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 № 5

по дисциплине: Проектирование и реализация баз данных

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверила:
Говорова М.М.
Дата: _____2023 г.
Оценка

Выполнил:
студент группы К3241
Борисова Э. Е.

Санкт-Петербург 2023/2024

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

• Создать процедуры/функции согласно индивидуальному заданию (часть4).

• 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.

Предметная область: Вариант 16. БД "Спортивный клуб"

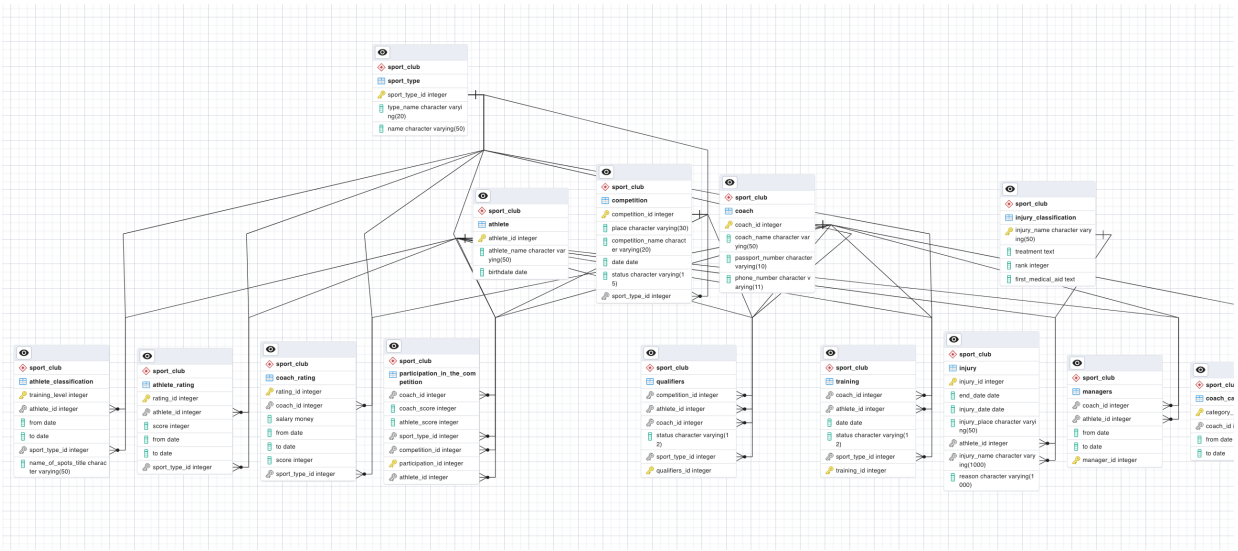

Рисунок 1 – ERD базы данных

Выполнение работы:

Процедуры и функции.

Для вывода данных о результатах заданного спортсмена за прошедший год.

```
CREATE OR REPLACE FUNCTION get_athlete_results(ath_id integer)
RETURNS TABLE (
competition_name character varying(50),
athlete_name character varying(20),
place character varying(30),
athlete_score integer,
competition_date timestamp without time zone
) AS $$
BEGIN
RETURN QUERY
SELECT
    competition.competition_name,
    athlete.athlete_name,
    competition.place,
    participation_in_the_competition.athlete_score,
    competition.date::timestamp
FROM
    sport_club.participation_in_the_competition
JOIN
    sport_club.competition ON participation_in_the_competition.competition_id =
competition.competition_id
JOIN
    sport_club.athlete ON participation_in_the_competition.athlete_id = athlete.athlete_id
WHERE
    athlete.athlete_id = ath_id
    AND competition.status = 'проведены'
    AND competition.date >= NOW() - INTERVAL '1 year';
END;
$$ LANGUAGE plpgsql;
```

```
psql (16.0)
Type "help" for help.

sport_club=# select * from get_athlete_results(1008);
 competition_name | athlete_name  | place | athlete_score |  competition_date
------------------+---------------+-------+---------------+---------------------
 Чемпионат        | Аркадий Кашин | 2     |           105 | 2023-08-10 00:00:00
(1 row)

sport_club=#
```

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

```
CREATE OR REPLACE FUNCTION get_competitions_first_quarter()
RETURNS TABLE (
competition_id integer,
competition_name character varying(50),
competition_date timestamp without time zone,
sport_type_name character varying(20),
status character varying(15)
)
AS $$
BEGIN
RETURN QUERY
SELECT
    competition.competition_id,
    competition.competition_name,
    CAST(competition.date AS timestamp) AS date,
 sport_type.type_name,
    competition.status
FROM
    sport_club.competition
JOIN
    sport_club.sport_type ON competition.sport_type_id = sport_type.sport_type_id
WHERE
    competition.date >= DATE_TRUNC('year', CURRENT_DATE)::DATE
```

```
    AND competition.date < DATE_TRUNC('year', CURRENT_DATE)::DATE + INTERVAL '3 month';

END;

$$ LANGUAGE plpgsql;
```

```
sport_club=# select * from get_competitions_first_quarter();
 competition_id | competition_name |  competition_date   | sport_type_name |  status
----------------+------------------+---------------------+-----------------+-----------
             14 | Соревнование 14  | 2023-02-10 00:00:00 | Единоборства    | проведены
             15 | Соревнование 15  | 2023-03-05 00:00:00 | Гимнастика      | проведены
             13 | Соревнование 13  | 2023-01-25 00:00:00 | Водный          | проведены
```

