МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**КУРСОВАЯ РАБОТА Ч2**

по дисциплине

«ИНФОРМАЦИОННЫЕ СИСТЕМЫ»

***Выполнил:***

Студент группы P3318

Рамеев Тимур

Ильгизович

***Преподаватель:***

Николаев Владимир

Вячеславович

Оглавление

[Задание 3](#_Toc194307302)

[ER-модель 4](#_Toc194307303)

[Даталогическая модель 5](#_Toc194307304)

[Реализация датологической модели в PostgeSQL 6](#_Toc194307305)

[Создание отношений 6](#_Toc194307306)

[Удаление сущностей 6](#_Toc194307307)

[Генерация данных 7](#_Toc194307308)

[Функции/процедуры и триггеры 9](#_Toc194307309)

[Триггеры 9](#_Toc194307310)

[Добавление индексов 10](#_Toc194307311)

# Задание

1. Сформировать ER-модель базы данных (на основе описаний предметной области и прецедентов из предыдущего этапа). ER-модель должна:
2. включать в себя не менее 10 сущностей;
3. ​​​​​​​содержать хотя бы одно отношение вида «многие-ко-многим».
4. Согласовать ER-модель с преподавателем. На основе ER-модели построить даталогическую модель.
5. Реализовать даталогическую модель в реляционной СУБД PostgreSQL.
6. Обеспечить целостность данных при помощи средств языка DDL и триггеров.
7. Реализовать скрипты для создания, удаления базы данных, заполнения базы тестовыми данными.
8. Предложить pl/pgsql-функции и процедуры, для выполнения критически важных запросов (которые потребуются при последующей реализации прецедентов).
9. Создать индексы на основе анализа использования базы данных в контексте описанных на первом этапе прецедентов. Обосновать полезность созданных индексов для реализации представленных на первом этапе бизнес-процессов.
10. Составить отчет.

# ER-модель

Изображение выглядит как снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

# Даталогическая модель

Изображение выглядит как снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

# Реализация датологической модели в PostgeSQL

## Создание отношений

|  |
| --- |
| «СЛИШКОМ БОЛЬШОЙ СКРИПТ (НАХОДИТСЯ В init.sql) |

## Удаление сущностей

|  |
| --- |
| DO $$ DECLARE r RECORD; BEGIN  EXECUTE 'SET session\_replication\_role = replica';  FOR r IN (SELECT tablename FROM pg\_tables WHERE schemaname = 'public') LOOP  EXECUTE 'DROP TABLE IF EXISTS ' || quote\_ident(r.tablename) || ' CASCADE'; END LOOP;  FOR r IN (SELECT sequencename FROM pg\_sequences WHERE schemaname = 'public') LOOP  EXECUTE 'DROP SEQUENCE IF EXISTS ' || quote\_ident(r.sequencename) || ' CASCADE'; END LOOP;  FOR r IN (SELECT table\_name FROM information\_schema.views WHERE table\_schema = 'public') LOOP  EXECUTE 'DROP VIEW IF EXISTS ' || quote\_ident(r.table\_name) || ' CASCADE'; END LOOP;  FOR r IN (SELECT routine\_name FROM information\_schema.routines WHERE routine\_schema = 'public') LOOP  EXECUTE 'DROP FUNCTION IF EXISTS ' || quote\_ident(r.routine\_name) || ' CASCADE'; END LOOP;  FOR r IN (SELECT typname FROM pg\_type WHERE typnamespace = (SELECT oid FROM pg\_namespace WHERE nspname = 'public')) LOOP  EXECUTE 'DROP TYPE IF EXISTS ' || quote\_ident(r.typname) || ' CASCADE'; END LOOP;  EXECUTE 'SET session\_replication\_role = DEFAULT'; END $$; |

