**­СОДЕРЖАНИЕ**

ВВЕДЕНИЕ…………………………………………………………………………..4

1 ПРОЕКТИРОВАНИЕ ……………………………………………………………...5

1.1 Анализ предметной области……………………………………………………..5

1.2 Выделение сущностей…………………………………………………………...5

1.3 Проектирование инфологической модели базы данных……………………….6

1.4 Проектирование даталогической модели базы данных……………………….7

2 РЕАЛИЗАЦИЯ……………………………………………………………………12

2.1 Написание скрипта создания базы данных……………………………………12

2.2 Заполнение базы данных……………………………………………………….15

2.3 Интерфейс……………………………………………………………………….25

2.4 Запросы………………………………………………………………………….26

ЗАКЛЮЧЕНИЕ……………………………………………………………………..33

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ……………………………………34

**ВВЕДЕНИЕ**

Создаваемая система предназначена для ведения учета работников на предприятии. Разрабатываемая база данных позволяет получить информацию о структурных подразделениях предприятия, должностях и работниках. Помимо этого, в системе хранятся данные об окладах и надбавках каждого из сотрудников, а также открытые вакансии на должности, кандидаты на них и их отклики.

Актуальность разработки данной системы состоит в том, что она позволяет эффективно получать, обновлять и поддерживать в актуальном и согласованном состояниях множество связных между собой данных о сотрудниках, что повысит эффективность рабочих процессов на предприятии.

Для реализации используется реляционная база данных PostgerSQL 14.4, консольная утилита psql, язык программирования Ruby 3.0.2, язык программирования Go 1.19.5 вместе с драйвером pgx базы данных PostgerSQL.

**1 ПРОЕКТИРОВАНИЕ**

**1.1 Анализ предметной области**

Предметной областью проектируемой базы данных является учет сотрудников предприятия. Для функционирования предприятия база данных должна хранить данные о структурных подразделениях предприятия, сотрудниках, их окладах и надбавках, должностях, а также информацию о соискателях на них.

**1.2 Выделение сущностей**

Для проектирования базы данных необходимо выделить элементы предметной области. Можно выделить следующие сущности:

1. Структурное подразделение

Структурное подразделение обладает уникальным кодом и названием. У каждого подразделения есть связанная с ним позиция штатного расписания.

1. Позиция штатного расписания

В позиции штатного расписания должно быть указано структурное подразделение, должность и ее оклад, а также надбавки, уникальные для каждой позиции. Кроме того, в позиции штатного расписания указывается количество штатных единиц на каждую позицию, можно ли совмещать ее и примечания к ней.

1. Должность

Должность содержит название, оклад и обязанности. На предприятии не может быть двух одинаковых должностей с одинаковым окладом.

1. Надбавка

Так как каждая позиция штатного расписания может иметь не только несколько надбавок, но и не иметь их вовсе, надбавка была выделена как отдельная сущность. В надбавке указывается причина и денежный размер надбавки.

1. Кандидат на должность

Кандидат на должность представляет собой отдельную сущность. Необходимо хранить имя кандидата, его дату рождения и квалификацию.

1. Вакансия

Каждая вакансия привязана к позиции штатного расписания, причем число вакансий не должно превышать количество штатных единиц. Вакансия содержит текстовое описание, дату открытия и закрытия.

1. Работник

Работник связан с кандидатом один к одному. Это сделано с целью избежать дублирования данных. К работнику добавляется дата найма и дата увольнения.

1. Отклик на вакансию

Отклик на вакансию связывает кандидата на вакансию и вакансию. Помимо этого, он хранит дату отклика.

1. Список исполняющих обязанностей

Один работник может занимать несколько должностей. Для этого необходимо выделить в отдельную сущность список исполняющих обязанностей, который содержит ссылки на работника и позицию штатного расписания.

Главной сущностью в системе является позиция штатного расписания.

