МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЕТ

Инфологическое проектирование модели базы данных

«Информационная система аптеки»

Студента Нелтанова Баярто Васильевича

группы 21209

Новосибирск

Оглавление

Инфологическое проектирование

Сущности

Стержневые

medicine - медикамент, который входит в рецепты.
substance - вещество, ингредиент, из которого состоят лекарства.
receipt - рецепт, предоставляемый клиентом аптеке.

Обозначающие

production_technology - технология производства медикамента, включающая в себя описание и время изготовления медикамента.

Ассоциативные

medicine_list - связь рецепта и медикамента.
medicine_composition - связь медикамента и ингредиента.

Характеристические

order - заказ, созданный на основе рецепта от клиента.
doctor - лечащий врач, который выдает рецепт пациенту.
patient - пациент, которому выписывается рецепт от лечащего врача.
customer - клиент - делает заказ на медикаменты.
medicine_warehouse - склад медикаментов.
substance_warehouse - склад ингридиентов.
medicine_usage_statistics - статистика использования медикаментов.
substance_usage_statistics - статистика использования ингредиентов.
local_medicine - лекарство, изготовляемое аптекой.
imported_medicine - готовое лекарство.

Атрибуты сущностей

- **medicine:**
  - id (int, PRIMARY KEY) - указывающий
  - name (varchar, NOT NULL) - описательный
  - type (medicine_type, NOT NULL) - описатальный
  - price (float, NOT NULL) - описательный
  - expiration_date (date, NOT NULL) - описательный

- **substance:**
  - id (int, PRIMARY KEY) - указывающий
  - name (varchar, NOT NULL) - описательный
  - price (float, NOT NULL) - описательный

- **medicine_warehouse:**
  - id (int, PRIMARY KEY) - указывающий
  - total_amount (int, NOT NULL) - описательный
  - critical_limit (int, NOT NULL) - описательный
  - medicine_id (int, NOT NULL, FOREIGN KEY REFERENCES medicine(id)) - вспомогательный

- **substance_warehouse:**
  - id (int, PRIMARY KEY) - указывающий
  - total_amount (int, NOT NULL) - описательный
  - critical_limit (int, NOT NULL) - описательный
  - substance_id (int, NOT NULL, FOREIGN KEY REFERENCES substance(id)) - вспомогательный

- **medicine_list:**
  - id (int, PRIMARY KEY) - указывающий
  - receipt_id (int, NOT NULL, FOREIGN KEY REFERENCES receipt(id)) - вспомогательный
  - medicine_id (int, NOT NULL, FOREIGN KEY REFERENCES medicine(id)) - вспомогательный
  - quantity_used (float, NOT NULL) - описательный

- **medicine_composition:**
  - id (int, PRIMARY KEY) - указывающий
  - substance_id (int, NOT NULL, FOREIGN KEY REFERENCES substance(id)) - вспомогательный
  - medicine_id (int, NOT NULL, FOREIGN KEY REFERENCES local_medicine(id)) - вспомогательный
  - required_quantity (float, NOT NULL) - описательный

- **medicine_usage_statistics:**
    - id (int, PRIMARY KEY) - указывающий
    - medicine_id (int, NOT NULL, FOREIGN KEY REFERENCES medicine(id)) - вспомогательный
    - quantity_used (float, NOT NULL) - описательный
    - usage_time (timestamp, NOT NULL) - описательный

- **substance_usage_statistics:**
    - id (int, PRIMARY KEY) - указывающий
    - substance_id (int, NOT NULL, FOREIGN KEY REFERENCES substance(id)) - вспомогательный
    - quantity_used (float, NOT NULL) - описательный
    - usage_time (timestamp, NOT NULL) - описательный

- **order:**
    - id (int, PRIMARY KEY) - указывающий
    - customer_id (int, NOT NULL, FOREIGN KEY REFERENCES customer(id)) - вспомогательный
    - receipt_id (int, NOT NULL, FOREIGN KEY REFERENCES receipt(id)) - вспомогательный
    - order_date (timestamp, NOT NULL) - описательный
    - production_date (date, NOT NULL) - описательный
    - status (order_status, NOT NULL) - описательный

- **imported_medicine:**
    - id (int, PRIMARY KEY) - указывающий
    - medicine_id (int, NOT NULL, FOREIGN KEY REFERENCES medicine(id)) - вспомогательный

