Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 3

по теме:

«Процедуры, функции, триггеры в PostgreSQL»

по дисциплине: Проектирование и реализация баз данных

Специальность:
45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверила:
Говорова М.М.
Дата: «..» … 2023 г.
Оценка _____

Выполнила:
студентка группы К32421
Милька К.А

Санкт-Петербург 2022/2023

Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание:

Вариант 1

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).

2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

Вариант 2

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2.
   2.1. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу.
   2.2. Создать авторский триггер по варианту индивидуального задания.

Предметная область – Вариант 16. БД "Спортивный клуб"

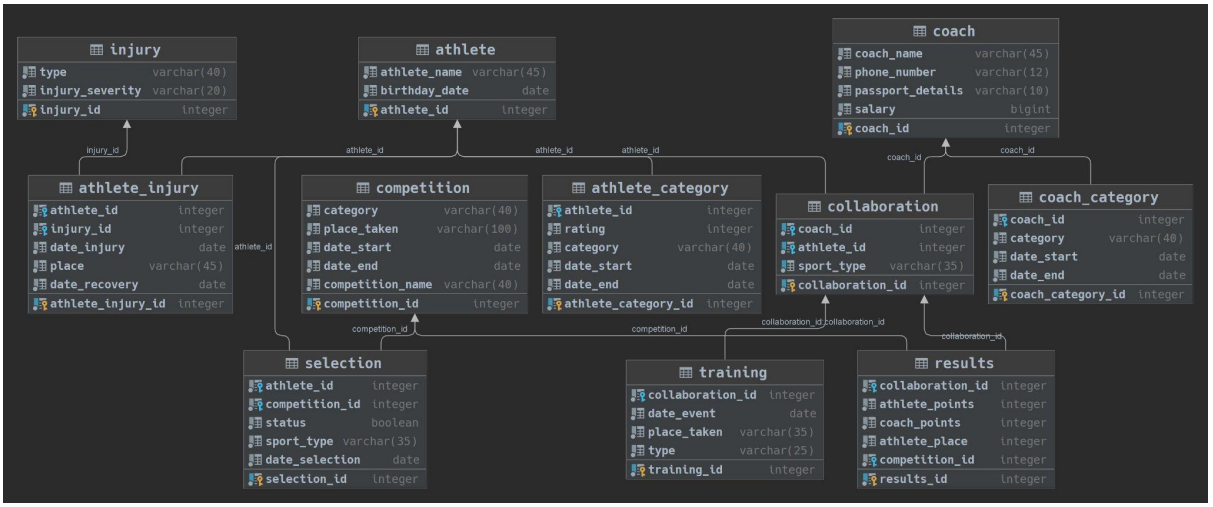

Рисунок 1 – ERD базы данных

Выполнение работы:

1. **Для вывода данных о результатах заданного спортсмена за прошедший год.**

Создание процедуры:

```
CREATE OR REPLACE FUNCTION get_athlete_results_past_year(_name TEXT)
RETURNS TABLE(
    athlete_id INT,
    athlete_name VARCHAR,
    competition_id INT,
    competition_name VARCHAR,
    results_id INT,
    athlete_points INT,
    athlete_place INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        a.athlete_id,
        a.athlete_name,
        com.competition_id,
        com.competition_name,
        r.results_id,
        r.athlete_points,
        r.athlete_place
    FROM athlete a
        JOIN collaboration col USING(athlete_id)
        JOIN results r USING(collaboration_id)
        JOIN competition com USING(competition_id)
    WHERE a.athlete_name = _name
        AND com.date_end >= NOW() - INTERVAL '1 year';
END;
$$;
```

Алгоритм: Функция принимает в качестве аргумента имя спортсмена и возвращает таблицу сформированную из данных возвращаемых запросом SELECT в теле функции. Отбираются только те данные которые принадлежат соответствующему спортсмену и где дата окончания соревнования в пределах прошедшего года.