**1.3 Проектирование инфологической модели базы данных**

Исходя из сущностей и их свойств, определенных в пункте 1.2, возможно построить инфологическую модель базы данных. Это необходимо для дальнейшего корректного проектирования базы данных. Инфологическая модель базы данных представлена на рисунке 1.

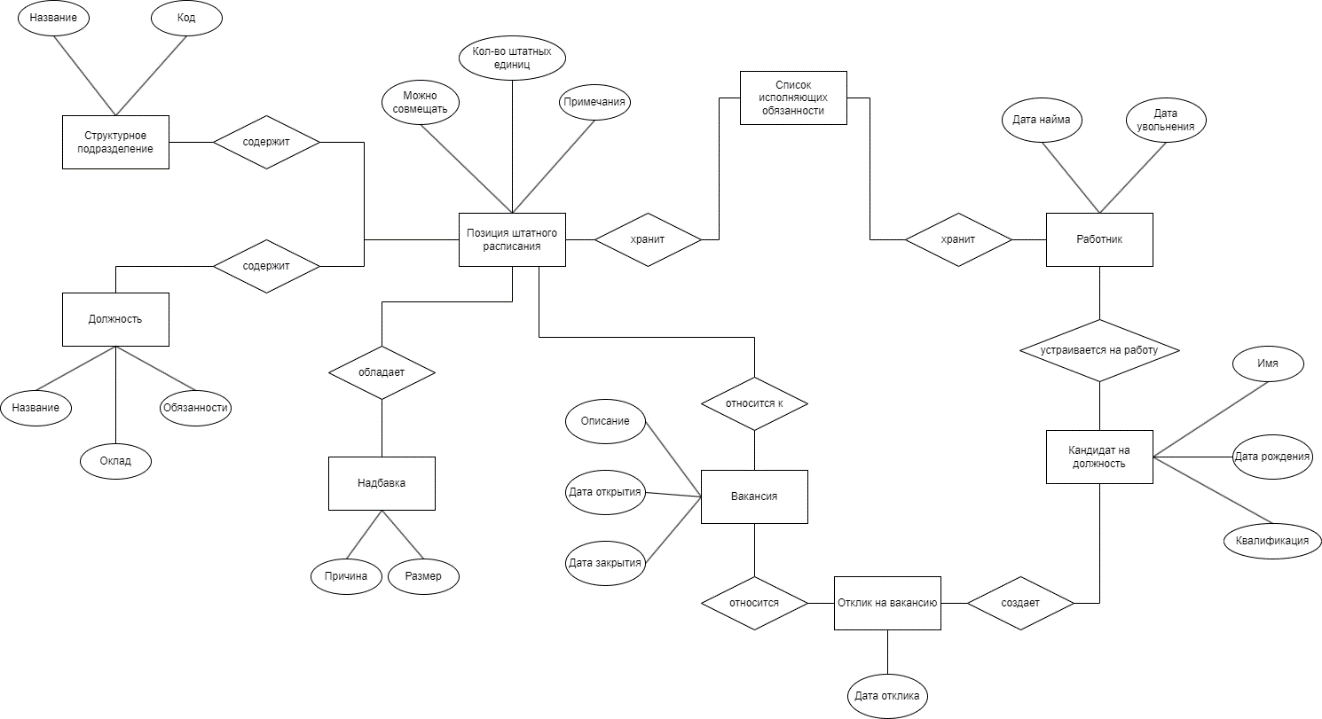


Рисунок 1 – Инфологическая модель базы данных

При разработке инфологической модели были выделены основные процессы, через которые происходит взаимодействие сущностей.

**1.4 Проектирование датологической модели базы данных**

На основании инфологической модели можно построить даталогическую модель, необходимую для написания корректного скрипта создания базы данных и входящих в нее таблиц. Даталогическая модель представлена на рисунке 2.

