ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Б.К. Акопян
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

ТРИГГЕРЫ

по курсу: БАЗЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4217		Д.М. Никитин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1. **Цель работы:** получение практических навыков по программной реализации триггеров на сервере MySQL.

2. **Номер варианта, описание предметной области в тестовом формате:** 24 – база данных железнодорожной станции («Новосибирск-главный»).

1. Реализуйте триггер, привязанный к событию INSERT (для вашей предметной области). Проверьте его работоспособность. Результат покажите на скриншоте. Разработайте триггер, привязанный к событию UPDATE (для вашей предметной области). Проверьте его работоспособность. Результат покажите на скриншоте.

3. Создайте триггер, привязанный к событию DELETE (для вашей предметной области). Проверьте его работоспособность. Результат покажите на скриншоте.

4. Посмотри список реализованных триггеров и сделайте скриншот.

5. Выполните отчет в соответствие с требованиями ГОСТ 7.32-2017 и ГОСТ 2.105-2019: по оформлению отчетов (https://guap.ru/standart/doc).

3. **Схема данных из MySQL Workbench:**

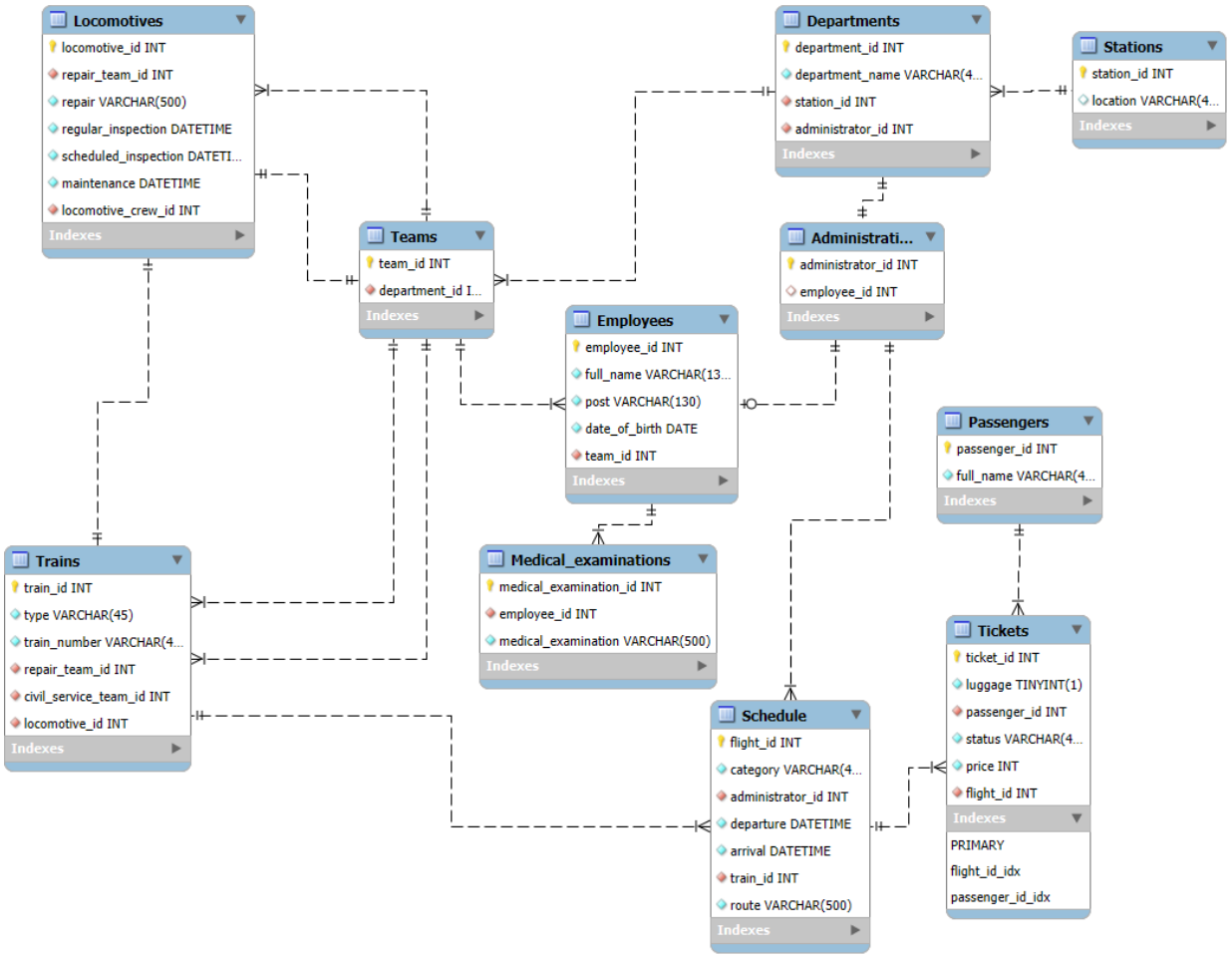

Рисунок 1 – Схема данных, на которой выполняется вариант

4. **Список триггеров на сервере:**

	Trigger	Event	Table	Statement	Timing	Created	sql_mode
▶	Medical_after_employee_insert	INSERT	employees	BEGIN -- Автоматически добавляем запись о ...	AFTER	2024-12-02 18:47:56.15	ONLY_FULL_GROUP
	after_employee_update	UPDATE	employees	BEGIN -- Проверим, изменилась ли должнос...	AFTER	2024-12-02 19:33:36.64	ONLY_FULL_GROUP
	check_employee_before_delete	DELETE	employees	BEGIN DECLARE employee_exists INT; -- ...	BEFORE	2024-12-02 20:34:33.60	ONLY_FULL_GROUP

Рисунок 2 – Список триггеров на сервере

