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 №3  Объектно-реляционные базы данных. Проектирование и создание |
| **по дисциплине: Проектирование баз данных** |
|  |
|  |

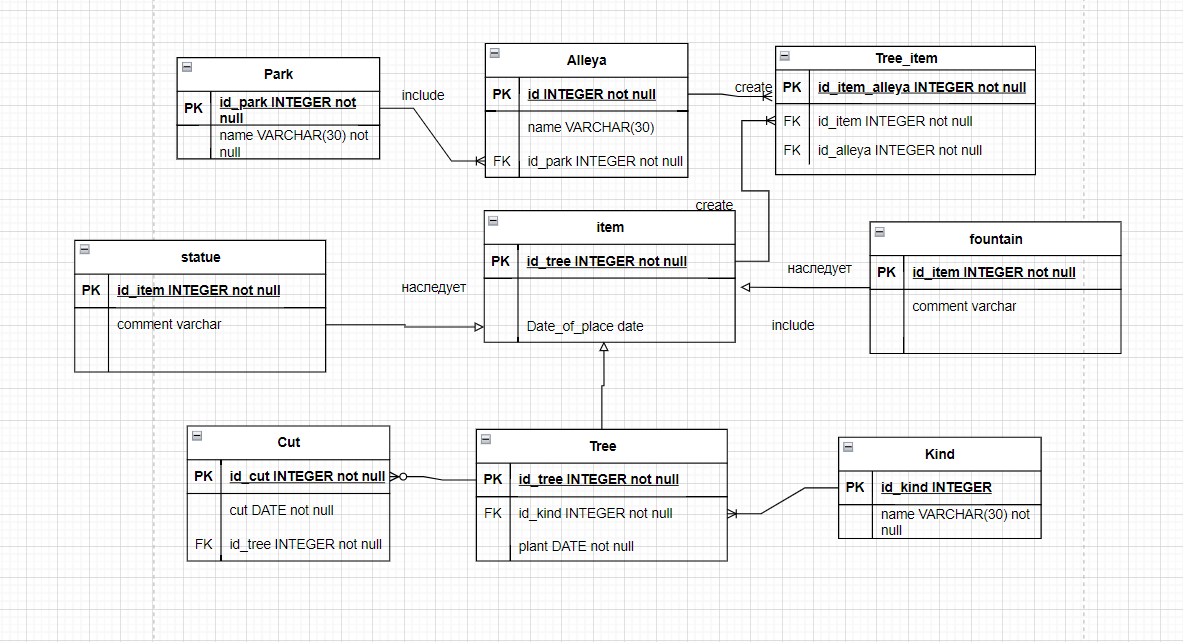
РАБОТУ ВЫПОЛНИЛ

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

Санкт-Петербург

2024

## Задание:

Спроектировать физическую модель базы данных, находящуюся в третьей нормальной форме, и включающую наследование и хотя бы один пользовательский тип (составной и/или перечислимый) в соответствии с заданным вариантом. Написать соответствующий скрипт создания базы данных.  
Обратите внимание: 1)Для корректной работы внешних ключей ссылающихся на предка необходимо ограничение внешнего ключа реализовывать не декларативно, а с помощью триггеров.  
2)Если у Вас не будет составного или диапазонного пользовательского типа будет сложно делать агрегатную функцию в следующей работе

## Листинг

CREATE TABLE PARK (

    ID\_PARK SERIAL PRIMARY KEY,

    NAME VARCHAR(30) NOT NULL

);

CREATE TABLE ALLEYA (

    ID\_ALLEYA SERIAL PRIMARY KEY,

    NAME VARCHAR(30) NOT NULL,

    ID\_PARK INTEGER,

    FOREIGN KEY (ID\_PARK) REFERENCES PARK(ID\_PARK) ON UPDATE CASCADE ON DELETE RESTRICT

);

CREATE TABLE KIND (

    ID\_KIND SERIAL PRIMARY KEY,

    NAME VARCHAR(30) NOT NULL

);

CREATE TABLE ITEM (

    ID\_ITEM SERIAL NOT NULL PRIMARY KEY,

    COMMENTARY varchar(255) default null

);

CREATE TYPE TREE\_STATUS AS ENUM ('GOOD', 'BAD', 'DEAD');

CREATE TABLE TREE (

    PRIMARY KEY (ID\_ITEM),

    WEALTH TREE\_STATUS,

    DATE\_OF\_PLANT DATE,

    ID\_KIND INTEGER NOT NULL,

    FOREIGN  KEY (ID\_KIND) REFERENCES KIND(ID\_KIND) ON DELETE CASCADE ON UPDATE CASCADE

    ) INHERITS (ITEM);

CREATE TABLE FOUNTAIN (

    PRIMARY KEY (ID\_ITEM),

    DATE\_OF\_PLACE DATE

) INHERITS (ITEM);

CREATE TABLE STATUE (

    PRIMARY KEY (ID\_ITEM),

    DATE\_OF\_PLACE DATE

) INHERITS (ITEM);

CREATE TABLE CUT (

    ID\_CUT SERIAL PRIMARY KEY,

    ID INTEGER NOT NULL,

    CUT DATE NOT NULL,

    CONSTRAINT FK\_CUT\_TREE FOREIGN KEY (ID) REFERENCES TREE(ID\_ITEM) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE ITEM\_ALLEYA (

    ID\_ITEM\_ALLEYA SERIAL PRIMARY KEY,

    ID INTEGER NOT NULL,

    ID\_ALLEYA INTEGER NOT NULL

);

