**Код**

-- Таблица категорий

CREATE TABLE Categories (

category\_id SERIAL PRIMARY KEY,

name VARCHAR(50) UNIQUE NOT NULL,

description TEXT DEFAULT NULL

);

-- Таблица локаций

CREATE TABLE Locations (

location\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

address VARCHAR(255) NOT NULL,

phone VARCHAR(15),

opening\_hours VARCHAR(50),

CONSTRAINT chk\_opening\_hours CHECK (opening\_hours ~ '^[0-9]{2}:[0-9]{2}-[0-9]{2}:[0-9]{2}$' OR opening\_hours IS NULL)

);

-- Таблица сотрудников

CREATE TABLE Employees (

employee\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

position VARCHAR(50) DEFAULT 'Сотрудник',

phone VARCHAR(15),

email VARCHAR(100),

CONSTRAINT chk\_email\_format CHECK (email ~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' OR email IS NULL)

);

-- Таблица владельцев

CREATE TABLE Owners (

owner\_id SERIAL PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

phone VARCHAR(15),

email VARCHAR(100),

address TEXT,

CONSTRAINT chk\_email\_format\_owner CHECK (email ~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' OR email IS NULL)

);

-- Таблица вещей

CREATE TABLE Items (

item\_id SERIAL PRIMARY KEY,

description TEXT NOT NULL,

found\_date DATE NOT NULL,

location\_id INT REFERENCES Locations(location\_id),

category\_id INT REFERENCES Categories(category\_id),

condition VARCHAR(50) DEFAULT 'Неизвестно',

owner\_id INT REFERENCES Owners(owner\_id) ON DELETE SET NULL,

is\_returned BOOLEAN DEFAULT FALSE,

current\_status VARCHAR(50) NOT NULL CHECK (current\_status IN ('Найдено', 'В процессе возврата', 'Возвращено владельцу', 'Передано в полицию')),

days\_since\_found INT DEFAULT 0,

brought\_by VARCHAR(100),

metadata JSONB DEFAULT '{}',

CONSTRAINT chk\_condition CHECK (condition IN ('Неизвестно', 'Новое', 'Хорошее', 'Среднее', 'Использованное'))

);

-- Таблица передачи вещей

CREATE TABLE ItemTransfers (

transfer\_id SERIAL PRIMARY KEY,

item\_id INT REFERENCES Items(item\_id) ON DELETE CASCADE,

brought\_by VARCHAR(100) NOT NULL,

brought\_from VARCHAR(255) NOT NULL,

brought\_time TIMESTAMP DEFAULT NOW(),

accepted\_by INT REFERENCES Employees(employee\_id),

comments TEXT DEFAULT NULL

);

/////////Вставка данных//////

INSERT INTO Categories (name, description) VALUES

('Одежда', 'Куртки, пальто, шапки и другие предметы одежды'),

('Электроника', 'Телефоны, ноутбуки, планшеты'),

('Документы', 'Паспорта, права, банковские карты'),

('Аксессуары', 'Сумки, зонты, украшения'),

('Игрушки', 'Детские игрушки, куклы, машинки'),

('Прочее', 'Все, что не подходит под другие категории');

INSERT INTO Locations (name, address, phone, opening\_hours) VALUES

('Аэропорт Иваново', 'ул. Авиационная, 1', '+7-4932-123456', '00:00-24:00'),

('Железнодорожный вокзал Иваново', 'ул. Вокзальная, 10', '+7-4932-234567', '00:00-24:00'),

('Торговый центр "Серебряный город"', 'ул. Ленина, 55', '+7-4932-345678', '10:00-22:00'),

('Центральный парк', 'ул. Парковая, 3', NULL, '00:00-24:00'),

('Автовокзал Иваново', 'ул. Пушкина, 12', '+7-4932-456789', '05:00-23:00'),

('Музей Ивановского ситца', 'ул. Советская, 29', '+7-4932-567890', '09:00-18:00');

INSERT INTO Employees (name, position, phone, email) VALUES

('Иван Иванов', 'Менеджер', '+7-900-1234567', 'ivanov@example.com'),

('Анна Петрова', 'Сотрудник', '+7-900-2345678', 'petrova@example.com'),

('Ольга Смирнова', 'Сотрудник', '+7-900-3456789', 'smirnova@example.com'),

('Петр Сергеев', 'Администратор', '+7-900-4567890', 'sergeev@example.com');