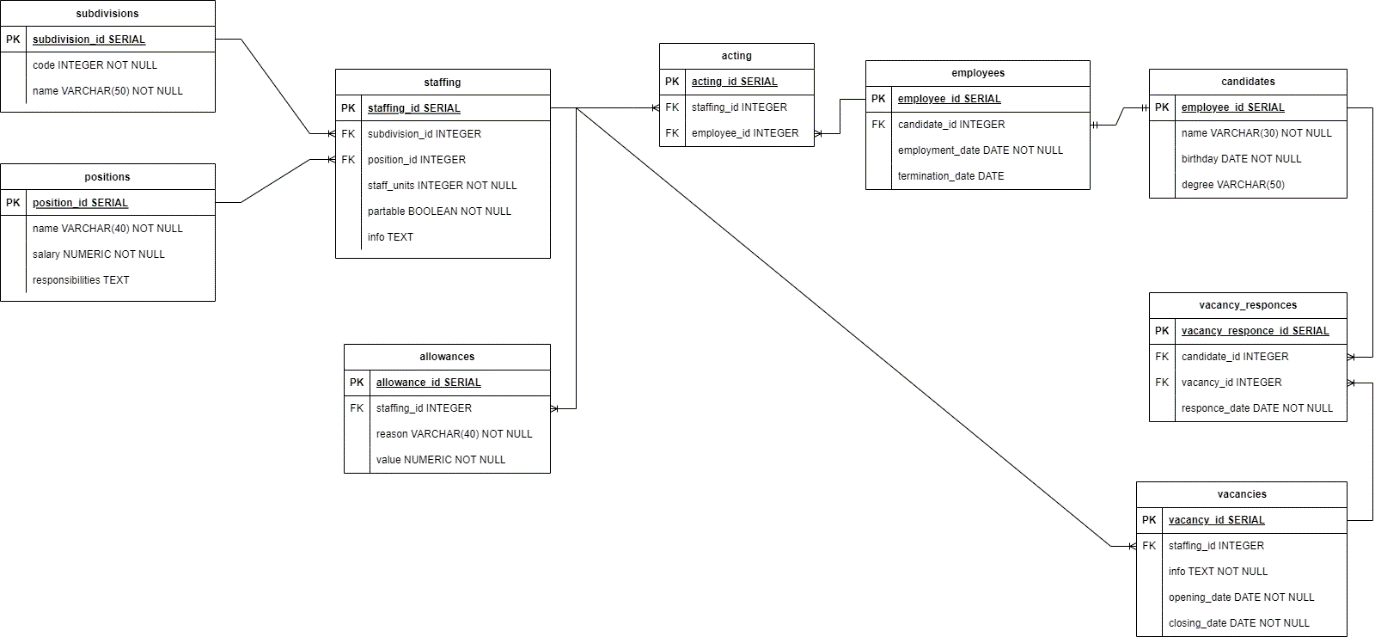


Рисунок 2 – Даталогическая модель базы данных

Для того, чтобы пояснить назначение полей в каждой из таблиц, а связи между ними были более очевидными, ниже приведено описание каждой из таблиц и её полей.

Таблица 1 – Описание полей таблицы subdivisions

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| subdivision\_id | SERIAL | Суррогатный первичный ключ |
| code | INTEGER | Код структурного подразделения |
| name | VARCHAR(40) | Название структурного подразделения |

Таблица 2 – Описание полей таблицы positions

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| position\_id | SERIAL | Суррогатный первичный ключ |
| name | VARCHAR(40) | Наименование должности |
| salary | NUMERIC | Размер оклада |
| responsibilities | TEXT | Должностные обязанности |

Таблица 3 – Описание полей таблицы staffing

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| staffing\_id | SERIAL | Суррогатный первичный ключ |
| subdivision\_id | INTEGER | Ссылка на структурное подразделение |
| position\_id | INTEGER | Ссылка на должность |
| staff\_units | INTEGER | Кол-во штатных единиц |
| partable | BOOLEAN | Можно ли совмещать позицию |
| info | TEXT | Примечания |