5. **Текстовое описание назначения триггеров, тип связанного с ними события, листинг созданных триггеров с комментариями, скриншоты выполнения:**

medical_after_employee_insert:

Данный триггер предназначен для того, чтобы при добавлении нового работника, его запись появилась и в медицинском журнале. Автоматически устанавливается значение «Предстоит пройти» для состояния мед осмотра. Работу данного триггера можно увидеть на рисунках 3 и 4.

employee_id	full_name	post	date_of_birth	team_id
1	Иван Иванов	Водитель	1980-01-01	1
2	Анна Петрова	Диспетчер	1985-02-15	2
3	Алексей Смирнов	Ремонтник	1990-03-20	3
4	Дмитрий Кузнецов	Путеец	1995-04-25	4
5	Екатерина Соколова	Кассир	2000-05-30	5
6	Мария Федорова	Служба подготовки составов	2001-06-05	6
7	Ольга Морозова	Справочная служба	2000-07-10	7
21	Михаил Ярошенко	Водитель локомотива	1985-03-12	1
NULL	NULL	NULL	NULL	NULL

Рисунок 3 – Был добавлен работник с employee_id = 21

medical_examination_id	employee_id	medical_examination
1	1	Пройден
2	2	Пройден
3	3	Пройден
4	4	Пройден
5	5	Пройден
6	6	Пройден
7	7	Предстоит пройти
21	21	Предстоит пройти
NULL	NULL	NULL

Рисунок 4 – В таблице с медосмотрами появилась строка с employee_id = 21

```
DELIMITER //

CREATE TRIGGER `train_station`.`Medical_after_employee_insert`
AFTER INSERT ON `train_station`.`Employees`
FOR EACH ROW
BEGIN
          -- Автоматически добавляем запись о необходимости
медосмотра
     INSERT INTO `train_station`.`Medical_examinations` (
          `employee_id`,
          `medical_examination`
     ) VALUES (
          NEW.employee_id,
          'Предстоит пройти'
     );
END//

DELIMITER ;
```

after_employee_update:

Данный триггер срабатывает при обновлении значения должности сотрудника. Предполагается, что изменение должности сотрудника требует от него прохождения медосмотра, поэтому значение старого медосмотра

