МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра Информационные системы

Дисциплина Базы данных

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

Тема

Проектирование и разработка модели данных учета оказываемых услуг IT-компанией

Выполнил студент / Т.А. Кувшинов /

подпись инициалы, фамилия

Курс 2 Группа ПИбд-21

Направление/ специальность Программная инженерия

Руководитель ст. преподаватель

должность, ученая степень, ученое звание

Строева Юлия Владимировна

фамилия, имя, отчество

Дата сдачи:

« » 2023 г.

Дата защиты:

« » 2023 г.

Оценка:

Ульяновск

2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра Информационные системы

Дисциплина Базы данных

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

студенту ПИбд-21 Кувшинов Т.А.

группа фамилия, инициалы

Тема проекта (работы)

Проектирование и разработка модели данных учета оказываемых услуг IT-компанией

Срок сдачи законченного проекта (работы) « » 2023 г.

Исходные данные к проекту (работе)

Для хранения и использования данных учета оказываемых услуг IT-компанией требуется разработать базу данных для IT-компании

Данная база данных должна являться основой для эффективного учета, хранения и анализа информации об оказываемых IT-услугах.

Содержание пояснительной записки (перечень подлежащих разработке вопросов)

Описание поставленной задачи

Модель хранения данных представленная в ER-диаграмме

Описание структуры разработанной базы данных

Руководитель ст. преподаватель / Ю.В. Строева /

должность подпись инициалы, фамилия

« » 2023 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**/** Т.А. Кувшинов **/**

подпись инициалы, фамилия

« » 2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ  
руководителя на курсовой проект (работу)**

студента Кувшинова Тимура Александровича

фамилия, имя и отчество

Факультет ИСТ группа ПИбд-21 курс 2

Дисциплина\_\_\_\_\_\_Базы данных

Тема проекта (работы)

Проектирование и разработка модели данных учета оказываемых услуг IT-компанией

Руководитель ст. преподаватель / Ю.В. Строева /

должность, учёная степень, ученое звание подпись инициалы, фамилия

« » 2023 г.

Оглавление

[Введение 5](#_Toc121822076)

[1. Постановка задачи 7](#_Toc121822077)

[2. Модель данных в третьей нормальной форме 8](#_Toc121822078)

[3. Описание базы данных 10](#_Toc121822079)

[3.1. Описание структуры таблиц 10](#_Toc121822080)

[3.2. Описание набора данных 12](#_Toc121822081)

[3.3. Описание запросов 14](#_Toc121822082)

[3.4. Описание представлений 17](#_Toc121822083)

[3.5. Описание триггеров 18](#_Toc121822084)

[3.6. Описание хранимых процедур и функций 19](#_Toc121822085)

[Заключение 20](#_Toc121822086)

[Список литературы 21](#_Toc121822087)

[Приложение А 23](#_Toc121822088)

# Введение

В современном мире информационные технологии стали непреложной составляющей успеха множества компаний. Рост зависимости бизнеса от ИТ-инфраструктуры подчеркивает значимость эффективного учета и управления предоставляемыми IT-услугами. Однако, с развитием технологий и увеличением спектра услуг, возникают сложности в эффективной организации процессов учета и анализа предоставляемых услуг.

Для обеспечения конкурентоспособности и повышения оперативности принятия управленческих решений IT-компаниям необходимо иметь надежную и оптимизированную модель данных, способную учитывать специфику услуг, оказываемых клиентам. Обзор существующих моделей учета услуг позволяет выявить проблемы и недостатки в системах управления данными, а также предоставляет возможность для разработки более эффективной и гибкой модели.

Данная работа посвящена созданию базы данных для эффективного учета и анализа оказываемых IT-услуг. С развитием информационных технологий и расширением спектра услуг, предоставляемых IT-компаниями, возникает потребность в надежной и гибкой модели данных для эффективного управления информацией.

Целью данного исследования является разработка оптимальной базы данных для учета оказываемых услуг IT-компанией. Фирма заключает договоры на сопровождение готовых систем, а также разрабатывает свои собственные программные продукты. Работа выполняется со сторонними организациями. При заключении договора определяется перечень оказываемых услуг, сумма договора, ответственный исполнитель и сроки выполнения работ. Необходимо контролировать выполнение договоров в разрезе ответственных исполнителей и организаций.

Создание данной базы данных имеет целью оптимизировать процессы учета и анализа предоставляемых услуг IT-компанией, а также повысить уровень удовлетворенности клиентов за счет оперативного и точного управления данными

# Постановка задачи

Целью данного исследования является создание базы данных для учета и анализа оказываемых IT-услуг, что включает в себя ряд конкретных задач:

1. Анализ требований к базе данных: Изучение потребностей бизнеса и конечных пользователей для определения основных функциональных и нефункциональных требований к базе данных.
2. Проектирование структуры данных: Разработка оптимальной структуры базы данных, учитывающей разнообразие услуг, их атрибуты, связи между данными и требования к хранению информации.
3. Создание базы данных в PgSQL. Создание таблиц, заполнение данными.
4. Разработка методов взаимодействия с базой данных: Создание механизмов ввода, хранения, изменения и извлечения данных, обеспечивающих эффективную работу с информацией о предоставляемых услугах.