Проверка функции:

```
SELECT * FROM get_athlete_results_past_year('Артемова Мария Алиевна')
```

	athlete_id integer	athlete_name character varying	competition_id integer	competition_name character varying	results_id integer	athlete_points integer	athlete_place integer
1	500	Артемова Мария Алиевна	6	Стремительная Гонка	813	3	276
2	500	Артемова Мария Алиевна	12	Бросок Веры	1792	2	464
3	500	Артемова Мария Алиевна	60	Гольфский Рай	8895	4	249

2. **Для вывода данных о соревнованиях, проводимых в первом квартале текущего года.**

Создание процедуры:

```
CREATE OR REPLACE FUNCTION get_competitions_first_quarter()
RETURNS SETOF competition
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT *
    FROM competition
    WHERE date_start >= DATE_TRUNC('year', CURRENT_DATE)::DATE
        AND date_start < DATE_TRUNC('year', CURRENT_DATE)::DATE + INTERVAL '3 month';
END;
$$;
```

Алгоритм: Функция возвращает все соревнования в виде “среза” таблицы “*competition*”, где дата начала находится в диапазоне квартала нынешнего года.

Проверка функции:

```
SELECT * FROM get_competitions_first_quarter();
```

	competition_id integer	category character varying (40)	place_taken character varying (100)	date_start date	date_end date	competition_name character varying (40)
1	4	юниорские	Екатеринбург, Россия	2023-01-17	2023-01-24	Огненная Олимпиада
2	29	областные	Оренбург, Россия	2023-02-17	2023-02-24	Парусная Гонка
3	56	любительские	Кишинев, Молдова	2023-03-07	2023-05-09	Боулинг Шторм
4	60	межшкольные	Будапешт, Венгрия	2023-01-17	2023-02-28	Гольфский Рай
5	92	университетские	Каир, Египет	2023-01-12	2023-02-23	Автокросс-Ралли

3. **Для повышения рейтинга и оклада тренера после участия в соревновании.**

```
CREATE OR REPLACE PROCEDURE increase_coach_salary_and_rating(
    _coach_id INT,
    new_category VARCHAR(40)
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO coach_category (
        coach_id,
        category,
        date_start,
        date_end
    )
    VALUES (
        _coach_id,
        new_category,
        CURRENT_DATE,
        CURRENT_DATE + INTERVAL '6 months'
    );

    UPDATE coach
    SET salary = salary * 1.15
    WHERE coach_id = _coach_id;
END;
$$;
```

Алгоритм: Процедура принимает ID тренера и категорию на которую необходимо изменить существующую в качестве аргументов. Процедура создает новую категорию для соответствующего тренера и назначает срок действия в пол года от текущего момента. Кроме того также повышается зарплата тренера на 15 процентов.

Проверка процедуры:

```
DO $$
DECLARE
    _coach_id INT;
BEGIN
    SELECT coach_id INTO _coach_id FROM coach WHERE coach_name = 'Попов Дамир Лукич';
    CALL increase_coach_salary_and_rating(_coach_id, 'регионально-признаный');
END;
$$;

SELECT coach_name, salary, category
FROM coach
    JOIN coach_category USING(coach_id)
WHERE coach_name = 'Попов Дамир Лукич'
```

	coach_name character varying (45)	salary bigint	category character varying (40)
1	Попов Дамир Лукич	250125	международно-признаный
2	Попов Дамир Лукич	250125	местно-признаный
3	Попов Дамир Лукич	250125	местно-признаный
4	Попов Дамир Лукич	250125	опытный
5	Попов Дамир Лукич	250125	регионально-признаный

4. **Создать триггер для логирования событий вставки, удаления, редактирования данных.**

   Суть триггера: Спортзалу необходимо четко контролировать все тренировки спортсменов, поэтому необходимо создать таблицу логов всех изменений при работе с таблицей training.