- **local_medicine:**
    - id (int, PRIMARY KEY) - указывающий
    - medicine_id (int, NOT NULL, FOREIGN KEY REFERENCES medicine(id)) - вспомогательный
    - production_techology (int, NOT NULL, FOREIGN KEY REFERENCES production_techonology(id)) - вспомогательный

- **receipt:**
    - id (int, PRIMARY KEY) - указывающий
    - doctor_id (int, NOT NULL, FOREIGN KEY REFERENCES doctor(id)) - вспомогательный
    - patient_id (int, NOT NULL, FOREIGN KEY REFERENCES patient(id)) - вспомогательный

- **doctor:**

    - id (int, PRIMARY KEY) - указывающий
    - surname (varchar, NOT NULL) - описательный
    - name (varchar, NOT NULL) - описательный
    - middle_name (varchar) - описательный

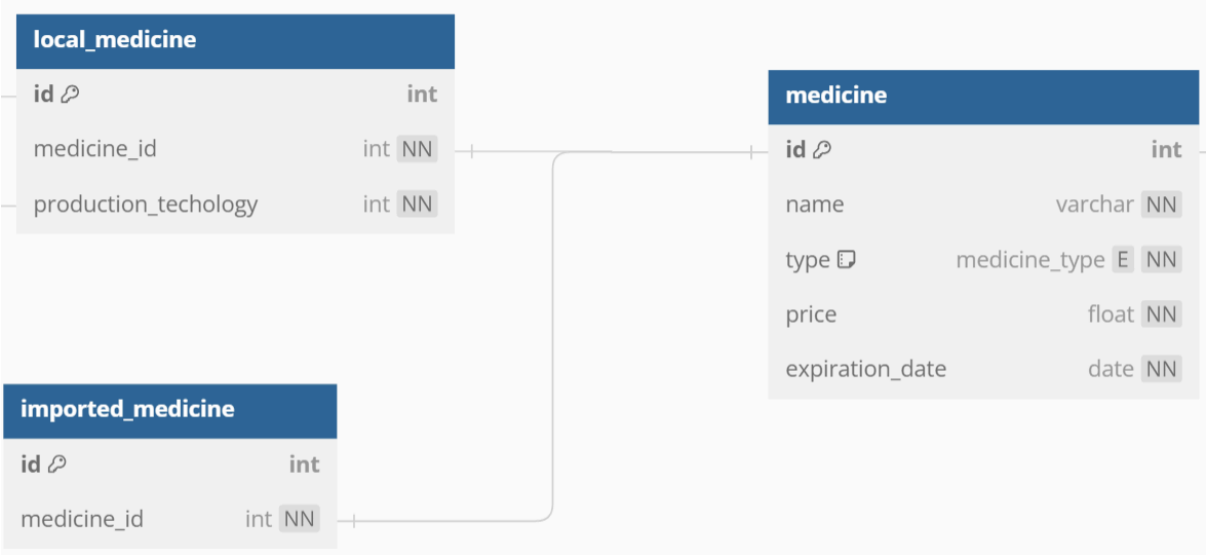
- **patient:**
    - id (int, PRIMARY KEY) - указывающий
    - surname (varchar, NOT NULL) - описательный
    - name (varchar, NOT NULL) - описательный
    - middle_name (varchar) - описательный
    - age (int) - описательный
    - diagnosis (varchar) - описательный

- **production_techonology:**
    - id (int, PRIMARY KEY) - указывающий
    - method_of_production (varchar, NOT NULL) - описательный
    - time_to_product (varchar, NOT NULL) - описательный

- **customer:**
    - id (int, PRIMARY KEY) - указывающий
    - surname (varchar, NOT NULL) - описательный
    - name (varchar, NOT NULL) - описательный
    - middle_name (varchar) - описательный
    - phone_number (varchar) - описательный
    - address (varchar) - описательный

Сущности вида супертип/подтип

medicine - супертип
local_medicine - подтип **medicine**
imported_medicine - подтип **medicine**