# Модель данных в третьей нормальной форме

Описание базы данных представлено в ER-диаграмме(рисунок 1)

Сущность контракт – сущность для хранения информации о контракте: стоимость, крайняя дата выполнения, id организации, id ответственного работника.

Работник – сущность в которой хранится информация о работнике: Имя, фамилия, должность, адрес, почта, телефон.

Организация – сущность для хранения информации о организации: Имя, почта, номер, адрес.

Услуга – сущность для хранения информации: заголовок, описание, id контракта которому эта услуга принадлежит.

Продукт – сущность для хранения информации о продуктах, с которым связан контракт: название, описание, id контракта.

Отчёт – сущность для хранения информации о продуктах, с которым связан контракт: заголовок, текст, id контракта.

Работа – сущность описывающая работу выполняемую компанией: заголовок, выполненное кол-во задач и общее кол-во задач, id контракта.

Задача – сущность для хранения информации: описания задачи, статуса задачи, работы к которой задача относится.

Статусы – список возможных статусов для задачи.

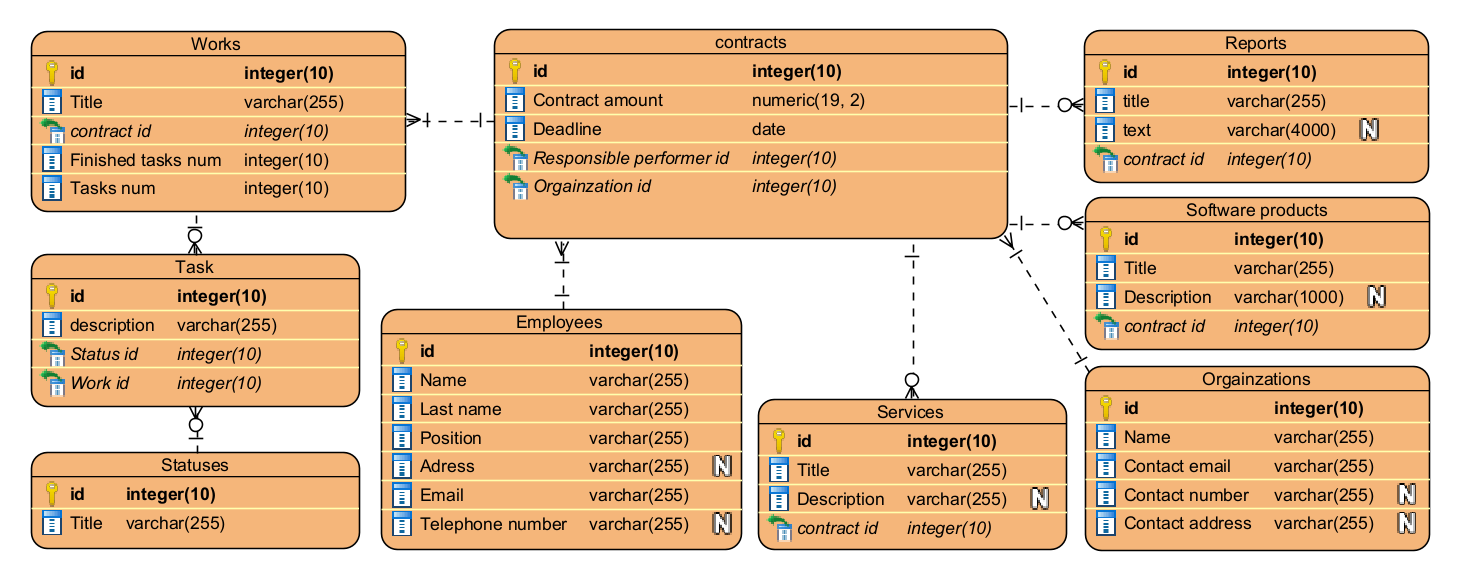


Рисунок 1 ER - диаграмма

# Описание базы данных

## Описание структуры таблиц

Все таблицы есть в диаграмме см. Рисунок 1, далее идёт краткое объяснение использованных типов данных.

Везде в качестве ключа используется id целочисленный тип.

Для всех полей Title, Name, description, email, number, address, используется varchar так как это поля хранящие строки.

Поля Finished tasks num и Tasks num таблицы Works хранят в себе соответственно количество законченных задач и привязанных задач к этой работе.