Таблица 4 – Описание полей таблицы allowances

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| allowance\_id | SERIAL | Суррогатный первичный ключ |
| staffing\_id | INTEGER | Ссылка на позицию штатного расписания |
| reason | VARCHAR(40) | Причина надбавки |
| value | NUMERIC | Величина надбавки |

Таблица 5 – Описание полей таблицы candidates

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| candidate\_id | SERIAL | Суррогатный первичный ключ |
| name | VARCHAR(30) | ФИО кандидата |
| birthday | DATE | Дата рождения |
| degree | VARCHAR(50) | Квалификация |

Таблица 6 – Описание полей таблицы employees

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| employee\_id | SERIAL | Суррогатный первичный ключ |
| candidate\_id | INTEGER | Ссылка на кандидата |
| employment\_date | DATE | Дата найма |
| termination\_date | DATE | Дата увольнения |

Таблица 7 – Описание полей таблицы vacancies

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| vacancy\_id | SERIAL | Суррогатный первичный ключ |
| staffing\_id | INTEGER | Ссылка на позицию штатного расписания |
| info | TEXT | Описание вакансии |
| opening\_date | DATE | Дата открытия |
| closing\_date | DATE | Дата закрытия |

Таблица 8 – Описание полей таблицы vacancy\_responces

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| vacancy\_responce\_id | SERIAL | Суррогатный первичный ключ |
| candidate\_id | INTEGER | Ссылка на кандидата |
| vacancy\_id | INTEGER | Ссылка на вакансию |
| responce\_date | DATE | Дата отклика |

Таблица 9 – Описание полей таблицы acting

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Значение** |
| acting\_id | SERIAL | Суррогатный первичный ключ |
| staffing\_id | INTEGER | Ссылка на позицию штатного расписания |
| employee\_id | INTEGER | Ссылка на работника |

**2 РЕАЛИЗАЦИЯ**

**2.1 Написание скрипта создания базы данных**

Напишем SQL-скрипт создания таблиц, описанных на этапе проектирования. Помимо этого, создадим роль директора производства, который будет иметь доступ к чтению и изменению данных таблиц, а также два триггера для поддержания системы в согласованном состоянии.

Листинг 1 – Создание объектов базы данных

DROP DATABASE IF EXISTS staffing;

CREATE DATABASE staffing;

\connect staffing;

CREATE TABLE subdivisions (

subdivision\_id SERIAL PRIMARY KEY,

code INTEGER NOT NULL UNIQUE,

name VARCHAR(40) NOT NULL UNIQUE

);

CREATE TABLE positions (

position\_id SERIAL PRIMARY KEY,

name VARCHAR(40) NOT NULL UNIQUE,

salary NUMERIC NOT NULL CONSTRAINT non\_zero\_pos\_salary CHECK(salary > 0) NOT NULL,

responsibilities TEXT NOT NULL

);

CREATE TABLE staffing (

staffing\_id SERIAL PRIMARY KEY,

subdivision\_id INTEGER REFERENCES subdivisions,

position\_id INTEGER REFERENCES positions,

staff\_units INTEGER CONSTRAINT non\_zero\_pos\_staff\_units CHECK(staff\_units > 0) NOT NULL,

partable BOOLEAN NOT NULL,

info TEXT

);

CREATE TABLE allowances (

allowance\_id SERIAL PRIMARY KEY,

staffing\_id INTEGER REFERENCES staffing,

reason VARCHAR(40) NOT NULL UNIQUE,

value NUMERIC NOT NULL

);

CREATE TABLE candidates (

candidate\_id SERIAL PRIMARY KEY,

name VARCHAR(30) NOT NULL UNIQUE,

birthday DATE NOT NULL,

degree VARCHAR(50)

);

CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

candidate\_id INTEGER REFERENCES candidates,

employment\_date DATE NOT NULL,

termination\_date DATE,

CONSTRAINT termination\_date\_greater CHECK(employment\_date < termination\_date)