Связи между сущностями

1. medicine_list - receipt
    - Тип связи: M:1 (многие к одному)
    - "medicine_list.receipt_id" -> "receipt.id"
    - Описание: Каждая запись в таблице medicine_list относится к одному рецепту, но один рецепт может включать множество записей в medicine_list.
2. medicine_list - medicine
    - Тип связи: M:1 (многие к одному)
    - "medicine_list.medicine_id" -> "medicine.id"
    - Описание: Каждая запись в таблице medicine_list относится к одному лекарству, но одно лекарство может присутствовать в множестве записей medicine_list.
3. receipt - doctor
    - Тип связи: M:1 (многие к одному)
    - "receipt.doctor_id" -> "doctor.id"
    - Описание: Каждый рецепт назначается одним доктором, но один доктор может назначить множество рецептов.
4. receipt - patient
    - Тип связи: M:1 (многие к одному)
    - "receipt.patient_id" -> "patient.id"
    - Описание: Каждый рецепт назначается одному пациенту, но один пациент может иметь множество рецептов.
5. medicine_warehouse - medicine
    - Тип связи: M:1 (многие к одному)
    - "medicine_warehouse.medicine_id" -> "medicine.id"
    - Описание: Каждая запись в таблице medicine_warehouse относится к одному лекарству, но одно лекарство может присутствовать в множестве записей medicine_warehouse.
6. substance_warehouse - substance
    - Тип связи: M:1 (многие к одному)
    - "substance_warehouse.substance_id" -> "substance.id"
    - Описание: Каждая запись в таблице substance_warehouse относится к одному веществу, но одно вещество может присутствовать в множестве записей substance_warehouse.
7. order - customer
    - Тип связи: M:1 (многие к одному)
    - "order.customer_id" -> "customer.id"
    - Описание: Каждый заказ сделан одним покупателем, но один покупатель может сделать множество заказов.
8. order - receipt
    - Тип связи: M:1 (многие к одному)
    - "order.receipt_id" -> "receipt.id"

- Описание: Каждый заказ связан с одним рецептом, но один рецепт может быть связан с множеством заказов.

9. medicine_composition - substance
    - Тип связи: M:1 (многие к одному)
    - "medicine_composition.substance_id" -> "substance.id"
    - Описание: Каждая запись в таблице medicine_composition относится к одному веществу, но одно вещество может присутствовать в множестве записей medicine_composition.
10. medicine_composition - local_medicine
    - Тип связи: M:1 (многие к одному)
    - "medicine_composition.medicine_id" -> "local_medicine.id"
    - Описание: Каждая запись в таблице medicine_composition относится к одному локальному лекарству, но одно локальное лекарство может присутствовать в множестве записей medicine_composition.
11. imported_medicine - medicine
    - Тип связи: 1:1 (один к одному)
    - "imported_medicine.medicine_id" -> "medicine.id"
    - Описание: Каждое импортное лекарство относится к одному лекарству, и одно лекарство может быть только одним импортным лекарством.
12. local_medicine - medicine
    - Тип связи: 1:1 (один к одному)
    - "local_medicine.medicine_id" -> "medicine.id"
    - Описание: Каждое локальное лекарство относится к одному лекарству, и одно лекарство может быть только одним локальным лекарством.
13. local_medicine - production_techonology
    - Тип связи: M:1 (многие к одному)
    - "local_medicine.production_techology" -> "production_techonology.id"
    - Описание: Каждое локальное лекарство имеет одну технологию производства, но одна технология производства может использоваться для множества локальных лекарств.
14. medicine_usage_statistics - medicine
    - Тип связи: M:1 (многие к одному)
    - "medicine_usage_statistics.medicine_id" -> "medicine.id"
    - Описание: Каждая запись в таблице medicine_usage_statistics относится к одному лекарству, но одно лекарство может иметь множество записей в таблице статистики использования.
15. substance_usage_statistics - substance
    - Тип связи: M:1 (многие к одному)
    - "substance_usage_statistics.substance_id" -> "substance.id"
    - Описание: Каждая запись в таблице substance_usage_statistics относится к одному веществу, но одно вещество может иметь множество записей в таблице статистики использования.

ER-диаграмма модели базы данных

Ссылка на диаграмму.

Логическое проектирование

Приведение к нормальной форме

Все таблицы приведены к 3-ей нормальной форме.

- Таблицы находятся во второй нормальной форме.
- Каждый неключевой атрибут зависит только от ключа таблицы непосредственно, а не через другие атрибуты.
- Каждый атрибут имеет прямую функциональную зависимость от первичного ключа и не зависит от других неключевых атрибутов.

