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

Обеспечение активной целостности данных базы данных

по дисциплине: Проектирование баз данных

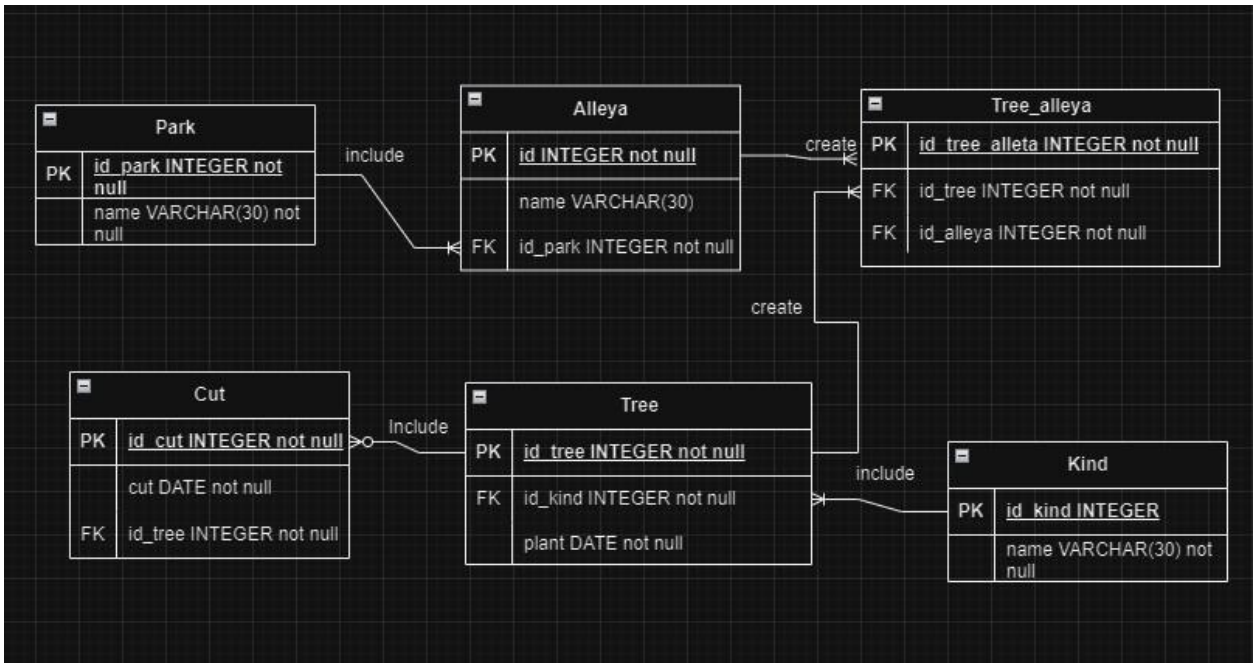
РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4134к		Костяков Н.А.
	подпись, дата	инициалы, фамилия

Санкт-Петербург
2023

Задание

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 (полей);
- логирование (запись) изменений;
- обеспечения безопасности данных;
- логическое (мягкое) удаление данных
- проверка корректности проводимых действий.).
Вычислимые поля можно добавить при необходимости.



Before insert

```
----лимит даты tree before insert


    CREATE OR REPLACE FUNCTION check_plant_date()
    RETURNS TRIGGER AS $$
    BEGIN
        IF NEW.plant > CURRENT_DATE THEN
            RAISE EXCEPTION 'Дата посадки не может быть в будущем';
        END IF;
        RETURN NEW;
    END;
    $$ LANGUAGE plpgsql;

    CREATE TRIGGER before_insert_check_plant_date
    BEFORE INSERT ON TREE
    FOR EACH ROW
```

```
    EXECUTE FUNCTION check_plant_date();
```

Query | Query History

```
1  insert into tree (id_tree, id_kind, plant) values (55, 1, '22-12-2024');
```