);

CREATE TABLE vacancies(

vacancy\_id SERIAL PRIMARY KEY,

staffing\_id INTEGER REFERENCES staffing,

info TEXT NOT NULL,

opening\_date DATE NOT NULL,

closing\_date DATE NOT NULL,

CONSTRAINT closing\_date\_greater CHECK(opening\_date < closing\_date)

);

CREATE TABLE vacancy\_responces(

vacancy\_responce\_id SERIAL PRIMARY KEY,

candidate\_id INTEGER REFERENCES candidates,

vacancy\_id INTEGER REFERENCES vacancies,

responce\_date DATE DEFAULT CURRENT\_DATE

);

CREATE TABLE acting (

acting\_id SERIAL PRIMARY KEY,

staffing\_id INTEGER REFERENCES staffing,

employee\_id INTEGER REFERENCES employees

);

DROP USER IF EXISTS headman;

CREATE USER headman;

ALTER ROLE headman PASSWORD 'headman';

GRANT pg\_read\_all\_data TO headman;

GRANT pg\_write\_all\_data TO headman;

CREATE FUNCTION check\_vacancies\_count() RETURNS TRIGGER AS $$

DECLARE

staff\_units integer = (SELECT staff\_units FROM staffing WHERE staffing\_id = NEW.staffing\_id);

vacancies\_count integer = (SELECT COUNT(\*) FROM vacancies WHERE staffing\_id = NEW.staffing\_id);

BEGIN

if (vacancies\_count + 1 > staff\_units) then

RAISE EXCEPTION 'Exceeded the number of staff units (%)', staff\_units;

end if;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_vacancies\_count BEFORE INSERT OR UPDATE ON vacancies

FOR EACH ROW EXECUTE PROCEDURE check\_vacancies\_count();

CREATE FUNCTION check\_partable\_position() RETURNS TRIGGER AS $$

DECLARE

is\_partable boolean = (SELECT partable FROM staffing WHERE staffing\_id = NEW.staffing\_id);

now\_act integer = (SELECT COUNT(\*) FROM acting WHERE staffing\_id = NEW.staffing\_id);

BEGIN

if (is\_partable = false AND now\_act > 0) then

RAISE EXCEPTION 'Position is not partable';

end if;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_partable\_position BEFORE INSERT OR UPDATE ON acting

FOR EACH ROW EXECUTE PROCEDURE check\_vacancies\_count();

**2.2 Заполнение базы данных**

Для заполнения базы данных были написана программа на языке программирования Ruby с использованием Faker, которая создает SQL-скрипты, содержащие необходимые инструкции для заполнения базы данных. Каждый SQL-скрипт содержит 100.000 инструкций вставки данных. Для исполнения данных SQL-скриптов был написан PowerShell-скрипт, который вызывает исполнение всех файлов консольной утилитой psql в нужном порядке. Исходные коды скриптов на Ruby и PowerShell представлены в листингах 2 и 3.

Листинг 2 – Скрипт создания SQL-скриптов для заполнения

require 'faker'

ratio = 10\_000

subdivisions\_count = ratio

positions\_count = ratio \* 100

staffing\_count = ratio \* 1000

allowances\_count = ratio / 10

candidates\_count = ratio \* 3000

employees\_count = ratio \* 1000

vacancies\_count = ratio

vacancy\_responces\_count = ratio \* 6000

acting\_count = ratio \* 10

path\_prefix = './fill scritps/'

# Filling subdivisions

subdivisions\_files = 1

f = File.open(path\_prefix + "subdivisions #{subdivisions\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..subdivisions\_count do

if i % 100\_000 == 0

subdivisions\_files += 1

f = File.open(path\_prefix + "subdivisions #{subdivisions\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

f.write(

"INSERT INTO subdivisions(code, name)

VALUES (

#{i},

\'#{Faker::Alphanumeric.alpha(number: 10)}\'