INSERT INTO Owners (full\_name, phone, email, address) VALUES

('Сергей Павлов', '+7-900-1112233', 'spavlov@example.com', 'г. Иваново, ул. Цветочная, д. 5'),

('Елена Алексеева', '+7-900-2223344', 'ealekseeva@example.com', 'г. Иваново, ул. Зеленая, д. 12'),

('Мария Иванова', '+7-900-3334455', 'mivanova@example.com', 'г. Иваново, ул. Ленина, д. 20');

INSERT INTO Items (description, found\_date, location\_id, category\_id, condition, owner\_id, current\_status, brought\_by) VALUES

('Черный зонт', '2023-10-01', 1, 4, 'Хорошее', NULL, 'Найдено', 'Алексей Иванов'),

('Смартфон Samsung', '2023-10-02', 2, 2, 'Использованное', 1, 'В процессе возврата', 'Ольга Смирнова'),

('Кошелек с документами', '2023-10-03', 3, 3, 'Среднее', 2, 'Возвращено владельцу', 'Иван Иванов');

INSERT INTO ItemTransfers (item\_id, brought\_by, brought\_from, accepted\_by, comments) VALUES

(1, 'Алексей Иванов', 'Железнодорожный вокзал Иваново', 2, 'Найден в зале ожидания.'),

(2, 'Ольга Смирнова', 'Торговый центр "Серебряный город"', 1, 'Передано на кассе.'),

(3, 'Иван Иванов', 'Автовокзал Иваново', 3, 'Кошелек найден на лавке.');

**Запросы**

**1. Получение всех найденных вещей с указанием места и категории.**

SELECT Items.description, Locations.name AS location, Categories.name AS category

FROM Items

JOIN Locations ON Items.location\_id = Locations.location\_id

JOIN Categories ON Items.category\_id = Categories.category\_id;

**2. Поиск вещей, которые найдены на станции метро.**

SELECT Items.description

FROM Items

JOIN Locations ON Items.location\_id = Locations.location\_id

WHERE Locations.name = 'Аэропорт Иваново';

**3. Получение всех отчетов с указанием вещи, статуса и сотрудника.**

SELECT ItemTransfers.transfer\_id, Items.description, ItemTransfers.brought\_by, Employees.name AS accepted\_by, ItemTransfers.brought\_time

FROM ItemTransfers

JOIN Items ON ItemTransfers.item\_id = Items.item\_id

JOIN Employees ON ItemTransfers.accepted\_by = Employees.employee\_id;

**4. Подсчет общего количества забытых вещей в каждой категории.**

SELECT Categories.name, COUNT(Items.item\_id) AS total\_items

FROM Items

JOIN Categories ON Items.category\_id = Categories.category\_id

GROUP BY Categories.name;

**5. Получение списка сотрудников и количества отчетов, созданных каждым из них.**

SELECT Employees.name, COUNT(ItemTransfers.transfer\_id) AS accepted\_items

FROM Employees

JOIN ItemTransfers ON Employees.employee\_id = ItemTransfers.accepted\_by

GROUP BY Employees.name;

**6. Список вещей, которые возвращены владельцам.**

SELECT Items.description

FROM Items

WHERE Items.current\_status = 'Возвращено владельцу';

**7. Поиск вещей, которые были найдены за последние 7 дней.**

SELECT Items.description

FROM Items

WHERE Items.found\_date >= CURRENT\_DATE - INTERVAL '7 days';

**8. Создание представления для всех вещей, найденных в аэропорту.**

CREATE VIEW AirportFoundItems AS

SELECT Items.description, Items.found\_date

FROM Items

JOIN Locations ON Items.location\_id = Locations.location\_id

WHERE Locations.name = 'Аэропорт Иваново';

**9. Список всех вещей и количество дней с момента нахождения.**

SELECT Items.description, Items.days\_since\_found

FROM Items;

**10. Список всех статусов и количество вещей в каждом статусе.**

SELECT Items.current\_status, COUNT(Items.item\_id) AS count

FROM Items

GROUP BY Items.current\_status;