Поля таблицы contracts:

* Contract amount – numeric потому что это сумма за контракт
* Deadline – date потому что это день до которого надо выполнить контракт

Листинг 1. Создание таблиц

DROP TABLE Employees;

CREATE TABLE Employees(

ID SERIAL PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

position VARCHAR(255) NOT NULL,

adress VARCHAR(255),

email VARCHAR(255) NOT NULL,

telephone VARCHAR(20) CHECK(telephone LIKE '+7-(\_\_\_)-\_\_\_-\_\_-\_\_')

);

DROP TABLE Organizations;

CREATE TABLE Organizations(

ID SERIAL PRIMARY KEY,

name\_ VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

telephone VARCHAR(20) CHECK(telephone LIKE '+7-(\_\_\_)-\_\_\_-\_\_-\_\_'),

adress VARCHAR(255)

);

DROP TABLE Contracts;

CREATE TABLE Contracts(

id SERIAL PRIMARY key,

contract\_amount NUMERIC(19, 2) NOT NULL,

deadline DATE NOT NULL,

responsible\_performer\_id SERIAL NOT NULL,

organization\_id SERIAL NOT NULL,

FOREIGN KEY (responsible\_performer\_id) REFERENCES employees (id) ON DELETE NO ACTION,

FOREIGN KEY (organization\_id) REFERENCES organizations (id) ON DELETE NO ACTION

);

DROP TABLE Services;

CREATE TABLE Services(

id SERIAL PRIMARY key,

title VARCHAR(255) NOT NULL,

description\_ VARCHAR(255),

contract\_id SERIAL,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE Reports;

CREATE TABLE Reports(

id SERIAL PRIMARY key,

title VARCHAR(255) NOT NULL,

text\_ VARCHAR(4000),

contract\_id SERIAL,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE Sofware\_products;

CREATE TABLE Sofware\_products(

id SERIAL PRIMARY key,

title VARCHAR(255) NOT NULL,

description\_ VARCHAR(255),

contract\_id SERIAL,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE works;

CREATE TABLE Works(

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

contract\_id SERIAL NOT NULL,

finished\_tasks\_num INT NOT NULL check(finished\_tasks\_num <= tasks\_num) DEFAULT 0,

tasks\_num INT NOT NULL DEFAULT 0,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE Statuses;

CREATE TABLE Statuses(

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL

);

DROP TABLE Tasks;

CREATE TABLE Tasks(

id SERIAL PRIMARY KEY,

description\_ VARCHAR(255) NOT NULL,

status\_id SERIAL REFERENCES statuses ON DELETE NO ACTION,

work\_id SERIAL REFERENCES works ON DELETE NO ACTION

);

## Описание набора данных

В качестве набора данных я решил взять те работы, которые мне приходилось выполнять в последнюю неделю семестра:

Листинг 2. Заполнение данными

INSERT INTO Employees (first\_name, last\_name, position, adress, email, telephone)

VALUES (

'Timour',

'Kouvshinov',

'senior engineer',

'Ouliyanovsk',

'timourkot@gmail.com',

'+7-(902)-129-29-10'

);

INSERT INTO Organizations (name\_, email, telephone, adress)

VALUES(

'SuperItCompany',

'it@milo.global',

'+7-(777)-777-77-77',

'Ouliyanovsk'

);

INSERT INTO Contracts (contract\_amount, deadline, responsible\_performer\_id, organization\_id)

VALUES (

100500,

'27-12-2023',

1,

1

);

INSERT INTO Services (title, description\_, contract\_id)

VALUES (

'написать бд для учёта контрактов ит компании',

'траляля',

1

);

INSERT INTO Reports (title, text\_, contract\_id)

VALUES (

'приступил к работе',

'приступил к работе выполнения контракта 11.12.23, посидел и подумал над реализацией пока был в питере',

1

);

INSERT INTO Sofware\_products (title, description\_, contract\_id)

VALUES (

'database',

'база данных для ит компании',

1

);

INSERT INTO Works (title, contract\_id )

VALUES (

'создание бд',

1

);

INSERT INTO Statuses (title)

VALUES (

'выполнено'

),

(

'в процессе'

),

(

'заморожено'

);

INSERT INTO Tasks (description\_, status\_id, work\_id)

VALUES (

'придумать структуру',

2,

1

),

(

'придумать чем заполнить данными',

3,

1

);

INSERT INTO Organizations (name\_, email, telephone, adress)

VALUES(

'Devochka',

'vino@milo.ru',

'+7-(111)-111-11-11',

'Saint petersbourg'

);

INSERT INTO Contracts (contract\_amount, deadline, responsible\_performer\_id, organization\_id)

VALUES (

2000,

'22-12-2023',

1,

2

);

INSERT INTO Services (title, description\_, contract\_id)

VALUES (

'сделать лабораторную HSE',

'лабораторная на с++ из курса ВШЭ',

2

);

INSERT INTO Reports (title, text\_, contract\_id)

VALUES (

'приступил к работе',

'приступил к работе выполнения контракта 16.12.23',

2

),

(

'работа выполнена',

'сделал лабораторную и отправил заказчику 18.12.23',

2

);

INSERT INTO Sofware\_products (title, description\_, contract\_id)

VALUES (

'лаба',

'лабораторная на с++ из курса ВШЭ',

2

);

INSERT INTO Works (title, contract\_id )

VALUES (

'сделать лабораторную',

2

);

INSERT INTO Tasks (description\_, status\_id, work\_id)

VALUES (

'прочитать задание',

2,

2

),

(

'написать код',

2,

2

),

(

'протестировать',

2,

2

),

(

'исправить баги',

2,

2

);

## Описание запросов

Есть необходимость в запросе (Листинг 3), который будет выводить информацию о всех контрактах вместе с ответственным работником, услугами и программными продуктами для этих контрактов, также нужно в запросе выводить прогресс выполнения контракта. Прогресс выполнения контракта будет подсчитываться при помощи функции calc\_fulfilment\_status(), которая будет описана далее в пункте 3.6.