меняется на «Предстоит пройти», даже если он уже был пройден. Работу данного триггера можно увидеть на рисунках 5, 6, 7 и 8.

employee_id	full_name	post	date_of_birth	team_id
1	Иван Иванов	Водитель	1980-01-01	1
2	Анна Петрова	Диспетчер	1985-02-15	2
3	Алексей Смирнов	Ремонтник	1990-03-20	3
4	Дмитрий Кузнецов	Путеец	1995-04-25	4
5	Екатерина Соколова	Кассир	2000-05-30	5
6	Мария Федорова	Служба подготовки составов	2001-06-05	6
7	Ольга Морозова	Справочная служба	2000-07-10	7
24	Михаил Ярошенко	Водитель локомотива	1985-03-12	1
NULL	NULL	NULL	NULL	NULL

Рисунок 5 – Добавлен тестовый работник в должности водителя локомотива

medical_examination_id	employee_id	medical_examination
1	1	Пройден
2	2	Пройден
3	3	Пройден
4	4	Пройден
5	5	Пройден
6	6	Пройден
7	7	Предстоит пройти
24	24	Пройден
NULL	NULL	NULL

Рисунок 6 – Медосмотр тестового работника пройден

employee_id	full_name	post	date_of_birth	team_id
1	Иван Иванов	Водитель	1980-01-01	1
2	Анна Петрова	Диспетчер	1985-02-15	2
3	Алексей Смирнов	Ремонтник	1990-03-20	3
4	Дмитрий Кузнецов	Путеец	1995-04-25	4
5	Екатерина Соколова	Кассир	2000-05-30	5
6	Мария Федорова	Служба подготовки составов	2001-06-05	6
7	Ольга Морозова	Справочная служба	2000-07-10	7
24	Михаил Ярошенко	Помощник машиниста	1985-03-12	1
NULL	NULL	NULL	NULL	NULL

Рисунок 7 – Тестовый работник понижен до должности помощника машиниста

	medical_examination_id	employee_id	medical_examination
▶	1	1	Пройден
	2	2	Пройден
	3	3	Пройден
	4	4	Пройден
	5	5	Пройден
	6	6	Пройден
	7	7	Предстоит пройти
	24	24	Предстоит пройти
*	NULL	NULL	NULL

Рисунок 8 – Состояние медосмотра изменено на «Предстоит пройти»

```
DELIMITER //

CREATE TRIGGER `train_station`.`after_employee_update`
AFTER UPDATE ON `train_station`.`employees`
FOR EACH ROW
BEGIN
  -- Проверим, изменилась ли должность
  IF OLD.post != NEW.post THEN
    -- Если должность изменилась, обновим медосмотр сотрудника
    UPDATE `train_station`.`medical_examinations`
    SET `medical_examination` = 'Предстоит пройти'
    WHERE `employee_id` = NEW.employee_id;
  END IF;
END //

DELIMITER ;
```

check_employee_before_delete:

Данный триггер активируется при удалении сотрудника. Он создаёт временную таблицу с данными удалённого сотрудника, которую можно вывести при помощи использования процедуры `show_temp_log`. При этом, если сотрудника не существует, то данный триггер записывает сообщение об этом в таблицу. Работу данного триггера и процедуры можно увидеть на рисунках 9, 10, 11.

employee_id	full_name	post	date_of_birth	team_id
1	Иван Иванов	Водитель	1980-01-01	1
2	Анна Петрова	Диспетчер	1985-02-15	2
3	Алексей Смирнов	Ремонтник	1990-03-20	3
4	Дмитрий Кузнецов	Путеец	1995-04-25	4
5	Екатерина Соколова	Кассир	2000-05-30	5
6	Мария Федорова	Служба подготовки составов	2001-06-05	6
7	Ольга Морозова	Справочная служба	2000-07-10	7
28	Михаил Ярошенко	Водитель локомотива	1985-03-12	1
NULL	NULL	NULL	NULL	NULL