);\n"

)

end

f.write("\n")

puts "Subdivisions are done"

# Filling positions

positions\_files = 1

f = File.open(path\_prefix + "positions #{positions\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..positions\_count do

if i % 100\_000 == 0

positions\_files += 1

f = File.open(path\_prefix + "positions #{positions\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

f.write(

"INSERT INTO positions(name, salary, responsibilities)

VALUES (

\'#{Faker::Alphanumeric.alpha(number: 10)}\',

#{Faker::Number.between(from: 10\_000, to: 1\_000\_000)},

\'#{Faker::Lorem.sentence}\'

);\n"

)

end

f.write("\n")

puts "Positions are done"

# Filling staffing

staffing\_files = 1

f = File.open(path\_prefix + "staffing #{staffing\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..staffing\_count do

if i % 100\_000 == 0

staffing\_files += 1

f = File.open(path\_prefix + "staffing #{staffing\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

f.write(

"INSERT INTO staffing(subdivision\_id, position\_id, staff\_units, partable, info)

VALUES (

#{Faker::Number.between(from: 1, to: subdivisions\_count)},

#{Faker::Number.between(from: 1, to: positions\_count)},

#{Faker::Number.between(from: 1, to: 10)},

#{Faker::Boolean.boolean(true\_ratio: 0.7)},

\'#{Faker::Lorem.sentence}\'

);\n"

)

end

f.write("\n")

puts "Staffing are done"

# Filling allowances

allowances\_files = 1

f = File.open(path\_prefix + "allowances #{allowances\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..allowances\_count do

if i % 100\_000 == 0

allowances\_files += 1

f = File.open(path\_prefix + "allowances #{allowances\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

f.write(

"INSERT INTO allowances(staffing\_id, reason, value)

VALUES (

#{Faker::Number.between(from: 1, to: staffing\_count)},

\'#{Faker::Alphanumeric.alpha(number: 10)}\',

#{Faker::Number.between(from: 1000, to: 10\_000)}

);\n"

)

end

f.write("\n")

puts "Allowances are done"

# Filling candidates

candidates\_files = 1

f = File.open(path\_prefix + "candidates #{candidates\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..candidates\_count do

if i % 100\_000 == 0

candidates\_files += 1

f = File.open(path\_prefix + "candidates #{candidates\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

f.write(

"INSERT INTO candidates(name, birthday, degree)

VALUES (

\'#{Faker::Name.name.sub(/\'/, "\\\'")}\',

\'#{Faker::Date.between(from: '1950-01-01', to: '2022-01-01')}\',

\'#{Faker::Educator.degree}\'

);\n"

)

end

f.write("\n")

puts "Candidates are done"

# Filling employees

employees\_files = 1

f = File.open(path\_prefix + "employees #{employees\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..employees\_count do

if i % 100\_000 == 0

employees\_files += 1

f = File.open(path\_prefix + "employees #{employees\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

if rand(100) % 3 == 0

date = Faker::Date.between(from: '2001-01-01', to: '2023-01-01')

f.write(

"INSERT INTO employees(candidate\_id, employment\_date, termination\_date)

VALUES (

#{Faker::Number.between(from: 1, to: candidates\_count)},

\'#{date}\',

\'#{Faker::Date.between(from: String(date), to: '2023-01-01')}\'

);\n"

)

else

f.write(

"INSERT INTO employees(candidate\_id, employment\_date, termination\_date)

VALUES (

#{Faker::Number.between(from: 1, to: candidates\_count)},

\'#{Faker::Date.between(from: '2001-01-01', to: '2023-01-01')}\',

NULL

);\n"

)

end

end

f.write("\n")

puts "Employees are done"

# Filling vacancies

vacancies\_files = 1

f = File.open(path\_prefix + "vacancies #{vacancies\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..vacancies\_count do

if i % 100\_000 == 0

vacancies\_files += 1

f = File.open(path\_prefix + "vacancies #{vacancies\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

date = Faker::Date.between(from: '2001-01-01', to: '2023-01-01')

f.write(

"INSERT INTO vacancies(staffing\_id, info, opening\_date, closing\_date)