Реляционная модель данных

1. medicine_type (ENUM)
    - Значения: 'pill', 'ointment', 'tincture', 'mixture', 'solution', 'powder'
2. order_status (ENUM)
    - Значения: 'in_production', 'done'
3. medicine
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `name` varchar NOT NULL (название лекарства)
        - `type` medicine_type NOT NULL (тип лекарства)
        - `price` float NOT NULL (цена лекарства)
        - `expiration_date` date NOT NULL (срок годности лекарства)
4. substance
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `name` varchar NOT NULL (название вещества)
        - `price` float NOT NULL (цена вещества)
5. medicine_list
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `receipt_id` int NOT NULL (ссылка на рецепт)
        - `medicine_id` int NOT NULL (ссылка на лекарство)
        - `quantity_used` float NOT NULL (используемое количество лекарства)
    - Ограничения:
        - FOREIGN KEY (`receipt_id`) REFERENCES `receipt`(`id`)
        - FOREIGN KEY (`medicine_id`) REFERENCES `medicine`(`id`)
6. receipt
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `doctor_id` int NOT NULL (ссылка на врача)
        - `patient_id` int NOT NULL (ссылка на пациента)
    - Ограничения:
        - FOREIGN KEY (`doctor_id`) REFERENCES `doctor`(`id`)
        - FOREIGN KEY (`patient_id`) REFERENCES `patient`(`id`)
7. doctor
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `surname` varchar NOT NULL (фамилия врача)
        - `name` varchar NOT NULL (имя врача)

        - `middle_name` varchar (отчество врача)
8. patient
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `surname` varchar NOT NULL (фамилия пациента)
        - `name` varchar NOT NULL (имя пациента)
        - `middle_name` varchar (отчество пациента)
        - `age` int (возраст пациента)
        - `diagnosis` varchar (диагноз пациента)
9. medicine_warehouse
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `total_amount` float NOT NULL (общее количество лекарства)
        - `critical_limit` float NOT NULL (критический лимит лекарства)
        - `medicine_id` int NOT NULL (ссылка на лекарство)
    - Ограничения:
        - FOREIGN KEY (`medicine_id`) REFERENCES `medicine`(`id`)
10. substance_warehouse
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `total_amount` float NOT NULL (общее количество вещества)
        - `critical_limit` float NOT NULL (критический лимит вещества)
        - `substance_id` int NOT NULL (ссылка на вещество)
    - Ограничения:
        - FOREIGN KEY (`substance_id`) REFERENCES `substance`(`id`)
11. production_techonology
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `method_of_production` varchar NOT NULL (метод производства)
        - `time_to_product` varchar NOT NULL (время на производство)
12. customer
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `surname` varchar NOT NULL (фамилия покупателя)
        - `name` varchar NOT NULL (имя покупателя)
        - `middle_name` varchar (отчество покупателя)
        - `phone_number` varchar (телефонный номер покупателя)
        - `address` varchar (адрес покупателя)
13. order

- Атрибуты:
    - `id` int PRIMARY KEY (уникальный идентификатор)
    - `customer_id` int NOT NULL (ссылка на покупателя)
    - `receipt_id` int NOT NULL (ссылка на рецепт)
    - `order_date` timestamp NOT NULL (дата заказа)
    - `production_date` date NOT NULL (дата производства)
    - `status` order_status NOT NULL (статус заказа)
- Ограничения:
    - FOREIGN KEY (`customer_id`) REFERENCES `customer`(`id`)
    - FOREIGN KEY (`receipt_id`) REFERENCES `receipt`(`id`)

14. medicine_composition
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `substance_id` int NOT NULL (ссылка на вещество)
        - `medicine_id` int NOT NULL (ссылка на лекарство)
        - `required_quantity` float NOT NULL (необходимое количество вещества)
    - Ограничения:
        - FOREIGN KEY (`substance_id`) REFERENCES `substance`(`id`)
        - FOREIGN KEY (`medicine_id`) REFERENCES `local_medicine`(`id`)
15. imported_medicine
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `medicine_id` int NOT NULL (ссылка на лекарство)
    - Ограничения:
        - FOREIGN KEY (`medicine_id`) REFERENCES `medicine`(`id`)
16. local_medicine
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `medicine_id` int NOT NULL (ссылка на лекарство)
        - `production_techology` int NOT NULL (ссылка на технологию производства)
    - Ограничения:
        - FOREIGN KEY (`medicine_id`) REFERENCES `medicine`(`id`)
        - FOREIGN KEY (`production_techology`) REFERENCES `production_techonology`(`id`)
17. medicine_usage_statistics
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `medicine_id` int NOT NULL (ссылка на лекарство)
        - `quantity_used` float NOT NULL (использованное количество лекарства)