Для отслеживания работ вместе с тасками, был создан запрос, объединяющий 3 таблицы (Листинг 4).

Для отслеживания ответственным работником его работ и связанных с ними задач был написан запрос (Листинг 5).

Для отслеживания количества готовых и не готовых контрактов был создан запрос (Листинг 6).

Листинг 3 запрос

SELECT

C.id, C.amount, C.deadline, calc\_fulfilment\_status(C.id) AS fulfilment\_status,

format('%s %s', E.first\_name, E.last\_name) AS responsible\_performer\_name,

O.name\_ as organization\_name,

S.title as service\_title,

Sp.title as software\_product\_title,

W.title as work\_title

FROM Contracts C

JOIN Employees E ON C.responsible\_performer\_id = E.id

JOIN Organizations O ON C.organization\_id = O.id

JOIN Services S ON S.contract\_id = C.id

JOIN Software\_products Sp ON Sp.contract\_id = C.id

JOIN Works W ON W.contract\_id = C.id

ORDER BY C.id;

Листинг 4 запрос

SELECT

W.id, W.title as work\_title, T.description\_ task, S.title as status

FROM works W

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY W.id;

Листинг 5 запрос

SELECT

E.id, format('%s %s', E.first\_name, E.last\_name) AS employe\_name,

W.title AS work, T.description\_ AS task, S.title AS status

FROM Employees E

JOIN contracts C ON C.responsible\_performer\_id = E.id

JOIN works W ON W.contract\_id = C.id

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY E.id

Листинг 6 запрос

SELECT 'quantity' AS \_, T.\*, D.\*, total-done AS not\_done

FROM (

SELECT count(C.id) AS total

FROM contracts C

) T, (

SELECT count(C.id) AS done

FROM contracts C

WHERE calc\_fulfilment\_status(C.id) = 1

) D;

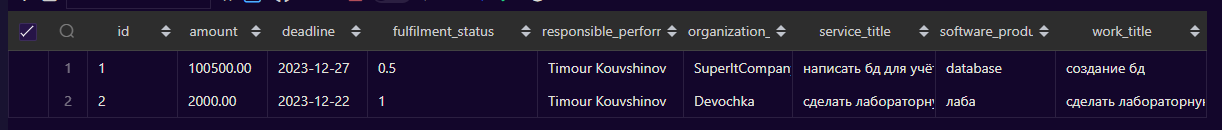


Рисунок 2 Результат запроса 1

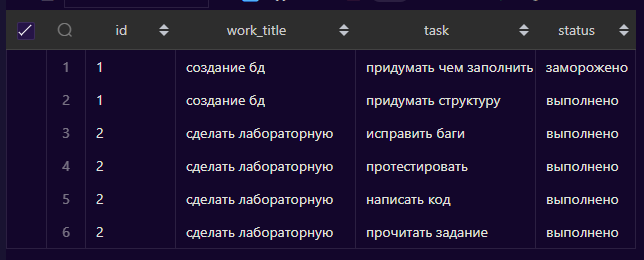


Рисунок 3 Результат запроса 2

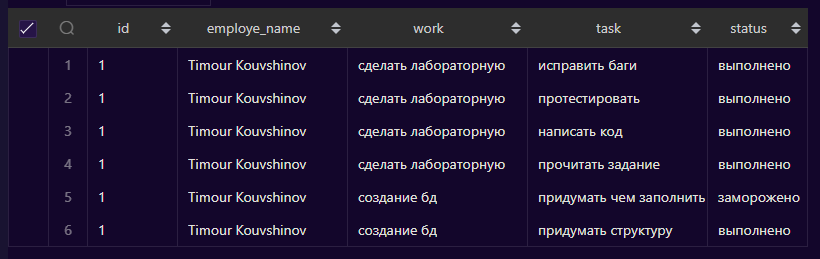
****

Рисунок 4 Результат запроса 3

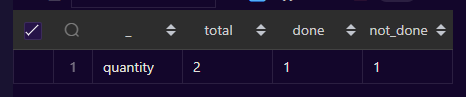
****

Рисунок 5 Результат запроса 4

## Описание представлений

Представление нужно, что бы оптимизировать запросы и скрыть сложную логику. Поэтому мои представления будут основаны на моих же запросах, ведь это позволит кэшировать их код в памяти, что ускорит работу, и скроет код запроса от пользователя.

