ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | Н.В. Путилова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| Триггеры. Обеспечение активной целостности данных базы данных |
| по курсу: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4236 |  |  |  | Л. Мвале |
|  |  |  | подпись, дата |  | 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 вычислимых (производных) атрибутов - логирование (запись) изменений; - проверка корректности проводимых действий.). Вычислимые поля можно добавить при необходимости.

**Текст задания Вариант 15**

1. 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

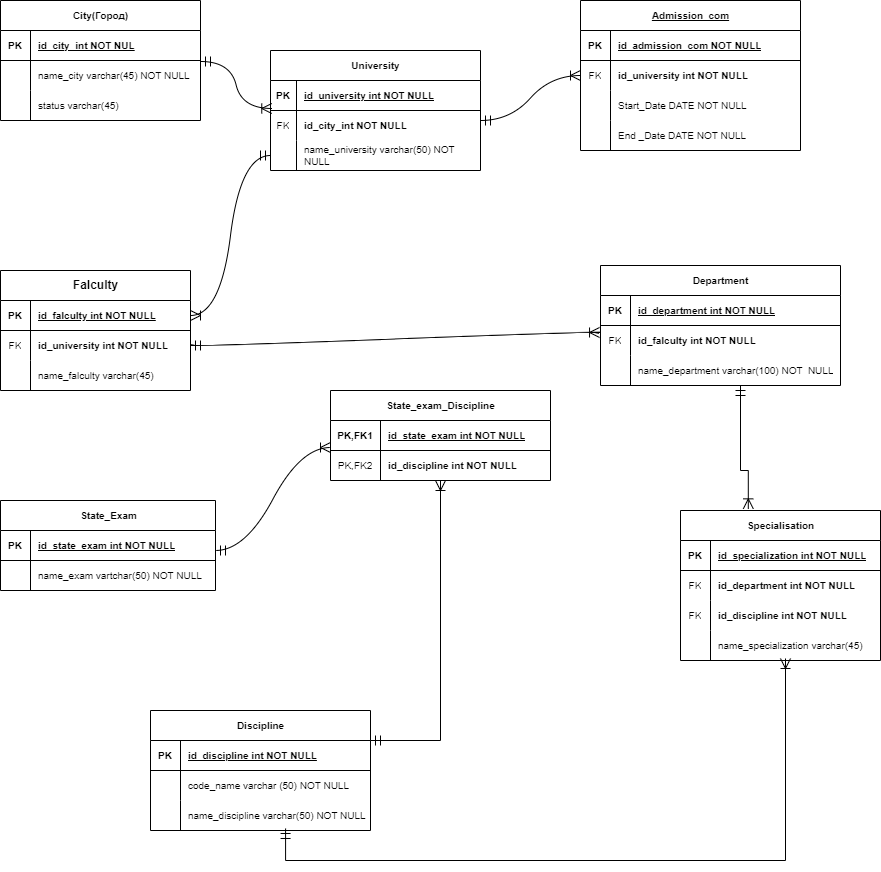
г. факультет, принимающий на количество направлений больше среднего

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 (Falculty, 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 Falculty 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 Falculty records

    DELETE FROM Falculty 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\_falculty IN (

        SELECT id\_falculty FROM Falculty 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\_falculty IN (

            SELECT id\_falculty FROM Falculty 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)

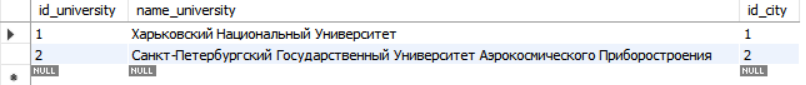
    VALUES (OLD.id\_university, 'DELETE', CURRENT\_TIMESTAMP);

END$$

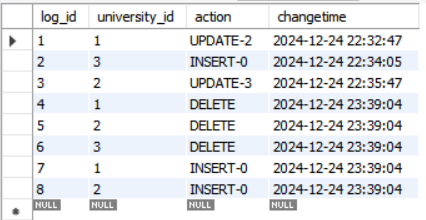
DELIMITER ;

**SQL операторы и скриншоты наборов данных, иллюстрирующие работу триггеров**

Начальное состояние университетской таблицы



SELECT \* FROM univ\_log;



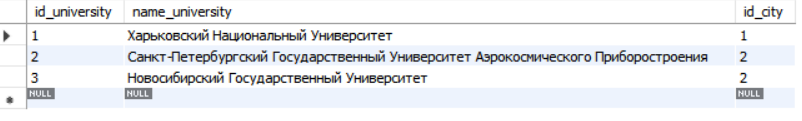
1. Триггер: Проверка стоимости перед вставкой (before\_university\_insert)

Вставка с корректной переменой (должен пройти успешно):

INSERT INTO University (name\_university, id\_city)

VALUES ('Новосибирский Государственный Университет', 2);

Результаты



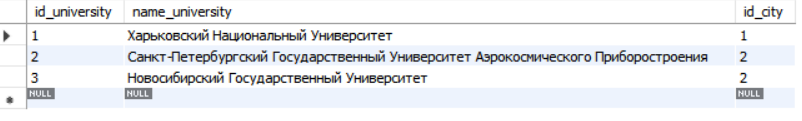
1. Триггер: Проверка стоимости перед вставкой (before\_university\_insert)

Вставка с пустым полем name\_university (должен дать ошибку):

INSERT INTO University (name\_university, id\_city)

VALUES ('', 2);

Никаких изменений в таблице



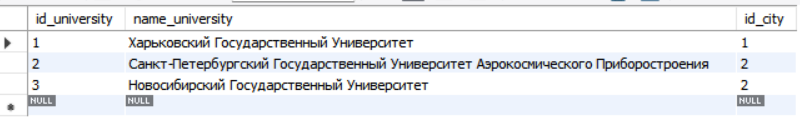
1. Триггер: Проверка стоимости перед обновлением (before\_university\_update)

Обновление с корректным значением name\_university(должен пройти успешно):

UPDATE University

SET name\_university = 'Харьковский Государственный Университет'

WHERE id\_university = 1;

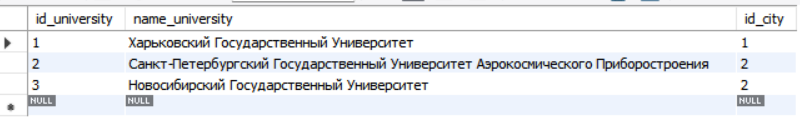
Результаты, показывающие обновленное название в 1-м id\_university

1. 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

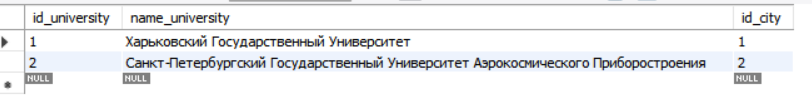


1. Триггер: Проверка стоимости перед удалением (before\_university\_delete)

Удаление университета с корректным id\_university (должен пройти успешно):

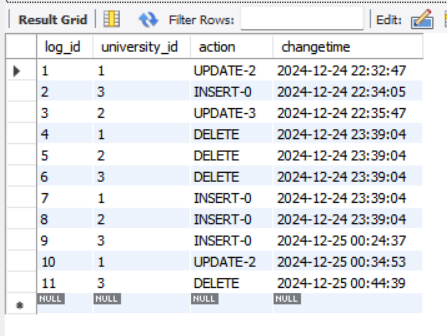
DELETE FROM University WHERE id\_university = 3;

Результаты, показывающие успешное удаление



SELECT \* FROM univ\_log;

Проверка таблицы журналов на наличие операций в таблице университетов



**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.