        - `usage_time` timestamp NOT NULL (дата использования лекарства)
    - Ограничения:
        - FOREIGN KEY (`medicine_id`) REFERENCES `medicine`(`id`)

18. substance_usage_statistics
    - Атрибуты:
        - `id` int PRIMARY KEY (уникальный идентификатор)
        - `substance_id` int NOT NULL (ссылка на вещество)
        - `quantity_used` float NOT NULL (использованное количество вещества)
        - `usage_time` timestamp NOT NULL (дата использования вещества)
    - Ограничения:
        - FOREIGN KEY (`substance_id`) REFERENCES `substance`(`id`)

Триггеры и процедуры

Процедуры

1. **Функция для триггера, выполняющая следующее:**
   - уменьшение количества медикаментов и ингредиентов на складах
   - проверка критического уровня медикаментов и ингредиентов
   - логирование использования медикаментов и ингредиентов

```
CREATE OR REPLACE FUNCTION decrease_medicine_and_substance_stock()
RETURNS TRIGGER AS $$
DECLARE
   rec RECORD;
   local_medicine_id INT;
BEGIN
      SELECT id INTO local_medicine_id FROM local_medicine WHERE medicine_id =
NEW.medicine_id;

   -- Уменьшение количества медикаментов на складе
   UPDATE medicine_warehouse
   SET total_amount = total_amount - NEW.quantity_used
   WHERE medicine_id = NEW.medicine_id;

   -- Проверка критического уровня медикаментов
   IF (SELECT total_amount FROM medicine_warehouse WHERE medicine_id =
NEW.medicine_id) < (SELECT critical_limit FROM medicine_warehouse WHERE
medicine_id = NEW.medicine_id) THEN
      RAISE EXCEPTION 'Critical limit reached for medicine_id %', NEW.medicine_id;
   END IF;

   -- Уменьшение количества ингредиентов на складе на основе состава медикамента
   FOR rec IN
      SELECT substance_id, required_quantity
      FROM medicine_composition
      WHERE medicine_id = local_medicine_id
   LOOP
      UPDATE substance_warehouse
      SET total_amount = total_amount - (rec.required_quantity * NEW.quantity_used)
      WHERE substance_id = rec.substance_id;

      -- Проверка критического уровня ингредиентов
      IF (SELECT total_amount FROM substance_warehouse WHERE substance_id =
rec.substance_id) < (SELECT critical_limit FROM substance_warehouse WHERE
substance_id = rec.substance_id) THEN
         RAISE EXCEPTION 'Critical limit reached for substance_id %', rec.substance_id;
      END IF;

      -- Логирование использования ингредиентов
      INSERT INTO substance_usage_statistics (substance_id, quantity_used, usage_time)
```

```
        VALUES (rec.substance_id, rec.required_quantity * NEW.quantity_used,
CURRENT_TIMESTAMP);
    END LOOP;

    -- Логирование использования медикаментов
    INSERT INTO medicine_usage_statistics (medicine_id, quantity_used, usage_time)
    VALUES (NEW.medicine_id, NEW.quantity_used, CURRENT_TIMESTAMP);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

2. **Функция для триггера, выполняющая следующее:**
   - увеличение количества медикаментов и ингредиентов на складах
   - проверка критического уровня медикаментов и ингредиентов
   - логирование использования медикаментов и ингредиентов

```
CREATE OR REPLACE FUNCTION increase_medicine_and_substance_stock() RETURNS
TRIGGER AS $$
DECLARE
    rec RECORD;
    local_medicine_id INT;
BEGIN
    -- Найти соответствующий local_medicine для medicine_id
    SELECT id INTO local_medicine_id FROM local_medicine WHERE medicine_id =
OLD.medicine_id;

    -- Возврат количества медикаментов на склад
    UPDATE medicine_warehouse
    SET total_amount = total_amount + OLD.quantity_used
    WHERE medicine_id = OLD.medicine_id;

    -- Логирование возврата медикаментов
    INSERT INTO medicine_usage_statistics (medicine_id, quantity_used, usage_time)
    VALUES (OLD.medicine_id, -OLD.quantity_used, CURRENT_TIMESTAMP);

    -- Возврат количества ингредиентов на склад на основе состава медикамента
    FOR rec IN
        SELECT substance_id, required_quantity
        FROM medicine_composition
        WHERE medicine_id = local_medicine_id
    LOOP
        UPDATE substance_warehouse
        SET total_amount = total_amount + (rec.required_quantity * OLD.quantity_used)
        WHERE substance_id = rec.substance_id;

