW
BEGIN
    DECLARE faculty_count INT;
```

```
-- Use the function to count faculties for the updated university
SET faculty_count = count_faculti(NEW.id_university);
```

```
-- Insert the log entry with the faculty count
INSERT INTO univ_log (university_id, action, changetime)
VALUES (NEW.id_university, CONCAT('UPDATE-', faculty_count), NOW());
END$$
```

```
DELIMITER ;
```

```
-- 6. After Delete
DROP TRIGGER IF EXISTS after_university_delete;
DELIMITER $$
```

```
CREATE TRIGGER after_university_delete
AFTER DELETE ON University
FOR EACH ROW
BEGIN
    INSERT INTO univ_log (university_id, action, changetime)
    VALUES (OLD.id_university, 'DELETE', CURRENT_TIMESTAMP);
END$$
```

```
DELIMITER ;
```

# SQL операторы и скриншоты наборов данных, иллюстрирующие работу триггеров

## Начальное состояние университетской таблицы

	id_university	name_university	id_city
▶	1	Харьковский Национальный Университет	1
	2	Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения	2
*	NULL	NULL	NULL

SELECT \* FROM univ\_log;

	log_id	university_id	action	changetime
▶	1	1	UPDATE-2	2024-12-24 22:32:47
	2	3	INSERT-0	2024-12-24 22:34:05
	3	2	UPDATE-3	2024-12-24 22:35:47
	4	1	DELETE	2024-12-24 23:39:04
	5	2	DELETE	2024-12-24 23:39:04
	6	3	DELETE	2024-12-24 23:39:04
	7	1	INSERT-0	2024-12-24 23:39:04
	8	2	INSERT-0	2024-12-24 23:39:04
*	NULL	NULL	NULL	NULL

1. Триггер: Проверка стоимости перед вставкой (before\_university\_insert)  
Вставка с корректной переменной (должен пройти успешно):

```
INSERT INTO University (name_university, id_city)
VALUES ('Новосибирский Государственный Университет', 2);
```

## Результаты

	id_university	name_university	id_city
▶	1	Харьковский Национальный Университет	1
	2	Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения	2
	3	Новосибирский Государственный Университет	2
*	NULL	NULL	NULL

2. Триггер: Проверка стоимости перед вставкой (before\_university\_insert)  
Вставка с пустым полем name\_university (должен дать ошибку):

```
INSERT INTO University (name_university, id_city)
VALUES ('', 2);
```

Никаких изменений в таблице

	id_university	name_university	id_city
▶	1	Харьковский Национальный Университет	1
	2	Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения	2
	3	Новосибирский Государственный Университет	2
*	NULL	NULL	NULL

3. Триггер: Проверка стоимости перед обновлением (before\_university\_update)  
Обновление с корректным значением name\_university(должен пройти успешно):

```
UPDATE University  
SET name_university = 'Харьковский Государственный Университет'  
WHERE id_university = 1;
```

Результаты, показывающие обновленное название в 1-м  
id\_university

	id_university	name_university	id_city
▶	1	Харьковский Государственный Университет	1
	2	Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения	2
	3	Новосибирский Государственный Университет	2
*	NULL	NULL	NULL

4. Триггер: Проверка стоимости перед обновлением (before\_university\_update)  
Обновление с пустым значением name\_university (должен дать ошибку):

```
UPDATE University SET name_university = '' WHERE id_university = 1;
```

Выполнение было неудачным, и в таблице ничего не изменилось

	id_university	name_university	id_city
▶	1	Харьковский Государственный Университет	1
	2	Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения	2
	3	Новосибирский Государственный Университет	2
*	NULL	NULL	NULL

5. Триггер: Проверка стоимости перед удалением (before\_university\_delete)  
Удаление университета с корректным id\_university (должен пройти успешно):

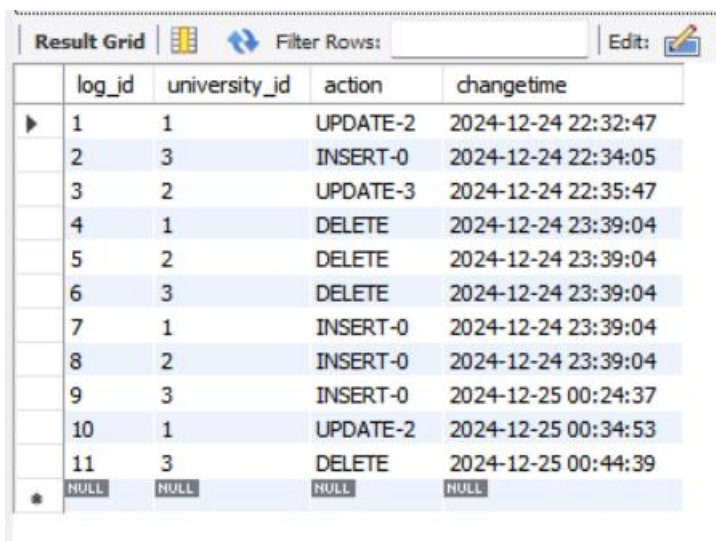
```
DELETE FROM University WHERE id_university = 3;
```

Результаты, показывающие успешное удаление

	id_university	name_university	id_city
▶	1	Харьковский Государственный Университет	1
	2	Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения	2
*	NULL	NULL	NULL

```
SELECT * FROM univ_log;
```

Проверка таблицы журналов на наличие операций в таблице университетов



	log_id	university_id	action	changetime
▶	1	1	UPDATE-2	2024-12-24 22:32:47
	2	3	INSERT-0	2024-12-24 22:34:05
	3	2	UPDATE-3	2024-12-24 22:35:47
	4	1	DELETE	2024-12-24 23:39:04
	5	2	DELETE	2024-12-24 23:39:04
	6	3	DELETE	2024-12-24 23:39:04
	7	1	INSERT-0	2024-12-24 23:39:04
	8	2	INSERT-0	2024-12-24 23:39:04
	9	3	INSERT-0	2024-12-25 00:24:37
	10	1	UPDATE-2	2024-12-25 00:34:53
	11	3	DELETE	2024-12-25 00:44:39
*	NULL	NULL	NULL	NULL

### **Выводы об использовании триггеров в разработанной базе данных:**

Использование триггеров в базе данных существенно улучшает управление данными и их целостность. Триггеры, такие как `before_university_insert` и `before_university_update`, предотвращают ввод университета с некорректными значениями, такими как пустое имя или неправильный `city_id`. Триггер `after_university_insert` фиксирует запись о новом университете в логах, что помогает отслеживать историю изменений. Аналогичным образом, триггеры для специализаций, такие как `before_specialization_insert` и `before_specialization_update`, обеспечивают корректность вводимых данных и предотвращают ошибки.

Триггеры, такие как `after_specialization_insert`, фиксируют состояние базы данных и помогают в аудите. Триггеры, удаляющие связанные записи при удалении университетов или специализаций, поддерживают целостность

базы данных, предотвращая "зависшие" записи. В целом, использование триггеров снижает вероятность ошибок, упрощает управление данными и повышает эффективность разработки.