**11. Поиск всех вещей, которые были найдены сотрудником "Иван Иванов".**

SELECT Items.description

FROM Items

JOIN ItemTransfers ON Items.item\_id = ItemTransfers.item\_id

JOIN Employees ON ItemTransfers.accepted\_by = Employees.employee\_id

WHERE Employees.name = 'Иван Иванов';

**12. Список всех вещей с указанием сотрудника, который их нашел.**

SELECT Items.description, Employees.name AS accepted\_by

FROM Items

JOIN ItemTransfers ON Items.item\_id = ItemTransfers.item\_id

JOIN Employees ON ItemTransfers.accepted\_by = Employees.employee\_id;

**13. Список всех вещей, найденных в торговом центре.**

SELECT Items.description

FROM Items

JOIN Locations ON Items.location\_id = Locations.location\_id

WHERE Locations.name = 'Торговый центр "Серебряный город"';

**14. Поиск всех вещей, которые еще не возвращены владельцам.**

SELECT Items.description

FROM Items

WHERE Items.current\_status != 'Возвращено владельцу';

**15. Получение списка вещей, отсортированных по дате нахождения (от новых к старым).**

SELECT Items.description, Items.found\_date

FROM Items

ORDER BY Items.found\_date DESC;

**16. Выбор всех вещей, найденных в октябре 2023 года.**

SELECT Items.description

FROM Items

WHERE Items.found\_date BETWEEN '2023-10-01' AND '2023-10-31';

**17. Получение всех отчетов с указанием текущего статуса.**

SELECT ItemTransfers.transfer\_id, Items.description, Employees.name AS accepted\_by, ItemTransfers.brought\_time

FROM ItemTransfers

JOIN Items ON ItemTransfers.item\_id = Items.item\_id

JOIN Employees ON ItemTransfers.accepted\_by = Employees.employee\_id;

**18. Выбор всех вещей, у которых категория "Одежда".**

SELECT Items.description

FROM Items

JOIN Categories ON Items.category\_id = Categories.category\_id

WHERE Categories.name = 'Одежда';

**19. Список всех уникальных категорий вещей.**

SELECT DISTINCT Categories.name

FROM Categories;

**20. Вывод списка всех мест и количество вещей, найденных в каждом из них.**

SELECT Locations.name, COUNT(Items.item\_id) AS item\_count

FROM Locations

JOIN Items ON Locations.location\_id = Items.location\_id

GROUP BY Locations.name;

**Триггер**

-- Создание функции для обновления days\_since\_found

CREATE OR REPLACE FUNCTION update\_days\_since\_found()

RETURNS TRIGGER AS $$

BEGIN

-- Вычисляем разницу между текущей датой и found\_date

NEW.days\_since\_found = (CURRENT\_DATE - NEW.found\_date)::INT;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Создание триггера, который вызывает функцию

CREATE TRIGGER trg\_update\_days\_since\_found

BEFORE INSERT OR UPDATE ON Items

FOR EACH ROW

EXECUTE FUNCTION update\_days\_since\_found();

Для поля days\_since\_found в таблице Items создаем триггер, который будет автоматически вычислять количество дней, прошедших с даты нахождения вещи:

Проверка работы триггера

--создать новую запись

INSERT INTO Items (description, found\_date, location\_id, category\_id, condition, current\_status, brought\_by)

VALUES ('Тестовый предмет', '2023-11-15', 1, 4, 'Новое', 'Найдено', 'Иван Иванов');

-- Проверьте результат

SELECT \* FROM Items WHERE description = 'Тестовый предмет';

--проверить запись в таблице

UPDATE Items

SET found\_date = '2023-11-10'

WHERE description = 'Тестовый предмет';

-- Проверьте снова

SELECT \* FROM Items WHERE description = 'Тестовый предмет';

-- Обновление всех записей в таблице Items, чтобы триггер пересчитал days\_since\_found

UPDATE Items

SET days\_since\_found = (CURRENT\_DATE - found\_date)::INT;

Индексы -

-- Индекс для ускорения поиска по текущему статусу

CREATE INDEX idx\_items\_current\_status ON Items(current\_status);

-- Индекс для ускорения поиска по дате нахождения

CREATE INDEX idx\_items\_found\_date ON Items(found\_date);

-- Индекс для ускорения поиска по имени владельца

CREATE INDEX idx\_owners\_full\_name ON Owners(full\_name);