Листинг 7. Представления

CREATE OR REPLACE VIEW contracts\_view AS

SELECT

C.id, C.amount, C.deadline, calc\_fulfilment\_status(C.id) AS fulfilment\_status,

format('%s %s', E.first\_name, E.last\_name) AS responsible\_performer\_name,

O.name\_ as organization\_name,

S.title as service\_title,

Sp.title as software\_product\_title,

W.title as work\_title

FROM Contracts C

JOIN Employees E ON C.responsible\_performer\_id = E.id

JOIN Organizations O ON C.organization\_id = O.id

JOIN Services S ON S.contract\_id = C.id

JOIN Software\_products Sp ON Sp.contract\_id = C.id

JOIN Works W ON W.contract\_id = C.id

ORDER BY C.id;

CREATE OR REPLACE VIEW works\_view AS

SELECT

W.id, W.title as work\_title, T.description\_ task, S.title as status

FROM works W

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY W.id;

CREATE OR REPLACE VIEW Empoyees\_works\_view AS

SELECT

E.id, format('%s %s', E.first\_name, E.last\_name) AS employe\_name, W.title AS work, T.description\_ AS task, S.title AS status

FROM Employees E

JOIN contracts C ON C.responsible\_performer\_id = E.id

JOIN works W ON W.contract\_id = C.id

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY E.id;

CREATE OR REPLACE VIEW Quantities AS

SELECT 'quantity' AS \_, T.\*, D.\*, total-done AS not\_done

FROM (

SELECT count(C.id) AS total

FROM contracts C

) T, (

SELECT count(C.id) AS done

FROM contracts C

WHERE calc\_fulfilment\_status(C.id) = 1

) D;

## Описание триггеров

Для базы данных я считаю оптимальным хранить кол-во выполненных и количество всего задач связанных с работой в переменных этой строчки работы: finished\_tasks\_num, tasks\_num. Поэтому каждый раз, когда я добавляю задачу надо в связанной работе tasks\_num инкрементировать, а когда задача выполняется, то есть статус становится 1, то finished\_tasks\_num прибавлять, и для удаления наоборот. Это можно сделать тригерами(Листинг \_)

Листинг 8. Тригеры

CREATE OR REPLACE FUNCTION add\_task\_triger()

RETURNS trigger AS $add\_task\_triger$

BEGIN

UPDATE works

SET tasks\_num = tasks\_num+1

WHERE id = NEW.work\_id;

RETURN NEW;

END;

$add\_task\_triger$ LANGUAGE plpgsql;

CREATE TRIGGER add\_task\_triger

AFTER INSERT ON tasks

FOR EACH ROW EXECUTE PROCEDURE add\_task\_triger();

CREATE OR REPLACE FUNCTION del\_task\_triger()

RETURNS trigger AS $del\_task\_triger$

BEGIN

IF OLD.status\_id = 1 THEN

UPDATE works

SET finished\_tasks\_num = finished\_tasks\_num-1

WHERE id = OLD.work\_id;

END IF;

UPDATE works

SET tasks\_num = tasks\_num-1

WHERE id = OLD.work\_id;

RETURN OLD;

END;

$del\_task\_triger$ LANGUAGE plpgsql;

CREATE TRIGGER del\_task\_triger

BEFORE DELETE ON tasks

FOR EACH ROW EXECUTE PROCEDURE del\_task\_triger();

CREATE OR REPLACE FUNCTION update\_task\_triger()

RETURNS trigger AS $update\_task\_triger$

BEGIN

IF NEW.status\_id = 1 and OLD.status\_id <> 1 THEN

UPDATE works

SET finished\_tasks\_num = finished\_tasks\_num+1

WHERE id = NEW.work\_id;

END IF;

IF OLD.status\_id = 1 and NEW.status\_id <> 1 THEN

UPDATE works

SET finished\_tasks\_num = finished\_tasks\_num-1

WHERE id = NEW.work\_id;

END IF;

RETURN NEW;

END;

$update\_task\_triger$ LANGUAGE plpgsql;

CREATE TRIGGER update\_task\_triger

AFTER UPDATE ON tasks

FOR EACH ROW EXECUTE PROCEDURE update\_task\_triger();

Листинг 9. Пример использования тригеров

UPDATE tasks

SET status\_id = 1

WHERE work\_id = 2;

UPDATE tasks

SET status\_id = 2

WHERE work\_id = 2;

DELETE FROM tasks

WHERE work\_id = 2;

## Описание хранимых процедур и функций

Необходимо реализовать функцию которая будет подсчитывать прогресс выполнения контракта по его Id назовём её calc\_Fulfilment\_status(Листинг 10)

Давайте создадим хранимую процедуру, которая будет выполнять все задание связанные с работой по её id.(Листинг 11)

Листинг 10. Функция

CREATE OR REPLACE FUNCTION calc\_Fulfilment\_status(id\_contract INTEGER)

RETURNS REAL AS $calc\_Fulfilment\_status$

DECLARE

finished\_tasks\_num DECIMAL;

tasks\_num DECIMAL;

BEGIN

SELECT sum(W.tasks\_num) INTO tasks\_num

FROM works W

WHERE W.contract\_id = id\_contract;

