КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

_____	_____	_____
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №9

Объектно-реляционные базы данных. Проектирование и создание

по дисциплине: Проектирование баз данных

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. _____ _____ _____

подпись, дата | инициалы, фамилия

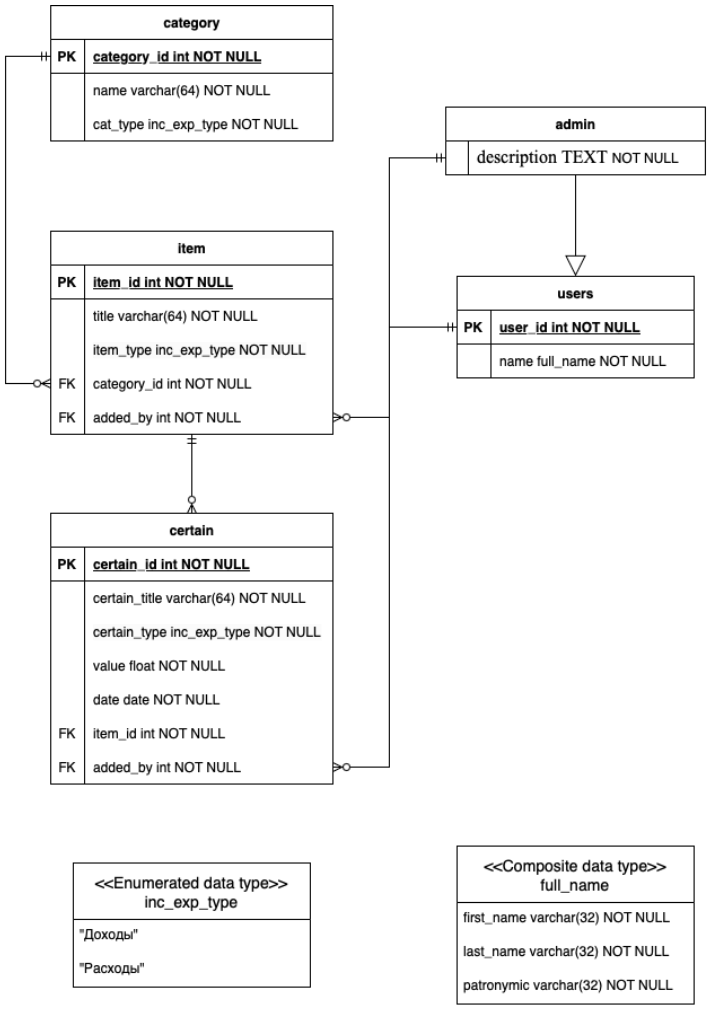
Санкт-Петербург
2024

Задание (13 вариант):

Спроектировать физическую модель базы данных, находящуюся в третьей нормальной форме и включающей наследование и хотя бы один пользовательский тип в соответствии с заданным вариантом. Написать соответствующий скрипт создания базы данных

13. калькулятор бюджета физического лица: пользователь калькулятора (разные уровни доступа), категория дохода (продажа, зарплата), категория расхода (еда, счета за КУ, здоровье…), статьи дохода и расхода, дата расхода/дохода
а. статьи категорий, которые относятся к спорту (содержат слово спорт)
б. месяц, в котором были разные статьи дохода
в. категория, по которой наибольшие расходы в текущем году
г. категория, по которой не было расходов в феврале
д. пользователь, добавивший наименьшее количество статей

Физическая модель:



Скрипт создания базы данных:

```
-- Создание перечислимого типа для категорий доходов и расходов
CREATE TYPE inc_exp_type AS ENUM ('Доходы', 'Расходы');

-- Создание составного типа для полного имени
CREATE TYPE full_name AS (
    first_name VARCHAR(32) NOT NULL,
    last_name VARCHAR(32) NOT NULL,
    patronymic VARCHAR(32) NOT NULL
);

-- Таблица users (просто user зарезервирован в PostgreSQL)
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    name full_name NOT NULL
);

-- Таблица admin, наследующая от users
CREATE TABLE admin (
     PRIMARY KEY (user_id),
    description TEXT NOT NULL
) INHERITS (users);

-- Обновлённая таблица category
CREATE TABLE category (
    category_id SERIAL PRIMARY KEY,
    name VARCHAR(64) NOT NULL,
    cat_type inc_exp_type NOT NULL
);
```

```
-- Обновлённая таблица item
CREATE TABLE item (
    item_id SERIAL PRIMARY KEY,
    title VARCHAR(64) NOT NULL,
    item_type inc_exp_type NOT NULL,
    category_id INT NOT NULL,
    added_by INT NOT NULL,
Foreign key (added_by) REFERENCES admin(user_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_category_id FOREIGN KEY (category_id) REFERENCES category(category_id) ON DELETE RESTRICT ON UPDATE CASCADE
);

-- Новая таблица certain
CREATE TABLE certain (
    certain_id SERIAL PRIMARY KEY,
    cert_title VARCHAR(64) NOT NULL,
    certain_type inc_exp_type NOT NULL,
    value FLOAT NOT NULL,
    date DATE NOT NULL,
    item_id INT NOT NULL,
    added_by INT NOT NULL,
    CONSTRAINT fk_item_id FOREIGN KEY (item_id) REFERENCES item(item_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
```

Триггеры для обеспечения целостности внешних ключей:

```
-- Функция и триггер для проверки наличия пользователя при добавлении или обновлении в certain

CREATE OR REPLACE FUNCTION verify_user_exists_for_certain() RETURNS TRIGGER AS $$

BEGIN

    -- Проверяем существование пользователя в таблице users

    IF NOT EXISTS (

        SELECT 1 FROM users WHERE user_id = NEW.added_by

    ) THEN

        RAISE EXCEPTION 'Пользователь с user_id % не найден', NEW.added_by;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;


CREATE TRIGGER check_user_before_insert_or_update_on_certain

BEFORE INSERT OR UPDATE ON certain

FOR EACH ROW EXECUTE FUNCTION verify_user_exists_for_certain();



-- Функция и триггер для обновления user_id в item и certain при обновлении пользователя

CREATE OR REPLACE FUNCTION update_user_in_item_certain() RETURNS TRIGGER AS $$

BEGIN

    -- Обновляем added_by в item и certain если user обновлён

    UPDATE certain SET added_by = NEW.user_id WHERE added_by = OLD.user_id;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trigger_update_user

AFTER UPDATE ON users

FOR EACH ROW EXECUTE FUNCTION update_user_in_item_certain();


-- Функция и триггер для ограничения удаления users, если ссылки присутствуют в item или certain

CREATE OR REPLACE FUNCTION restrict_delete_user() RETURNS TRIGGER AS $$

BEGIN

   IF EXISTS (

      SELECT 1 FROM item WHERE added_by = OLD.user_id

   ) OR EXISTS (

      SELECT 1 FROM certain WHERE added_by = OLD.user_id

   ) THEN

      RAISE EXCEPTION 'Нельзя удалить пользователя, поскольку у него есть зависимые записи в item или certain';

   END IF;

   RETURN OLD;

END;

$$ LANGUAGE plpgsql;


CREATE TRIGGER trigger_restrict_delete_user

BEFORE DELETE ON users

FOR EACH ROW EXECUTE FUNCTION restrict_delete_user();
```