## Генерация данных

|  |
| --- |
| INSERT INTO organizations (name, address, phone, type, hours\_from, hours\_to, minutes\_from, minutes\_to) VALUES  ('Банк крови Альфа', 'ул. Ленина, 10', '123-456-7890', 'BLOOD\_BANK', 8, 18, 0, 0),  ('Банк крови Бета', 'пр. Мира, 25', '987-654-3210', 'BLOOD\_BANK', 9, 17, 30, 30),  ('Городская больница №1', 'ул. Советская, 5', '555-555-5555', 'MEDICAL\_INSTITUTION', 7, 19, 0, 0),  ('Центральный медицинский центр', 'ул. Пушкина, 15', '444-333-2222', 'MEDICAL\_INSTITUTION', 8, 20, 15, 45),  ('Банк крови Гамма', 'ул. Победы, 42', '111-222-3333', 'BLOOD\_BANK', 7, 16, 0, 30);  INSERT INTO users (email, name, surname, password, post, role, organization\_id, is\_deleted) VALUES  ('admin@blood.com', 'Иван', 'Иванов', '8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918', 'Главный администратор', 'ADMIN', NULL, FALSE),  ('banker@alpha.com', 'Алиса', 'Смирнова', '8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918', 'Директор', 'BANK\_EMPLOYEE', 1, FALSE),  ('banker1@alpha.com', 'Роман', 'Курцын', '8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918', 'Рядовой сотрудник', 'BANK\_EMPLOYEE', 1, FALSE),  ('banker@beta.com', 'Дмитрий', 'Петров', '8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918', 'Рядовой сотрудник', 'BANK\_EMPLOYEE', 2, FALSE),  ('banker@gamma.com', 'Елена', 'Козлова', '8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918', 'Рядовой сотрудник', 'BANK\_EMPLOYEE', 5, FALSE),  ('medic@hospital1.com', 'Борис', 'Сидоров', '8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918', 'Главный врач', 'MEDICAL\_EMPLOYEE', 3, FALSE),  ('medic@clinic.com', 'Мария', 'Федорова', '8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918', 'Заведующий хирургическим отделением', 'MEDICAL\_EMPLOYEE', 4, FALSE);  DO $$ DECLARE  bank RECORD;  blood\_group TEXT;  rh\_factor TEXT;  i INTEGER;  random\_volume NUMERIC;  expiration\_days INTEGER;  created\_date DATE;  expiration\_date DATE; BEGIN   FOR bank IN SELECT id FROM organizations WHERE type = 'BLOOD\_BANK' LOOP  FOR blood\_group IN SELECT *unnest*(ARRAY['1', '2', '3', '4']) LOOP  FOR rh\_factor IN SELECT unnest(ARRAY['+', '-']) LOOP  FOR i IN 1..3 LOOP  random\_volume := 0.5 + floor(random() \* 16);  expiration\_days := 30 + floor(random() \* 60);  created\_date := CURRENT\_DATE - (floor(random() \* 10) || ' days')::interval;  expiration\_date := created\_date + (expiration\_days || ' days')::interval;   INSERT INTO blood\_units (  created\_at,  expiration\_date,  volume,  blood\_bank\_id,  blood\_group,  rh\_factor  ) VALUES (  created\_date,  expiration\_date,  random\_volume,  bank.id,  blood\_group,  rh\_factor  );  END LOOP;  END LOOP;  END LOOP;  END LOOP; END $$; |

# Функции/процедуры и триггеры

## Триггеры

|  |
| --- |
| CREATE FUNCTION public.*update\_blood\_reserves*() RETURNS trigger  LANGUAGE plpgsql  AS $$ BEGIN  IF TG\_OP = 'INSERT' THEN  INSERT INTO blood\_reserves (bank\_id, blood\_group, rh\_factor, total\_quantity)  VALUES (NEW.blood\_bank\_id, NEW.blood\_group, NEW.rh\_factor, NEW.volume)  ON CONFLICT (bank\_id, blood\_group, rh\_factor)  DO UPDATE SET total\_quantity = blood\_reserves.total\_quantity + NEW.volume;  ELSIF TG\_OP = 'DELETE' THEN UPDATE blood\_reserves SET total\_quantity = *GREATEST*(0, total\_quantity - OLD.volume) WHERE bank\_id = OLD.blood\_bank\_id  AND blood\_group = OLD.blood\_group  AND rh\_factor = OLD.rh\_factor;  ELSIF TG\_OP = 'UPDATE' THEN UPDATE blood\_reserves SET total\_quantity = *GREATEST*(0, total\_quantity - OLD.volume + NEW.volume) WHERE bank\_id = OLD.blood\_bank\_id  AND blood\_group = OLD.blood\_group  AND rh\_factor = OLD.rh\_factor; END IF;  RETURN NEW; END; $$;  CREATE TRIGGER trg\_update\_inventory AFTER INSERT OR DELETE OR UPDATE ON public.blood\_units FOR EACH ROW EXECUTE FUNCTION public.update\_blood\_reserves(); |

# Добавление индексов

1. **Индекс для поиска подходящих банков, имеющих соответствующие запасы**

|  |
| --- |
| CREATE INDEX idx\_blood\_group\_rh\_total  ON public.blood\_reserves (blood\_group, rh\_factor, total\_quantity); |

1. **Составной индекс для поиска непрочитанных уведомлений для указанного пользователя**

|  |
| --- |
| CREATE INDEX idx\_notifications\_user\_read  ON public.notifications (user\_id, is\_read); |