SELECT sum(W.finished\_tasks\_num) INTO finished\_tasks\_num

FROM works W

WHERE W.contract\_id = id\_contract;

IF tasks\_num = 0 THEN

RETURN 1;

END IF;

RETURN finished\_tasks\_num/tasks\_num;

END;

$calc\_Fulfilment\_status$ LANGUAGE plpgsql;

Листинг 10. Процедура

CREATE OR REPLACE PROCEDURE done\_linked(id\_work INTEGER)

LANGUAGE plpgsql AS $done\_linked$

BEGIN

UPDATE tasks

SET status\_id = 1

WHERE work\_id = id\_work;

END;

$done\_linked$

# Заключение

В ходе исследования была выполнена разработка базы данных, предназначенной для учета и анализа оказываемых IT-услуг. Была создана структура данных, соответствующая требованиям третьей нормальной формы, позволяющая эффективно хранить информацию о договорах, услугах, сотрудниках и организациях, связанных с оказанием услуг.

База данных обеспечивает возможность контроля выполнения договоров в разрезе ответственных исполнителей и организаций, а также предоставляет средства для удобного анализа информации и оперативного принятия управленческих решений.

Разработанные хранимые процедуры, функции, запросы, представления и триггеры позволяют эффективно взаимодействовать с данными и обеспечивают целостность информации.

В целом, была выполнена работа по созданию базы данных, которая станет важным инструментом для компании в учете и управлении предоставляемыми IT-услугами, способствуя оптимизации бизнес-процессов и повышению качества обслуживания клиентов.

# Список литературы

1. ISO/IEC 2382:2015 Information technology – Vocabulary [Электронный ресурс]. Режим доступа: <https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:24765:ed-1:v1:en> (дата обращения: 13.12.2022)
2. Робинсон Я., Вебер Д., Эифрем Э. Графовые базы данных. Новые возможности для работы со связанными данными. – Litres, 2022.
3. PostgresPRO. Сортировка строк [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/queries-order>
4. PostgresPRO Изменение таблиц [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/ddl-alter>
5. PostgresPRO Табличные выражения [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/queries-table-expressions>
6. PostgresPRO Строковые функции и операторы [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgrespro/9.5/functions-string>
7. PostgresPRO DELETE [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/sql-delete>
8. PostgresPRO Операторы сравнения [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.5/functions-comparison>
9. PostgresPRO Управляющие структуры [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/10/plpgsql-control-structures>
10. PostgresPRO UPDATE [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/sql-update>
11. PostgresPRO INSERT [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/sql-insert>
12. PostgresPRO Значения по умолчанию [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/ddl-default>
13. PostgresPRO Выражения значения [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/sql-expressions>
14. PostgresPRO Ограничения [Электронный ресурс]. Режим доступа: <https://postgrespro.ru/docs/postgrespro/9.5/ddl-constraints>

# Приложение А

DROP TABLE Employees;

CREATE TABLE Employees(

ID SERIAL PRIMARY KEY,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

position VARCHAR(255) NOT NULL,

adress VARCHAR(255),

email VARCHAR(255) NOT NULL,

telephone VARCHAR(20) CHECK(telephone LIKE '+7-(\_\_\_)-\_\_\_-\_\_-\_\_')

);

DROP TABLE Organizations;

CREATE TABLE Organizations(

ID SERIAL PRIMARY KEY,

name\_ VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

telephone VARCHAR(20) CHECK(telephone LIKE '+7-(\_\_\_)-\_\_\_-\_\_-\_\_'),

adress VARCHAR(255)

);

DROP TABLE Contracts;

CREATE TABLE Contracts(

id SERIAL PRIMARY key,

contract\_amount NUMERIC(19, 2) NOT NULL,

deadline DATE NOT NULL,

responsible\_performer\_id SERIAL NOT NULL,

organization\_id SERIAL NOT NULL,

FOREIGN KEY (responsible\_performer\_id) REFERENCES employees (id) ON DELETE NO ACTION,

FOREIGN KEY (organization\_id) REFERENCES organizations (id) ON DELETE NO ACTION

);

DROP TABLE Services;

CREATE TABLE Services(

id SERIAL PRIMARY key,

title VARCHAR(255) NOT NULL,

description\_ VARCHAR(255),

contract\_id SERIAL,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE Reports;

CREATE TABLE Reports(

id SERIAL PRIMARY key,

title VARCHAR(255) NOT NULL,

text\_ VARCHAR(4000),

contract\_id SERIAL,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE Sofware\_products;

CREATE TABLE Sofware\_products(

id SERIAL PRIMARY key,

title VARCHAR(255) NOT NULL,

description\_ VARCHAR(255),

contract\_id SERIAL,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE works;

CREATE TABLE Works(

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

contract\_id SERIAL NOT NULL,

finished\_tasks\_num INT NOT NULL check(finished\_tasks\_num <= tasks\_num) DEFAULT 0,

tasks\_num INT NOT NULL DEFAULT 0,

FOREIGN KEY (contract\_id) REFERENCES contracts (id) ON DELETE NO ACTION

);

DROP TABLE Statuses;