Data Output | Messages | Notifications

```
ERROR:  Дата посадки не может быть в будущем
CONTEXT:  функция PL/pgSQL check_plant_date(), строка 4, оператор RAISE

ОШИБКА:  Дата посадки не может быть в будущем
SQL state: P0001
```

After delete

```
--удаление парков если в нем нет аллей
CREATE OR REPLACE FUNCTION after_delete_alley_delete_park()
RETURNS TRIGGER AS $$
BEGIN

    -- Если в парке больше нет аллей, удаляем парк
    IF NOT EXISTS (SELECT 1 FROM ALLEYA a WHERE a.ID_PARK = OLD.ID_PARK) THEN
        DELETE FROM PARK WHERE ID_PARK = OLD.ID_PARK;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_delete_alley_delete_park
AFTER DELETE ON ALLEYA
FOR EACH ROW
EXECUTE FUNCTION after_delete_alley_delete_park();
```

Для того, чтобы показать, как парк удаляется приведу логи из change_log

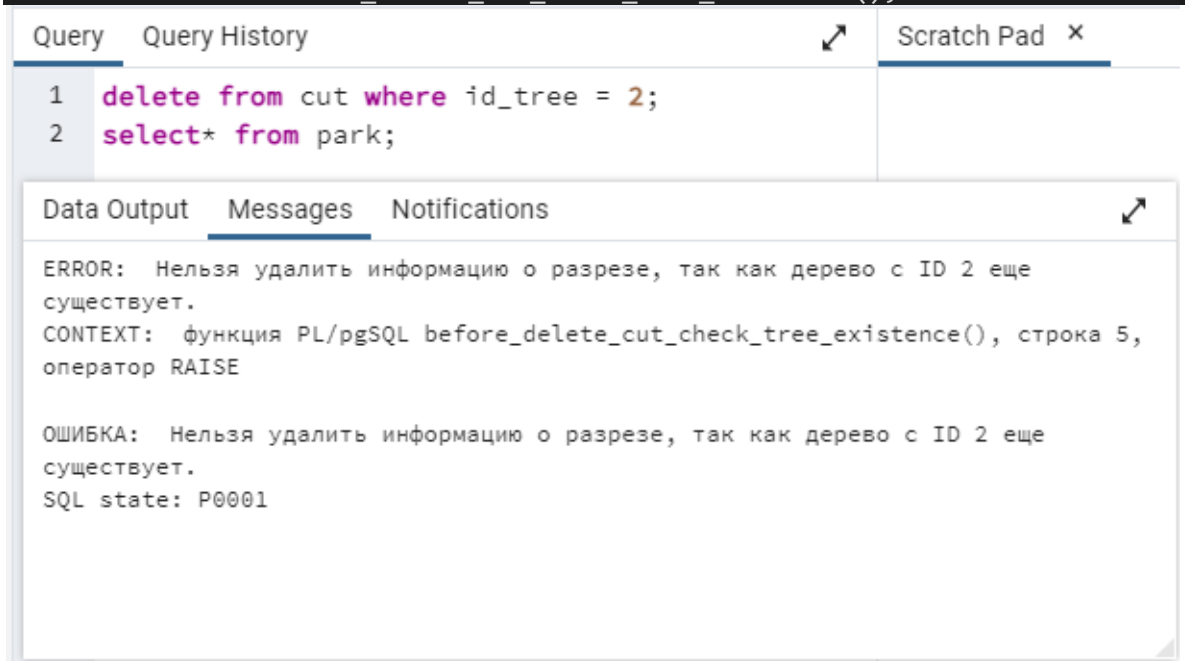
76	78	park	111	D	2023-12-22 19:13:28.114836+03
77	79	alleya	4	D	2023-12-22 19:13:28.114836+03
78	80	alleya	1	D	2023-12-22 19:13:28.114836+03
79	81	alleya	2	D	2023-12-22 19:13:28.114836+03
80	82	alleya	5	D	2023-12-22 19:13:28.114836+03