        -- Логирование возврата ингредиентов
        INSERT INTO substance_usage_statistics (substance_id, quantity_used, usage_time)
```

```
        VALUES (rec.substance_id, -(rec.required_quantity * OLD.quantity_used),
CURRENT_TIMESTAMP);
    END LOOP;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Триггеры

1. **Триггер уменьшения количества лекарств и ингредиентов, а также логирования их использования:**

```
CREATE TRIGGER trg_decrease_medicine_and_substance_stock
AFTER INSERT ON medicine_list
FOR EACH ROW
EXECUTE FUNCTION decrease_medicine_and_substance_stock();
```

2. **Триггер уменьшения количества лекарств и ингредиентов, а также логирования их использования:**

```
CREATE TRIGGER trg_increase_medicine_and_substance_stock
BEFORE DELETE ON medicine_list
FOR EACH ROW
EXECUTE FUNCTION increase_medicine_and_substance_stock();
```

Запросы

1) **Получить сведения о покупателях, которые не пришли забрать свой заказ в назначенное им время и общее их число.**

```
SELECT customer.*, orders.*
FROM customer
   JOIN orders ON customer.id = orders.customer_id
WHERE orders.status = 'done'
 AND orders.production_date < CURRENT_DATE;

SELECT COUNT(DISTINCT customer.id)
FROM customer
   JOIN orders ON customer.id = orders.customer_id
WHERE orders.status = 'done'
 AND orders.production_date < CURRENT_DATE;
```

2) **Получить перечень и общее число покупателей, которые ждут прибытия на склад нужных им медикаментов в целом и по указанной категории медикаментов.**

```
-- В целом
SELECT DISTINCT c.id,
                c.surname,
                c.name,
                c.middle_name,
                c.phone_number,
                c.address,
                o.receipt_id,
                o.order_date,
                o.production_date,
                o.status
FROM customer c
   JOIN orders o ON c.id = o.customer_id
   JOIN receipt ON o.receipt_id = receipt.id
WHERE o.status = 'in_production';

SELECT COUNT(DISTINCT customer.id)
FROM customer
    JOIN orders ON customer.id = orders.customer_id
WHERE orders.status = 'in_production';

-- Для указанной категории медикаментов
SELECT DISTINCT c.surname,
                c.name,
```

```
                c.middle_name,
                c.phone_number,
                c.address,
                o.receipt_id,
                o.order_date,
                o.production_date,
                o.status
FROM customer c
    JOIN orders o ON c.id = o.customer_id
    JOIN receipt ON o.receipt_id = receipt.id
    JOIN medicine_list ON receipt.id =
medicine_list.receipt_id
    LEFT JOIN medicine m ON medicine_list.medicine_id =
m.id
WHERE o.status = 'in_production' AND m.type = 'pill';

SELECT COUNT(DISTINCT customer.id)
FROM customer
        JOIN orders ON customer.id = orders.customer_id
        JOIN receipt ON orders.receipt_id = receipt.id
        JOIN medicine_list ON receipt.id =
medicine_list.receipt_id
        LEFT JOIN medicine ON medicine_list.medicine_id =
medicine.id
WHERE orders.status = 'in_production' AND medicine.type =
'pill';
```

3) **Получить перечень десяти наиболее часто используемых медикаментов в целом и указанной категории медикаментов.**

```
-- В целом
SELECT m.name, SUM(mus.quantity_used) AS total_used
FROM medicine_usage_statistics mus
        JOIN medicine m ON mus.medicine_id = m.id
GROUP BY m.id
HAVING SUM(mus.quantity_used) > 0
ORDER BY total_used DESC
LIMIT 10;

-- Перечень десяти наиболее часто используемых
медикаментов для указанной категории
-- Для локальных медикаментов
SELECT m.name, m.type, SUM(mus.quantity_used) AS
total_used
FROM medicine m
```

```
        JOIN medicine_usage_statistics mus ON m.id =
mus.medicine_id
WHERE m.type = 'pill' -- нужный тип локального
медикамента
GROUP BY m.id, m.name, m.type
HAVING SUM(mus.quantity_used) > 0
ORDER BY total_used DESC
LIMIT 10;
```