-- Индекс для ускорения поиска по месту нахождения

CREATE INDEX idx\_locations\_name ON Locations(name);

-- Индекс для ускорения соединений между Items и ItemTransfers

CREATE INDEX idx\_item\_transfers\_item\_id ON ItemTransfers(item\_id);

**Триггер для автоматического расчета поля days\_since\_found**

Триггер, созданный ранее, автоматизирует расчет days\_since\_found, что снижает необходимость ручных расчетов и улучшает целостность данных

Процедуры -

1. **Добавление новой вещи**:

CREATE OR REPLACE PROCEDURE add\_new\_item(

p\_description TEXT,

p\_found\_date DATE,

p\_location\_id INT,

p\_category\_id INT,

p\_condition VARCHAR,

p\_current\_status VARCHAR,

p\_brought\_by VARCHAR,

p\_brought\_from VARCHAR,

p\_accepted\_by INT,

p\_comments TEXT

)

LANGUAGE plpgsql AS $$

BEGIN

-- Добавление вещи в Items

INSERT INTO Items (description, found\_date, location\_id, category\_id, condition, current\_status, brought\_by)

VALUES (p\_description, p\_found\_date, p\_location\_id, p\_category\_id, p\_condition, p\_current\_status, p\_brought\_by);

-- Получение ID добавленной вещи

INSERT INTO ItemTransfers (item\_id, brought\_by, brought\_from, accepted\_by, comments)

VALUES (currval('items\_item\_id\_seq'), p\_brought\_by, p\_brought\_from, p\_accepted\_by, p\_comments);

END;

$$;

2. **Обновление статуса вещи Обновляет текущий статус вещи по её ID**

CREATE OR REPLACE PROCEDURE update\_item\_statusa(

p\_item\_id INT,

p\_new\_status VARCHAR

)

LANGUAGE plpgsql AS $$

BEGIN

UPDATE Items

SET current\_status = p\_new\_status

WHERE item\_id = p\_item\_id;

END;

$$;

3. **Получение списка вещей по категории Возвращает список всех вещей для указанной категории.**

CREATE OR REPLACE PROCEDURE get\_items\_by\_category(

p\_category\_name VARCHAR

)

LANGUAGE plpgsql AS $$

BEGIN

SELECT Items.description, Items.found\_date

FROM Items

JOIN Categories ON Items.category\_id = Categories.category\_id

WHERE Categories.name = p\_category\_name;

END;

$$;

4. **Поиск вещей по владельцу**

CREATE OR REPLACE PROCEDURE get\_items\_by\_owner(

p\_owner\_name VARCHAR

)

LANGUAGE plpgsql AS $$

BEGIN

SELECT Items.description, Items.current\_status

FROM Items

JOIN Owners ON Items.owner\_id = Owners.owner\_id

WHERE Owners.full\_name = p\_owner\_name;

END;

$$;

5. **Удаление старых записей**

CREATE OR REPLACE PROCEDURE delete\_old\_items()

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM Items

WHERE found\_date < CURRENT\_DATE - INTERVAL '365 days';

END;

$$;

Вызов процедур

1.

CALL add\_new\_item(

'Новая вещь',

'2023-11-01',

1,

2,

'Новое',

'Найдено',

'Иван Иванов',

'Аэропорт Иваново',

1,

'Передано сотруднику'

);

2.

CALL update\_item\_statusa(1, 'Возвращено владельцу');

3.

CALL get\_items\_by\_category('Одежда');

4.

CALL get\_items\_by\_owner('Сергей Павлов');

5.

CALL delete\_old\_items();

Программа для работы с интерфейсом на python

import psycopg2  
  
  
def connect\_db():  
 *"""Подключение к базе данных PostgreSQL."""* return psycopg2.connect(  
 dbname=" ",  
 user=" ",   
 password="", # Укажите ваш пароль  
 host=" ", # Укажите ваш хост  
 port="" # Порт PostgreSQL по умолчанию  
 )  
  
  
def add\_category(name, description):  
 *"""Добавление новой категории."""* conn = connect\_db()  
 cursor = conn.cursor()  
 cursor.execute("INSERT INTO Categories (name, description) VALUES (%s, %s);", (name, description))  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
  