   Создание таблицы логов:

```
CREATE TABLE training_audit_log (
    log_id SERIAL PRIMARY KEY,
    training_id INT NOT NULL,
    collaboration_id INT NOT NULL,
    date_event DATE NOT NULL,
    place_taken VARCHAR(35) NOT NULL,
    type VARCHAR(25) NOT NULL,
    operation CHAR NOT NULL,
    change_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    changed_by VARCHAR(50) NOT NULL
)
```

   Укажем в ней все атрибуты таблицы training и дополнительные для остлеживания различных параметров таких как:

   1. operation – тип операции
   2. change_timestamp – временная метка изменения
   3. changed_by – автор изменения

Создание триггерной функции:

```
create function log_training_changes()
returns trigger
language plpgsql
as $$
BEGIN
    CASE TG_OP
        WHEN 'INSERT' THEN
            INSERT INTO training_audit_log(
                training_id,
                collaboration_id,
                date_event,
                place_taken,
                type,
                operation,
                changed_by
            )
            VALUES (
                NEW.training_id,
                NEW.collaboration_id,
                NEW.date_event,
                NEW.place_taken,
                NEW.type,
                'I',
                CURRENT_USER
            );
            RETURN NEW;
        WHEN 'UPDATE' THEN
            INSERT INTO training_audit_log(
                training_id,
                collaboration_id,
                date_event,
                place_taken,
                type,
                operation,
                changed_by
            )
            VALUES (
                NEW.training_id,
                NEW.collaboration_id,
                NEW.date_event,
                NEW.place_taken,
                NEW.type,
                'U',
                CURRENT_USER
            );
            RETURN NEW;
        WHEN 'DELETE' THEN
            INSERT INTO training_audit_log(
                training_id,
                collaboration_id,
                date_event,
                place_taken,
                type,
                operation,
```

```
            changed_by
        )
        VALUES (
            OLD.training_id,
            OLD.collaboration_id,
            OLD.date_event,
            OLD.place_taken,
            OLD.type,
            'D',
            CURRENT_USER
        );
        RETURN OLD;
    END CASE;
END;
$$;
```

Создание самого триггера и привязка к нему уже существующей функции:

```
CREATE TRIGGER log_training_after_modify_operations
AFTER INSERT OR UPDATE OR DELETE ON training
FOR EACH ROW
EXECUTE FUNCTION log_training_changes();
```

Триггер будет запускать функцию *“log_training_changes()”* для каждой строки которую одна из операция (INSERT, UPDATE, DELETE) будет изменять.

Проверим работу данного триггера:

```
INSERT INTO training(collaboration_id, date_event, place_taken, type)
VALUES (1, CURRENT_DATE, 'беговой зал', 'Игровая');

UPDATE training
SET type = 'Восстановительная'
WHERE training_id = (SELECT MAX(training_id) from training);

DELETE FROM training
WHERE training_id = (SELECT MAX(training_id) from training);

SELECT * FROM training_audit_log
```

	log_id [PK] integer	training_id integer	collaboration_id integer	date_event date	place_taken character varying (35)	type character varying (25)	operation character (1)	change_timestamp timestamp without time zone	changed_by character varying (50)
1	1	7032	1	2023-05-24	беговой зал	Игровая	I	2023-05-24 23:38:15.988452	postgres
2	2	7032	1	2023-05-24	беговой зал	Восстановительная	U	2023-05-24 23:38:16.002978	postgres
3	3	7032	1	2023-05-24	беговой зал	Восстановительная	D	2023-05-24 23:38:16.014512	postgres

Как можно заметить, теперь каждая операция модификации данных логируется в отдельную таблицу *“training_audit_log”*

Заключение:

В ходе выполнения данной работы, были изучены методы использования процедур и функций в рамках СУБД PostgreSQL. Были реализованы функции и процедуры для штатных ситуация для предметной области спортивного класса.

Помимо того, были рассмотрены и созданы триггеры и триггерные функции для логирования событий над таблицей.

Данная лабораторная работа дает практические навыки использования и применения функций, процедур и триггеров над базой данных.