Before delete

```
--не даст удалить информацию из cut, если дерево еще существует
CREATE OR REPLACE FUNCTION before_delete_cut_check_tree_existence()
RETURNS TRIGGER AS $$
BEGIN
    -- Проверяем существование дерева
    IF EXISTS (SELECT 1 FROM TREE WHERE ID_TREE = OLD.ID_TREE) THEN
        RAISE EXCEPTION 'Нельзя удалить информацию об обрезке, так как дерево с
ID % еще существует.', OLD.ID_TREE;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_delete_cut_check_tree_existence
BEFORE DELETE ON CUT
FOR EACH ROW
EXECUTE FUNCTION before_delete_cut_check_tree_existence();
```



Before update

```
-- существует ли новый парк, и если нет, отменит операцию обновления.
CREATE OR REPLACE FUNCTION before_update_alleya_check_park_existence()
RETURNS TRIGGER AS $$
BEGIN
    -- Проверяем существование нового парка
    IF NOT EXISTS (SELECT 1 FROM PARK WHERE ID_PARK = NEW.ID_PARK) THEN
        RAISE EXCEPTION 'Нельзя обновить аллею. Парк с ID % не существует.',
NEW.ID_PARK;
```

```
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_update_alleya_check_park_existence
BEFORE UPDATE ON ALLEYA
FOR EACH ROW
EXECUTE FUNCTION before_update_alleya_check_park_existence();
```

Query | Query History

```
1  update alleya set id_park =10 where id_park = 2;
2
```

Data Output | Messages | Notifications

```
ERROR:  Нельзя обновить аллею. Парк с ID 10 не существует.
CONTEXT:  функция PL/pgSQL before_update_alleya_check_park_existence(), строка
5, оператор RAISE

ОШИБКА:  Нельзя обновить аллею. Парк с ID 10 не существует.
SQL state: P0001
```

After update

```
 --существует ли дерево с новым  id_tree
CREATE OR REPLACE FUNCTION after_update_cut_check_tree_existence()
RETURNS TRIGGER AS $$
BEGIN
    -- Проверяем наличие дерева с новым ID_TREE
    IF NOT EXISTS (SELECT 1 FROM TREE WHERE ID_TREE = NEW.ID_TREE) THEN
        RAISE EXCEPTION 'Дерева с ID_TREE = % не существует', NEW.ID_TREE;
    ELSE
        RAISE NOTICE 'Данные об обрезке изменены для дерева с ID_TREE = %',
NEW.ID_TREE;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER after_update_cut_check_tree_existence
AFTER UPDATE ON CUT
FOR EACH ROW
EXECUTE FUNCTION after_update_cut_check_tree_existence();
```

Query | Query History

```
1  update cut set id_tree =10 where id_tree = 2;
2
```

Data Output | Messages | Notifications

```
ЗАМЕЧАНИЕ:  Данные об обрезке изменены для дерева с ID_TREE = 10
ЗАМЕЧАНИЕ:  Данные об обрезке изменены для дерева с ID_TREE = 10
ЗАМЕЧАНИЕ:  Данные об обрезке изменены для дерева с ID_TREE = 10
UPDATE 3

Query returned successfully in 39 msec.
```