def get\_items\_by\_location(location\_name):  
 *"""Получение всех вещей по местоположению."""* conn = connect\_db()  
 cursor = conn.cursor()  
 query = """  
 SELECT Items.description, Items.found\_date, Categories.name AS category  
 FROM Items  
 JOIN Locations ON Items.location\_id = Locations.location\_id  
 JOIN Categories ON Items.category\_id = Categories.category\_id  
 WHERE Locations.name = %s;  
 """  
 cursor.execute(query, (location\_name,))  
 items = cursor.fetchall()  
 cursor.close()  
 conn.close()  
 return items  
  
  
def update\_item\_status(item\_id, new\_status):  
 *"""Обновление статуса вещи."""* conn = connect\_db()  
 cursor = conn.cursor()  
 query = "CALL update\_item\_status(%s, %s);"  
 cursor.execute(query, (item\_id, new\_status))  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
  
def delete\_item(item\_id):  
 *"""Удаление вещи."""* conn = connect\_db()  
 cursor = conn.cursor()  
 cursor.execute("DELETE FROM Items WHERE item\_id = %s;", (item\_id,))  
 conn.commit()  
 cursor.close()  
 conn.close()  
  
  
def get\_items():  
 *"""Получение всех вещей."""* conn = connect\_db()  
 cursor = conn.cursor()  
 cursor.execute("SELECT description, found\_date, current\_status FROM Items;")  
 items = cursor.fetchall()  
 cursor.close()  
 conn.close()  
 return items  
  
  
def main():  
 *"""Основная логика программы."""* while True:  
 print("\nВыберите действие:")  
 print("1. Добавить категорию")  
 print("2. Получить вещи по местоположению")  
 print("3. Обновить статус вещи")  
 print("4. Удалить вещь")  
 print("5. Показать все вещи")  
 print("6. Выйти")  
  
 choice = input("Введите номер действия: ")  
  
 if choice == "1":  
 name = input("Введите имя категории: ")  
 description = input("Введите описание категории: ")  
 add\_category(name, description)  
 print(f"Категория '{name}' добавлена.")  
  
 elif choice == "2":  
 location\_name = input("Введите имя местоположения: ")  
 items = get\_items\_by\_location(location\_name)  
 print("Найденные вещи:")  
 for item in items:  
 print(f"Описание: {item[0]}, Дата: {item[1]}, Категория: {item[2]}")  
  
 elif choice == "3":  
 item\_id = int(input("Введите ID вещи: "))  
 new\_status = input("Введите новый статус: ")  
 update\_item\_status(item\_id, new\_status)  
 print(f"Статус вещи с ID {item\_id} обновлен на '{new\_status}'.")  
  
 elif choice == "4":  
 item\_id = int(input("Введите ID вещи для удаления: "))  
 delete\_item(item\_id)  
 print(f"Вещь с ID {item\_id} удалена.")  
  
 elif choice == "5":  
 items = get\_items()  
 print("Все вещи:")  
 for item in items:  
 print(f"Описание: {item[0]}, Дата нахождения: {item[1]}, Статус: {item[2]}")  
  
 elif choice == "6":  
 print("Выход из программы.")  
 break  
  
 else:  
 print("Неверный выбор. Попробуйте снова.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Отчет**

**1. Подробное описание структуры базы данных**

**1. Categories – Таблица категорий**

Эта таблица хранит информацию о категориях вещей (например, одежда, электроника).

| **Поле** | **Тип данных** | **Описание** | **Ограничения** |
| --- | --- | --- | --- |
| category\_id | SERIAL | Уникальный идентификатор категории | PRIMARY KEY |
| name | VARCHAR(50) | Название категории | UNIQUE, NOT NULL |
| description | TEXT | Описание категории | NULL по умолчанию |

**2. Locations – Таблица местоположений**

Содержит информацию о местах, где были найдены вещи.

| **Поле** | **Тип данных** | **Описание** | **Ограничения** |
| --- | --- | --- | --- |
| location\_id | SERIAL | Уникальный идентификатор местоположения | PRIMARY KEY |
| name | VARCHAR(100) | Название местоположения | NOT NULL |
| address | VARCHAR(255) | Адрес местоположения | NOT NULL |
| phone | VARCHAR(15) | Контактный телефон | NULL по умолчанию |
| opening\_hours | VARCHAR(50) | Часы работы | CHECK (формат времени) |