CREATE TABLE Statuses(

id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL

);

DROP TABLE Tasks;

CREATE TABLE Tasks(

id SERIAL PRIMARY KEY,

description\_ VARCHAR(255) NOT NULL,

status\_id SERIAL REFERENCES statuses ON DELETE NO ACTION,

work\_id SERIAL REFERENCES works ON DELETE NO ACTION

);

INSERT INTO Employees (first\_name, last\_name, position, adress, email, telephone)

VALUES (

'Timour',

'Kouvshinov',

'senior db engineer',

'Ouliyanovsk',

'timourkot@gmail.com',

'+7-(902)-129-29-10'

);

INSERT INTO Organizations (name\_, email, telephone, adress)

VALUES(

'SuperItCompany',

'it@milo.global',

'+7-(777)-777-77-77',

'Ouliyanovsk'

);

INSERT INTO Contracts (amount, deadline, responsible\_performer\_id, organization\_id)

VALUES (

100500,

'27-12-2023',

1,

1

);

INSERT INTO Services (title, description\_, contract\_id)

VALUES (

'написать бд для учёта контрактов ит компании',

'траляля',

1

);

INSERT INTO Reports (title, text\_, contract\_id)

VALUES (

'приступил к работе',

'приступил к работе выполнения контракта 11.12.23, посидел и подумал над реализацией пока был в питере',

1

);

INSERT INTO Software\_products (title, description\_, contract\_id)

VALUES (

'database',

'база данных для ит компании',

1

);

INSERT INTO Works (title, contract\_id )

VALUES (

'создание бд',

1

);

INSERT INTO Statuses (title)

VALUES (

'выполнено'

),

(

'в процессе'

),

(

'заморожено'

);

INSERT INTO Tasks (description\_, status\_id, work\_id)

VALUES (

'придумать структуру',

2,

1

),

(

'придумать чем заполнить данными',

3,

1

);

INSERT INTO Organizations (name\_, email, telephone, adress)

VALUES(

'Devochka',

'vino@milo.ru',

'+7-(111)-111-11-11',

'Saint petersbourg'

);

INSERT INTO Contracts (amount, deadline, responsible\_performer\_id, organization\_id)

VALUES (

2000,

'22-12-2023',

1,

2

);

INSERT INTO Services (title, description\_, contract\_id)

VALUES (

'сделать лабораторную HSE',

'лабораторная на с++ из курса ВШЭ',

2

);

INSERT INTO Reports (title, text\_, contract\_id)

VALUES (

'приступил к работе',

'приступил к работе выполнения контракта 16.12.23',

2

),

(

'работа выполнена',

'сделал лабораторную и отправил заказчику 18.12.23',

2

);

INSERT INTO Software\_products (title, description\_, contract\_id)

VALUES (

'лаба',

'лабораторная на с++ из курса ВШЭ',

2

);

INSERT INTO Works (title, contract\_id )

VALUES (

'сделать лабораторную',

2

);

INSERT INTO Tasks (description\_, status\_id, work\_id)

VALUES (

'прочитать задание',

2,

2

),

(

'написать код',

2,

2

),

(

'протестировать',

2,

2

),

(

'исправить баги',

2,

2

);

-- Active: 1698417979383@@127.0.0.1@5432@Kursach

SELECT

C.id, C.amount, C.deadline, calc\_fulfilment\_status(C.id) AS fulfilment\_status,

format('%s %s', E.first\_name, E.last\_name) AS responsible\_performer\_name,

O.name\_ as organization\_name,

S.title as service\_title,

Sp.title as software\_product\_title,

W.title as work\_title

FROM Contracts C

JOIN Employees E ON C.responsible\_performer\_id = E.id

JOIN Organizations O ON C.organization\_id = O.id

JOIN Services S ON S.contract\_id = C.id

JOIN Software\_products Sp ON Sp.contract\_id = C.id

JOIN Works W ON W.contract\_id = C.id

ORDER BY C.id;

SELECT

W.id, W.title as work\_title, T.description\_ task, S.title as status

FROM works W

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY W.id;

SELECT

E.id, format('%s %s', E.first\_name, E.last\_name) AS employe\_name, W.title AS work, T.description\_ AS task, S.title AS status

FROM Employees E

JOIN contracts C ON C.responsible\_performer\_id = E.id

JOIN works W ON W.contract\_id = C.id

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY E.id;

SELECT 'quantity' AS \_, T.\*, D.\*, total-done AS not\_done

FROM (

SELECT count(C.id) AS total

FROM contracts C

) T, (

SELECT count(C.id) AS done

FROM contracts C

WHERE calc\_fulfilment\_status(C.id) = 1

) D;

-- Active: 1698417979383@@127.0.0.1@5432@Kursach

CREATE OR REPLACE FUNCTION add\_task\_triger()

RETURNS trigger AS $add\_task\_triger$

BEGIN

UPDATE works

SET tasks\_num = tasks\_num+1

WHERE id = NEW.work\_id;

RETURN NEW;

END;