-- Проверка на существование

CREATE OR REPLACE FUNCTION update\_item\_id\_on\_item\_alleya\_update() RETURNS TRIGGER AS $$

BEGIN

    IF NOT EXISTS (

        SELECT 1 FROM ITEM WHERE ID\_ITEM = NEW.ID

    ) THEN

        RAISE EXCEPTION 'Item with ID % does not exist', NEW.ID;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_item\_id\_on\_item\_alleya\_update

BEFORE UPDATE ON ITEM\_ALLEYA

FOR EACH ROW

EXECUTE FUNCTION update\_item\_id\_on\_item\_alleya\_update();

CREATE TRIGGER update\_item\_id\_on\_item\_alleya\_in

BEFORE INSERT ON ITEM\_ALLEYA

FOR EACH ROW

EXECUTE FUNCTION update\_item\_id\_on\_item\_alleya\_update();

-- Триггеры для обновления в таблице

CREATE OR REPLACE FUNCTION update\_item\_alleya\_on\_update() RETURNS TRIGGER AS $$

BEGIN

    UPDATE ITEM\_ALLEYA

    SET ID = NEW.ID\_ITEM

    WHERE ID = OLD.ID\_ITEM;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_item\_alleya\_on\_update\_fountain

AFTER UPDATE ON FOUNTAIN

FOR EACH ROW

EXECUTE FUNCTION update\_item\_alleya\_on\_update();

CREATE TRIGGER update\_item\_alleya\_on\_update\_tree

AFTER UPDATE ON TREE

FOR EACH ROW

EXECUTE FUNCTION update\_item\_alleya\_on\_update();

CREATE TRIGGER update\_item\_alleya\_on\_update\_statue

AFTER UPDATE ON STATUE

FOR EACH ROW

EXECUTE FUNCTION update\_item\_alleya\_on\_update();

-- Триггеры для удаления из таблицы

CREATE OR REPLACE FUNCTION delete\_item\_alleya\_on\_delete() RETURNS TRIGGER AS $$

BEGIN

    DELETE FROM ITEM\_ALLEYA

    WHERE ID = OLD.ID\_ITEM;

    RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER delete\_item\_alleya\_on\_fountain\_delete

BEFORE DELETE ON FOUNTAIN

FOR EACH ROW

EXECUTE FUNCTION delete\_item\_alleya\_on\_delete();

CREATE TRIGGER delete\_item\_alleya\_on\_tree\_delete

BEFORE DELETE ON TREE

FOR EACH ROW

EXECUTE FUNCTION delete\_item\_alleya\_on\_delete();

CREATE TRIGGER delete\_item\_statue\_on\_statue\_delete

BEFORE DELETE ON STATUE

FOR EACH ROW

EXECUTE FUNCTION delete\_item\_alleya\_on\_delete();

-------------------------------------------------------------------------

INSERT INTO PARK (ID\_PARK, NAME) VALUES (1, 'Космический');

INSERT INTO PARK (ID\_PARK, NAME) VALUES (2, 'Спортивный');

INSERT INTO ALLEYA (ID\_ALLEYA, NAME, ID\_PARK) VALUES (1, 'Скоростная', 1);

INSERT INTO ALLEYA (ID\_ALLEYA, NAME, ID\_PARK) VALUES (2, 'Ракетная', 1);

INSERT INTO ALLEYA (ID\_ALLEYA, NAME, ID\_PARK) VALUES (3, 'Беговая', 2);

INSERT INTO ALLEYA (ID\_ALLEYA, NAME, ID\_PARK) VALUES (4, 'Быстрая', 2);

INSERT INTO ALLEYA (ID\_ALLEYA, NAME, ID\_PARK) VALUES (5, 'Тройная липовая', 1);

INSERT INTO ALLEYA (ID\_ALLEYA, NAME, ID\_PARK) VALUES (6, 'Театральная', 2);

INSERT INTO KIND (ID\_KIND, NAME) VALUES (1, 'Клен');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (2, 'Ясень');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (3, 'Береза');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (4, 'Дуб');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (5, 'Липа');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (6, 'Ель');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (7, 'Яблоня');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (8, 'Ива');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (9, 'Тополь');

INSERT INTO KIND (ID\_KIND, NAME) VALUES (10, 'Пихта');

INSERT INTO TREE (DATE\_OF\_PLANT, ID\_KIND, WEALTH) VALUES ('16-04-2021', 1, 'GOOD');

INSERT INTO TREE (DATE\_OF\_PLANT, ID\_KIND, WEALTH) VALUES ('16-04-2021', 3, 'GOOD');

INSERT INTO FOUNTAIN (DATE\_OF\_PLACE) VALUES ('16-04-2024');

INSERT INTO STATUE (DATE\_OF\_PLACE) VALUES ('16-04-2024');

-- Inserting into ITEM\_ALLEYA

INSERT INTO ITEM\_ALLEYA (ID, ID\_ALLEYA) VALUES (1, 1);

INSERT INTO ITEM\_ALLEYA (ID, ID\_ALLEYA) VALUES (2, 2);

INSERT INTO ITEM\_ALLEYA (ID, ID\_ALLEYA) VALUES (3, 3);

INSERT INTO ITEM\_ALLEYA (ID, ID\_ALLEYA) VALUES (4, 4);

-- Inserting into CUT

INSERT INTO CUT (ID, CUT) VALUES (1, '2023-12-10');

INSERT INTO CUT (ID, CUT) VALUES (2, '2023-12-23');