**3. Employees – Таблица сотрудников**

Содержит информацию о сотрудниках, принимающих вещи или обрабатывающих заявки.

| **Поле** | **Тип данных** | **Описание** | **Ограничения** |
| --- | --- | --- | --- |
| employee\_id | SERIAL | Уникальный идентификатор сотрудника | PRIMARY KEY |
| name | VARCHAR(100) | Имя сотрудника | NOT NULL |
| position | VARCHAR(50) | Должность сотрудника | DEFAULT 'Сотрудник' |
| phone | VARCHAR(15) | Телефон | NULL по умолчанию |
| email | VARCHAR(100) | Электронная почта | NULL по умолчанию, CHECK (формат) |

**4. Owners – Таблица владельцев**

Содержит информацию о владельцах забытых вещей.

| **Поле** | **Тип данных** | **Описание** | **Ограничения** |
| --- | --- | --- | --- |
| owner\_id | SERIAL | Уникальный идентификатор владельца | PRIMARY KEY |
| full\_name | VARCHAR(100) | Полное имя владельца | NOT NULL |
| phone | VARCHAR(15) | Контактный телефон владельца | NULL по умолчанию |
| email | VARCHAR(100) | Электронная почта | NULL по умолчанию, CHECK (формат) |
| address | TEXT | Адрес владельца | NULL по умолчанию |

**5. Items – Таблица вещей**

Содержит информацию о найденных вещах, их статусе, местоположении и категории.

| **Поле** | **Тип данных** | **Описание** | **Ограничения** |
| --- | --- | --- | --- |
| item\_id | SERIAL | Уникальный идентификатор вещи | PRIMARY KEY |
| description | TEXT | Описание вещи | NOT NULL |
| found\_date | DATE | Дата нахождения | NOT NULL |
| location\_id | INT | Идентификатор местоположения | REFERENCES Locations |
| category\_id | INT | Идентификатор категории | REFERENCES Categories |
| condition | VARCHAR(50) | Состояние вещи | DEFAULT 'Неизвестно', CHECK |
| owner\_id | INT | Идентификатор владельца | REFERENCES Owners, NULL |
| is\_returned | BOOLEAN | Возвращена ли вещь владельцу | DEFAULT FALSE |
| current\_status | VARCHAR(50) | Текущий статус вещи | CHECK (значения из списка) |
| days\_since\_found | INT | Количество дней с момента нахождения | Рассчитывается триггером |
| brought\_by | VARCHAR(100) | Кто принёс вещь | NULL по умолчанию |
| metadata | JSONB | Дополнительная информация | NULL по умолчанию |

**6. ItemTransfers – Таблица передач вещей**

Содержит информацию о передаче вещей, включая время, место и сотрудника.

| **Поле** | **Тип данных** | **Описание** | **Ограничения** |
| --- | --- | --- | --- |
| transfer\_id | SERIAL | Уникальный идентификатор передачи | PRIMARY KEY |
| item\_id | INT | Идентификатор вещи | REFERENCES Items |
| brought\_by | VARCHAR(100) | Кто принёс вещь | NOT NULL |
| brought\_from | VARCHAR(255) | Откуда принесли вещь | NOT NULL |
| brought\_time | TIMESTAMP | Время, когда принесли вещь | DEFAULT NOW() |
| accepted\_by | INT | Сотрудник, принявший вещь | REFERENCES Employees |
| comments | TEXT | Комментарии | NULL по умолчанию |