4) **Получить какой объем указанных веществ использован за указанный период.**

```
SELECT substance.*,
SUM(substance_usage_statistics.quantity_used) AS
total_used
FROM substance
        JOIN substance_usage_statistics ON substance.id =
substance_usage_statistics.substance_id
WHERE substance.id IN (2, 14)
-- нужные вещества
 AND substance_usage_statistics.usage_time BETWEEN
'2024-01-01' AND '2024-08-01' -- нужный период
GROUP BY substance.id
HAVING SUM(substance_usage_statistics.quantity_used) > 0;
```

5) **Получить перечень и общее число покупателей, заказывавших определенное лекарство или определенные типы лекарств за данный период.**

```
SELECT DISTINCT c.surname,
                c.name,
                c.middle_name,
                c.phone_number,
                c.address,
                o.order_date,
                o.status
FROM customer c
        JOIN orders o ON c.id = o.customer_id
        JOIN receipt ON o.receipt_id = receipt.id
        JOIN medicine_list ON receipt.id =
medicine_list.receipt_id
WHERE medicine_list.medicine_id = 1
-- нужное лекарство
 AND o.order_date BETWEEN '2024-01-01' AND '2024-08-01';
-- нужный период
```

```sql
SELECT COUNT(DISTINCT customer.id)
FROM customer
        JOIN orders ON customer.id = orders.customer_id
        JOIN receipt ON orders.receipt_id = receipt.id
        JOIN medicine_list ON receipt.id =
medicine_list.receipt_id
WHERE medicine_list.medicine_id = 1
-- нужное лекарство
 AND orders.order_date BETWEEN '2024-01-01' AND
'2024-08-01';        -- нужный период

-- для определенных типов лекарств
SELECT DISTINCT c.surname,
                c.name,
                c.middle_name,
                c.phone_number,
                c.address,
                o.order_date,
                o.status
FROM customer c
        JOIN orders o ON c.id = o.customer_id
        JOIN receipt ON o.receipt_id = receipt.id
        JOIN medicine_list ON receipt.id =
medicine_list.receipt_id
        JOIN medicine ON medicine_list.medicine_id =
medicine.id
WHERE medicine.type = 'pill' AND o.order_date BETWEEN
'2024-01-01' AND '2024-08-01';

SELECT COUNT(DISTINCT customer.id)
FROM customer
        JOIN orders ON customer.id = orders.customer_id
        JOIN receipt ON orders.receipt_id = receipt.id
        JOIN medicine_list ON receipt.id =
medicine_list.receipt_id
        JOIN medicine ON medicine_list.medicine_id =
medicine.id
WHERE medicine.type = 'pill' AND orders.order_date
BETWEEN '2024-01-01' AND '2024-08-01';
```

6) **Получить перечень и типы лекарств, достигших своей критической нормы или закончившихся.**

```sql
SELECT m.name AS medicine_name,
      mw.total_amount,
```

```
       mw.critical_limit,
       m.type AS medicine_type
FROM medicine_warehouse mw
         JOIN medicine m ON mw.medicine_id = m.id
WHERE mw.total_amount <= mw.critical_limit;
```

7) **Получить перечень лекарств с минимальным запасом на складе в целом и по указанной категории медикаментов.**

```
-- в целом
SELECT m.name,
       mw.total_amount,
       m.type AS medicine_type
FROM medicine_warehouse mw
JOIN medicine m ON mw.medicine_id = m.id
ORDER BY mw.total_amount, m.id;

-- по указанной категории медикаментов
SELECT
   m.name,
   mw.total_amount,
   m.type AS medicine_type
FROM medicine_warehouse mw
JOIN medicine m ON mw.medicine_id = m.id
WHERE m.type = 'ointment'
ORDER BY mw.total_amount, m.id;
```

8) **Получить полный перечень и общее число заказов, находящихся в производстве.**

```
-- Полный перечень заказов находящихся в производстве
SELECT o.id,
       o.customer_id,
       o.receipt_id,
       o.order_date,
       o.production_date,
       o.status
FROM orders o
WHERE o.status = 'in_production';

-- Общее число заказов находящихся в производстве
SELECT COUNT(*) FROM orders
WHERE status = 'in_production';
```

9) **Получить полный перечень и общее число препаратов, требующихся для заказов, находящихся в производстве.**

```
SELECT m.name,
       SUM(ml.quantity_used) AS total_quantity
FROM medicine_list ml
   JOIN orders o ON ml.receipt_id = o.receipt_id
   JOIN medicine m ON ml.medicine_id = m.id
WHERE o.status = 'in_production'
GROUP BY ml.medicine_id, m.name;
```