Рисунок 9 – Добавление тестового работника с employee_id = 28

message
Работник Михаил Ярошенко с ID 28 был снят…

Работник Михаил Ярошенко с ID 28 был снят с должности Водитель локомотива

Рисунок 10 – Сообщение, выводимое процедурой

employee_id	full_name	post	date_of_birth	team_id
1	Иван Иванов	Водитель	1980-01-01	1
2	Анна Петрова	Диспетчер	1985-02-15	2
3	Алексей Смирнов	Ремонтник	1990-03-20	3
4	Дмитрий Кузнецов	Путеец	1995-04-25	4
5	Екатерина Соколова	Кассир	2000-05-30	5
6	Мария Федорова	Служба подготовки составов	2001-06-05	6
7	Ольга Морозова	Справочная служба	2000-07-10	7
NULL	NULL	NULL	NULL	NULL

Рисунок 11 – Список с удалённым тестовым работником

При этом, если удалять пользователей, при этом не выводя их на экран, они будут записываться в временную таблицу, и удалятся только после завершения сессии, либо при запуске нужной процедуры. Заполненную таблицу можно увидеть на рисунке 12.

message
Работник Михаил Ярошенко с ID 30 был снят…
Работник Михаил Ярошенко с ID 31 был снят…
Работник Михаил Ярошенко с ID 32 был снят…
Работник Михаил Ярошенко с ID 33 был снят…
Работник Михаил Ярошенко с ID 34 был снят…
Работник Михаил Ярошенко с ID 35 был снят…
Работник Михаил Ярошенко с ID 36 был снят…
Работник Михаил Ярошенко с ID 37 был снят…

Рисунок 12 – Таблица после активной деятельности

```
DELIMITER $$

CREATE PROCEDURE show_temp_log()
BEGIN
    -- Запросим и выведем данные из временной таблицы temp_log
    SELECT * FROM temp_log;
    -- Удаляем временную таблицу
    DROP TEMPORARY TABLE IF EXISTS temp_log;
END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER check_employee_before_delete
BEFORE DELETE ON `Employees`
FOR EACH ROW
BEGIN
    DECLARE employee_exists INT;

    -- Создаём временную таблицу для логирования
    CREATE TEMPORARY TABLE IF NOT EXISTS temp_log (
        message TEXT
    );

    -- Проверяем, существует ли работник
    SELECT COUNT(*) INTO employee_exists
    FROM `Employees`
    WHERE `employee_id` = OLD.employee_id;

    -- Если работник найден, записываем его данные в временную
таблицу
    IF employee_exists > 0 THEN
        INSERT INTO temp_log (message)
        VALUES (CONCAT('Работник ', OLD.full_name, ' с ID ',
OLD.employee_id, ' был снят с должности ', OLD.post));
    ELSE
        -- Если работник не найден, записываем сообщение об ошибке
в временную таблицу
        INSERT INTO temp_log (message)
        VALUES ('Работник для удаления не найден.');
    END IF;

END$$

DELIMITER ;
```

6. **Расширенный вывод в формате эссе:**

В ходе работы был реализован ряд триггеров, которые автоматически управляют данными в базе данных железнодорожной станции "Новосибирск-главный". Были созданы триггеры, привязанные к событиям `INSERT`, `UPDATE` и

DELETE, каждый из которых выполняет важную роль в процессе автоматизации работы с информацией о сотрудниках.

Триггер INSERT автоматически добавляет запись в таблицу медицинских осмотров при добавлении нового работника в систему, устанавливая статус "Предстоит пройти" для медосмотра. Триггер UPDATE срабатывает при изменении должности сотрудника и обновляет состояние медосмотра на "Предстоит пройти", если должность была изменена. Триггер DELETE используется для логирования информации об удаленных сотрудниках, создавая временную таблицу с их данными, которую можно вывести с помощью хранимой процедуры. Все эти триггеры позволили автоматизировать рутинные операции, улучшив эффективность работы с данными в базе.