Связи между таблицами

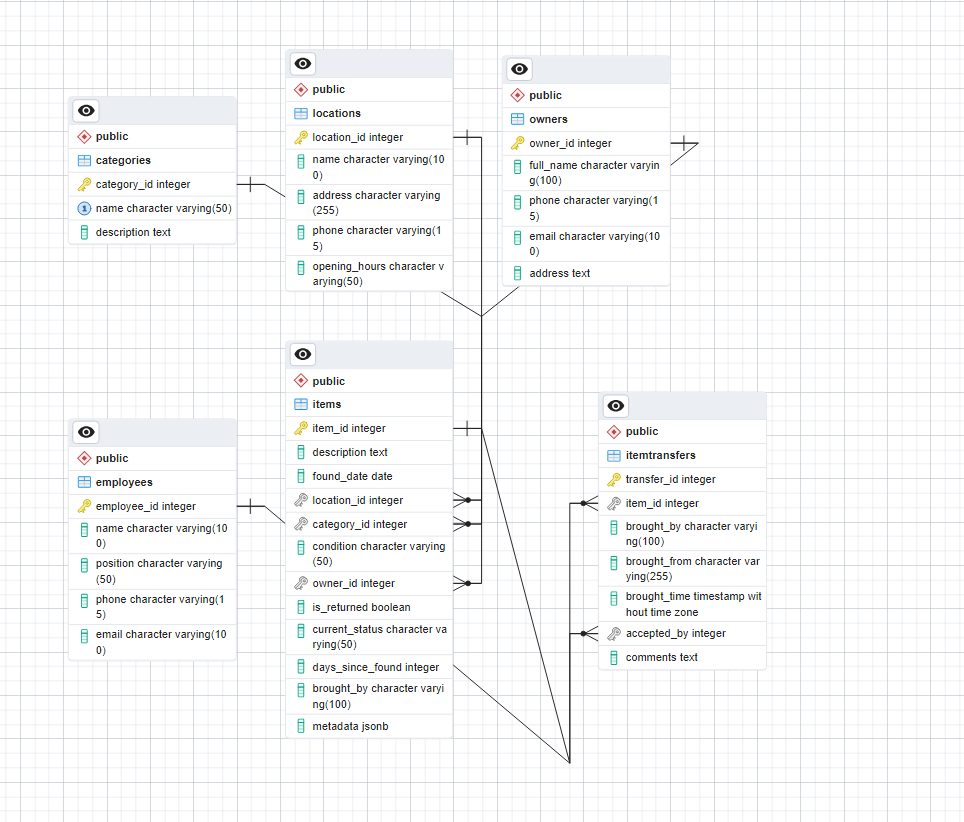
1. **Items ↔ Locations**:
   * Каждый предмет относится к одному месту нахождения.
   * Связь через location\_id.
2. **Items ↔ Categories**:
   * Каждый предмет относится к одной категории.
   * Связь через category\_id.
3. **Items ↔ Owners**:
   * Каждый предмет может быть связан с одним владельцем.
   * Связь через owner\_id.
4. **Items ↔ ItemTransfers**:
   * Каждая передача относится к одному предмету.
   * Связь через item\_id.
5. **ItemTransfers ↔ Employees**:
   * Каждый акт передачи связан с одним сотрудником.
   * Связь через accepted\_by.

**Логика использования базы данных**

1. **Добавление новой вещи**:
   * В таблицу Items добавляется запись с описанием, местоположением, категорией и статусом.
   * В таблицу ItemTransfers добавляется запись о передаче вещи.
2. **Обновление статуса вещи**:
   * Поле current\_status в таблице Items обновляется с помощью хранимой процедуры.
3. **Поиск вещей**:
   * Запросы могут использовать location\_id, category\_id или current\_status для фильтрации.
4. **Отслеживание передач**:
   * Таблица ItemTransfers позволяет отслеживать, кто, когда и откуда принёс вещь.
5. **Возврат владельцу**:
   * Обновляется поле is\_returned и current\_status в таблице Items.

**2. Схема базы данных**

*(Здесь вставьте схему базы данных, созданную в ERD-диаграмме или другой подходящей форме, например, с помощью pgAdmin.)*



**3. Список созданных запросов с их описанием**

**Расписал выше**

* 1. **Список созданных процедур, функций, представлений, триггеров с их описанием, параметрами работы**

**Расписал выше**

* 1. **Рекомендации по работе с базой данных – оптимизация с помощью индексов**

**6. Описание работы с графическим интерфейсом вашей программы**

**Интерфейс программы:**

* Программа написана на Python и предоставляет текстовый интерфейс для взаимодействия с базой данных.
* Пользователи могут добавлять категории, получать список вещей по местоположению, обновлять статус и удалять отчеты через меню.
* Обработка пользовательских запросов осуществляется с помощью функций, реализованных на Python.

**. Приложение – SQL код всех запросов к вашей базе данных**

*(Здесь вставьте все SQL запросы, используемые в вашей базе данных, включая создание таблиц, триггеров, функций и хранимых процедур.)*

Есть выше