10) **Получить все технологии приготовления лекарств указанных типов, конкретных лекарств, лекарств, находящихся в справочнике заказов в производстве.**

```
-- Технологии приготовления конкретных лекарств
SELECT
   m.id AS medicine_id,
   m.name AS medicine_name,
   pt.method_of_production,
   pt.time_to_product
FROM medicine m
   JOIN local_medicine lm ON m.id = lm.medicine_id
   JOIN production_techonology pt ON
lm.production_techology = pt.id
WHERE m.id IN (11, 12);

-- Технологии приготовления лекарств указанных типов
SELECT m.id AS medicine_id,
       m.name AS medicine_name,
       pt.method_of_production,
       pt.time_to_product
FROM medicine m
   JOIN local_medicine lm ON m.id = lm.medicine_id
   JOIN production_techonology pt ON
lm.production_techology = pt.id
WHERE
   m.type IN ('powder');

-- Технологии приготовления лекарств, находящихся в
заказах в производстве
SELECT DISTINCT m.id AS medicine_id,
                m.name AS medicine_name,
                pt.method_of_production,
                pt.time_to_product
FROM medicine m
```

```
    JOIN local_medicine lm ON m.id = lm.medicine_id
    JOIN production_techonology pt ON
lm.production_techology = pt.id
    JOIN medicine_list ml ON m.id = ml.medicine_id
    JOIN orders o ON ml.receipt_id = o.receipt_id
WHERE o.status = 'in_production';
```

11) Получить сведения о ценах на указанное лекарство в готовом виде, об объеме и ценах на все компоненты, требующиеся для этого лекарства.

```
-- Сведения о цене готового лекарства
SELECT m.id,
       m.name,
       m.price
FROM medicine m
WHERE m.id = 11;

-- Объем и цены на компоненты, требующиеся для лекарства
SELECT s.id AS id,
       s.name AS name,
       mc.required_quantity,
       s.price AS price
FROM medicine_composition mc
    JOIN substance s ON mc.substance_id = s.id
    JOIN local_medicine lm ON mc.medicine_id = lm.id
WHERE lm.medicine_id = 11;
```

12) Получить сведения о наиболее часто делающих заказы клиентах на медикаменты определенного типа, на конкретные медикаменты.

```
-- Сведения о клиентах, часто заказывающих медикаменты
определенного типа
SELECT c.surname,
       c.name,
       c.middle_name,
       COUNT(o.id) AS order_count
FROM customer c
    JOIN orders o ON c.id = o.customer_id
    JOIN medicine_list ml ON o.receipt_id = ml.receipt_id
    JOIN medicine m ON ml.medicine_id = m.id
WHERE m.type = 'pill'
GROUP BY c.id
ORDER BY order_count DESC
LIMIT 10;
```

```sql
-- Сведения о клиентах, часто заказывающих конкретные
медикаменты
SELECT c.surname,
       c.name,
       c.middle_name,
       COUNT(o.id) AS order_count
FROM customer c
   JOIN orders o ON c.id = o.customer_id
   JOIN medicine_list ml ON o.receipt_id = ml.receipt_id
WHERE ml.medicine_id = 11
GROUP BY c.id
ORDER BY order_count DESC
LIMIT 10;
```

13) **Получить сведения о конкретном лекарстве (его тип, способ приготовления, названия всех компонент, цены, его количество на складе).**

```sql
-- Сведения о конкретном лекарстве
SELECT m.name,
       m.price,
       mw.total_amount,
       m.type AS medicine_type,
       pt.method_of_production,
       pt.time_to_product
FROM medicine m
   LEFT JOIN local_medicine lm ON m.id = lm.medicine_id
   LEFT JOIN imported_medicine im ON m.id =
im.medicine_id
   LEFT JOIN production_techonology pt ON
lm.production_techology = pt.id
   LEFT JOIN medicine_warehouse mw ON m.id =
mw.medicine_id
WHERE m.id = 1;

-- Названия всех компонентов и их цены
SELECT s.name AS substance_name,
       mc.required_quantity,
       s.price AS substance_price
FROM medicine_composition mc
   JOIN substance s ON mc.substance_id = s.id
   JOIN local_medicine lm ON mc.medicine_id = lm.id
WHERE lm.medicine_id = 11;
```

Клиентское приложение

Описание

Серверная часть приложения: Go.
GUI: Go Fyne.
СУБД: PostgresPro Enterprise через JDBC