$add\_task\_triger$ LANGUAGE plpgsql;

CREATE TRIGGER add\_task\_triger

AFTER INSERT ON tasks

FOR EACH ROW EXECUTE PROCEDURE add\_task\_triger();

CREATE OR REPLACE FUNCTION del\_task\_triger()

RETURNS trigger AS $del\_task\_triger$

BEGIN

IF OLD.status\_id = 1 THEN

UPDATE works

SET finished\_tasks\_num = finished\_tasks\_num-1

WHERE id = OLD.work\_id;

END IF;

UPDATE works

SET tasks\_num = tasks\_num-1

WHERE id = OLD.work\_id;

RETURN OLD;

END;

$del\_task\_triger$ LANGUAGE plpgsql;

CREATE TRIGGER del\_task\_triger

BEFORE DELETE ON tasks

FOR EACH ROW EXECUTE PROCEDURE del\_task\_triger();

CREATE OR REPLACE FUNCTION update\_task\_triger()

RETURNS trigger AS $update\_task\_triger$

BEGIN

IF NEW.status\_id = 1 and OLD.status\_id <> 1 THEN

UPDATE works

SET finished\_tasks\_num = finished\_tasks\_num+1

WHERE id = NEW.work\_id;

END IF;

IF OLD.status\_id = 1 and NEW.status\_id <> 1 THEN

UPDATE works

SET finished\_tasks\_num = finished\_tasks\_num-1

WHERE id = NEW.work\_id;

END IF;

RETURN NEW;

END;

$update\_task\_triger$ LANGUAGE plpgsql;

CREATE TRIGGER update\_task\_triger

AFTER UPDATE ON tasks

FOR EACH ROW EXECUTE PROCEDURE update\_task\_triger();

UPDATE tasks

SET status\_id = 1

WHERE work\_id = 2;

UPDATE tasks

SET status\_id = 2

WHERE work\_id = 2;

DELETE FROM tasks

WHERE work\_id = 2;

UPDATE tasks

SET status\_id = 1

WHERE id = 1;

-- Active: 1698417979383@@127.0.0.1@5432@Kursach

CREATE OR REPLACE VIEW contracts\_view AS

SELECT

C.id, C.amount, C.deadline, calc\_fulfilment\_status(C.id) AS fulfilment\_status,

format('%s %s', E.first\_name, E.last\_name) AS responsible\_performer\_name,

O.name\_ as organization\_name,

S.title as service\_title,

Sp.title as software\_product\_title,

W.title as work\_title

FROM Contracts C

JOIN Employees E ON C.responsible\_performer\_id = E.id

JOIN Organizations O ON C.organization\_id = O.id

JOIN Services S ON S.contract\_id = C.id

JOIN Software\_products Sp ON Sp.contract\_id = C.id

JOIN Works W ON W.contract\_id = C.id

ORDER BY C.id;

CREATE OR REPLACE VIEW works\_view AS

SELECT

W.id, W.title as work\_title, T.description\_ task, S.title as status

FROM works W

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY W.id;

CREATE OR REPLACE VIEW Empoyees\_works\_view AS

SELECT

E.id, format('%s %s', E.first\_name, E.last\_name) AS employe\_name, W.title AS work, T.description\_ AS task, S.title AS status

FROM Employees E

JOIN contracts C ON C.responsible\_performer\_id = E.id

JOIN works W ON W.contract\_id = C.id

JOIN tasks T ON T.work\_id = W.id

JOIN statuses S ON T.status\_id = S.id

ORDER BY E.id;

CREATE OR REPLACE VIEW Quantities AS

SELECT 'quantity' AS \_, T.\*, D.\*, total-done AS not\_done

FROM (

SELECT count(C.id) AS total

FROM contracts C

) T, (

SELECT count(C.id) AS done

FROM contracts C

WHERE calc\_fulfilment\_status(C.id) = 1

) D;

-- Active: 1698417979383@@127.0.0.1@5432@Kursach

CREATE OR REPLACE FUNCTION calc\_Fulfilment\_status(id\_contract INTEGER)

RETURNS REAL AS $calc\_Fulfilment\_status$

DECLARE

finished\_tasks\_num DECIMAL;

tasks\_num DECIMAL;

BEGIN

SELECT sum(W.tasks\_num) INTO tasks\_num

FROM works W

WHERE W.contract\_id = id\_contract;

SELECT sum(W.finished\_tasks\_num) INTO finished\_tasks\_num

FROM works W

WHERE W.contract\_id = id\_contract;

IF tasks\_num = 0 THEN

RETURN 1;

END IF;

RETURN finished\_tasks\_num/tasks\_num;

END;

$calc\_Fulfilment\_status$ LANGUAGE plpgsql;

SELECT calc\_Fulfilment\_status(1);

CREATE OR REPLACE PROCEDURE done\_linked(id\_work INTEGER)

LANGUAGE plpgsql AS $done\_linked$

BEGIN

UPDATE tasks

SET status\_id = 1

WHERE work\_id = id\_work;

END;

$done\_linked$

CALL done\_linked(1);