After insert и логирование

```sql
--логирование
CREATE TABLE CHANGE_LOG (
    LOG_ID SERIAL PRIMARY KEY,
    TABLE_NAME VARCHAR(50) NOT NULL,
    RECORD_ID INTEGER NOT NULL,
    OPERATION CHAR(1) NOT NULL, -- 'I' (insert), 'U' (update), 'D' (delete)
    LOGGED_AT TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION log_change_park()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_PARK, 'I');
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_PARK, 'U');
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, OLD.ID_PARK, 'D');
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_park_changes
AFTER INSERT OR UPDATE OR DELETE ON PARK
FOR EACH ROW
EXECUTE FUNCTION log_change_park();


CREATE OR REPLACE FUNCTION log_change_alleya()
```

```
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_ALLEYA, 'I');
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_ALLEYA, 'U');
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, OLD.ID_ALLEYA, 'D');
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_alleya_changes
AFTER INSERT OR UPDATE OR DELETE ON ALLEYA
FOR EACH ROW
EXECUTE FUNCTION log_change_alleya();



CREATE OR REPLACE FUNCTION log_change_tree()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_TREE, 'I');
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_TREE, 'U');
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, OLD.ID_TREE, 'D');
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_tree_changes
AFTER INSERT OR UPDATE OR DELETE ON TREE
FOR EACH ROW
EXECUTE FUNCTION log_change_tree();


CREATE OR REPLACE FUNCTION log_change_kind()
RETURNS TRIGGER AS $$
BEGIN
```

```
    IF TG_OP = 'INSERT' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_KIND, 'I');
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_KIND, 'U');
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, OLD.ID_KIND, 'D');
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_kind_changes
AFTER INSERT OR UPDATE OR DELETE ON KIND
FOR EACH ROW
EXECUTE FUNCTION log_change_kind();


CREATE OR REPLACE FUNCTION log_change_cut()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_CUT, 'I');
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, NEW.ID_CUT, 'U');
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO CHANGE_LOG (TABLE_NAME, RECORD_ID, OPERATION)
        VALUES (TG_TABLE_NAME, OLD.ID_CUT, 'D');
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_cut_changes
AFTER INSERT OR UPDATE OR DELETE ON CUT
FOR EACH ROW
EXECUTE FUNCTION log_change_cut();
```

Query | Query History | Scratch Pad

```
1  select * from change_log;
2
```

Data Output | Messages | Notifications

	log_id [PK] integer	table_name character varying (50)	record_id integer	operation character	logged_at timestamp with time zone
1	1	park	5	I	2023-12-13 16:05:00.061002+03
2	2	cut	11	I	2023-12-13 16:06:39.956576+03
3	3	alleya	11	I	2023-12-13 16:15:21.894538+03
4	4	alleya	12	I	2023-12-13 16:15:30.501725+03
5	5	alleya	13	I	2023-12-13 16:15:34.666693+03
6	6	alleya	14	I	2023-12-13 16:15:45.524305+03
7	7	alleya	15	I	2023-12-13 16:15:48.042541+03
8	8	alleya	8	D	2023-12-13 16:16:56.504081+03
9	9	alleya	9	D	2023-12-13 16:16:56.504081+03
10	10	park	1	D	2023-12-13 16:16:56.504081+03
11	11	alleya	11	D	2023-12-13 16:16:56.504081+03
12	12	alleya	12	D	2023-12-13 16:16:56.504081+03
13	13	alleya	13	D	2023-12-13 16:16:56.504081+03
14	14	alleya	14	D	2023-12-13 16:16:56.504081+03
15	15	alleya	15	D	2023-12-13 16:16:56.504081+03
16	17	park	3	I	2023-12-13 16:19:05.785815+03
17	18	alleya	21	I	2023-12-13 16:19:05.785815+03
18	19	alleya	22	I	2023-12-13 16:19:25.792073+03
19	20	alleya	23	I	2023-12-13 16:19:25.792073+03
20	21	alleya	24	I	2023-12-13 16:19:25.792073+03
21	22	alleya	25	I	2023-12-13 16:19:25.792073+03
22	23	park	3	D	2023-12-13 16:24:06.581128+03
23	24	alleya	21	D	2023-12-13 16:24:06.581128+03
24	25	alleya	22	D	2023-12-13 16:24:06.581128+03
25	26	alleya	23	D	2023-12-13 16:24:06.581128+03
26	27	alleya	24	D	2023-12-13 16:24:06.581128+03