Для повышения рейтинга и оклада тренера после участия в соревновании.

```
CREATE OR REPLACE PROCEDURE upd_coach_salary_and_rating(
    _coach_id INT, new_category VARCHAR(40), max_salary_limit NUMERIC)
LANGUAGE plpgsql
AS $$
DECLARE
    new_salary NUMERIC;
BEGIN
    UPDATE sport_club.coach_category
    SET "to" = CURRENT_DATE, category = new_category, "from" = CURRENT_DATE + INTERVAL '6 months'
    WHERE coach_id = _coach_id AND category_number = category_number;


    SELECT CASE
        WHEN (salary * 1.15) <= max_salary_limit THEN salary * 1.15
        ELSE max_salary_limit
    END INTO new_salary
    FROM sport_club.coach_rating
    WHERE coach_id = _coach_id;


    UPDATE sport_club.coach_rating
    SET salary = new_salary
    WHERE coach_id = _coach_id;
END;
$$;
```

```
sport_club=# call upd_coach_salary_and_rating(4038, 'вторая категория', 6000.0);
CALL
sport_club=# select * from sport_club.coach_rating where coach_id = 4038;
 rating_id | coach_id |  salary   |    from    |     to     | score | sport_type_id
-----------+----------+-----------+------------+------------+-------+---------------
      3857 |     4038 | 3967.5000 | 2023-01-01 | 2023-06-10 |   154 |             6
(1 row)

sport_club=# select * from sport_club.coach_category where coach_id = 4038;
 category_number | coach_id |    from    |     to     |     category
-----------------+----------+------------+------------+-----------------
            8465 |     4038 | 2024-06-26 | 2023-12-26 | вторая категория
(1 row)
```

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.

Таблица логов:

```
CREATE TABLE log_competition (
    log_id SERIAL PRIMARY KEY,
    action_type VARCHAR(10) NOT NULL,
    table_nm VARCHAR(50) NOT NULL,
    comp_id INT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Функция:

```
CREATE OR REPLACE FUNCTION log_competition_ch()
RETURNS TRIGGER AS $$
DECLARE
    log_action_type VARCHAR(10);
    log_comp_id INT;
BEGIN
    IF TG_OP = 'INSERT' THEN
        log_action_type := 'INSERT';
        log_comp_id := NEW.competition_id;
    ELSIF TG_OP = 'UPDATE' THEN
```

```
        log_action_type := 'UPDATE';
        log_comp_id := NEW.competition_id;
    ELSIF TG_OP = 'DELETE' THEN
        log_action_type := 'DELETE';
        log_comp_id := OLD.competition_id;
    END IF;

    INSERT INTO log_competition(action_type, table_nm, comp_id)
    VALUES (log_action_type, 'competition', log_comp_id);

    IF NEW.status = 'проведено' THEN
        NEW.place = COALESCE(NEW.place, 'Место не указано');
    ELSIF NEW.status = 'отменено' THEN
        NEW.place = NULL;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER competition_logtr
BEFORE INSERT OR UPDATE OR DELETE ON sport_club.competition
FOR EACH ROW
EXECUTE FUNCTION log_competition_ch();
```

```
sport_club=# select * from log_competition;
 log_id | action_type |  table_nm   | comp_id |         timestamp
--------+-------------+-------------+---------+----------------------------
      1 | INSERT      | competition |    2777 | 2023-12-27 16:24:17.821032
      2 | UPDATE      | competition |       4 | 2023-12-27 16:25:35.693427
      3 | DELETE      | competition |     504 | 2023-12-27 16:27:40.673879
(3 rows)

sport_club=#
```

Триггер на проверку возраста спортсмена при занятии триатлоном.

```
CREATE OR REPLACE FUNCTION check_athlete_age()
RETURNS TRIGGER AS $$
BEGIN
   IF EXTRACT(YEAR FROM AGE((SELECT birthdate FROM sport_club.athlete WHERE athlete_id = NEW.athlete_id))) < 12
      AND NEW.sport_type_id = 56 THEN
       RAISE EXCEPTION 'Спортсменам младше 12 лет запрещено заниматься триатлоном.';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_athlete_classification
BEFORE INSERT ON sport_club.athlete_classification
FOR EACH ROW
EXECUTE FUNCTION check_athlete_age();
```

```
sport_club=# INSERT INTO sport_club.athlete_classification (training_level, athlete_id, "from", "to", sport_type_id, name_of_spots_title)
VALUES (1003, 3990, '2024-01-01', '2024-06-01', 56, 'Новичок');
ERROR:  Спортсменам младше 12 лет запрещено заниматься триатлоном.
CONTEXT:  PL/pgSQL function check_athlete_age() line 5 at RAISE
sport_club=#
```

Вывод: в ходе данной лабораторной работы были созданы триггеры и функции логирования. Эта работа позволила приобрести практические навыки программирования на SQL и лучше понять принципы функционирования триггеров.