VALUES (

#{Faker::Number.between(from: 1, to: staffing\_count)},

\'#{Faker::Lorem.sentence}\',

\'#{date}\',

\'#{Faker::Date.between(from: String(date), to: '2023-01-01')}\'

);\n"

)

end

f.write("\n")

puts "Vacancies are done"

# Filling vacancy responces

vacancy\_responces\_files = 1

f = File.open(path\_prefix + "vacancy responces #{vacancy\_responces\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..vacancy\_responces\_count do

if i % 100\_000 == 0

vacancy\_responces\_files += 1

f = File.open(path\_prefix + "vacancy responces #{vacancy\_responces\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

f.write(

"INSERT INTO vacancy\_responces(candidate\_id, vacancy\_id, responce\_date)

VALUES (

#{Faker::Number.between(from: 1, to: candidates\_count)},

#{Faker::Number.between(from: 1, to: vacancies\_count)},

\'#{Faker::Date.between(from: '2001-01-01', to: '2023-01-01')}\'

);\n"

)

end

f.write("\n")

puts "Vacancy responces are done"

# Filling acting

acting\_files = 1

f = File.open(path\_prefix + "acting #{acting\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

for i in 1..vacancy\_responces\_count do

if i % 100\_000 == 0

acting\_files += 1

f = File.open(path\_prefix + "acting #{acting\_files}.sql", 'w')

f.write("\\connect staffing;\n\n")

end

f.write(

"INSERT INTO acting(staffing\_id, employee\_id)

VALUES (

#{Faker::Number.between(from: 1, to: staffing\_count)},

#{Faker::Number.between(from: 1, to: employees\_count)}

);\n"

)

end

f.write("\n")

puts "Acting are done"

Листинг 3 – PowerShell скрипт

$subdivisions = Get-ChildItem -Path 'fill scritps' -Filter 'subdivisions \*.sql'

foreach ($subdiv in $subdivisions) {

psql -U postgres -f $subdiv.FullName

}

$positions = Get-ChildItem -Path 'fill scritps' -Filter 'positions \*.sql'

foreach ($pos in $positions) {

psql -U postgres -f $pos.FullName

}

$staffing = Get-ChildItem -Path 'fill scritps' -Filter 'staffing \*.sql'

foreach ($staff in $staffing) {

psql -U postgres -f $staff.FullName

}

$allowances = Get-ChildItem -Path 'fill scritps' -Filter 'allowances \*.sql'

foreach ($allw in $allowances) {

psql -U postgres -f $allw.FullName

}

$candidates = Get-ChildItem -Path 'fill scritps' -Filter 'candidates \*.sql'

foreach ($cand in $candidates) {

psql -U postgres -f $cand.FullName

}

$employees = Get-ChildItem -Path 'fill scritps' -Filter 'employees \*.sql'

foreach ($emp in $employees) {

psql -U postgres -f $emp.FullName

}

$vacancies = Get-ChildItem -Path 'fill scritps' -Filter 'vacancies \*.sql'

foreach ($vac in $vacancies) {

psql -U postgres -f $vac.FullName

}

$vacancy\_responces = Get-ChildItem -Path 'fill scritps' -Filter 'vacancy responces \*.sql'

foreach ($vac\_resp in $vacancy\_responces) {

psql -U postgres -f $vac\_resp.FullName

}

$acting = Get-ChildItem -Path 'fill scritps' -Filter 'acting \*.sql'

foreach ($act in $acting) {

psql -U postgres -f $act.FullName

}

**2.3 Интерфейс**

Для реализации интерфейса к полученной базе данных был выбран язык программирования Go. С его помощью было создано простое веб-приложение, внешний вид которого можно видеть на рисунке 3.

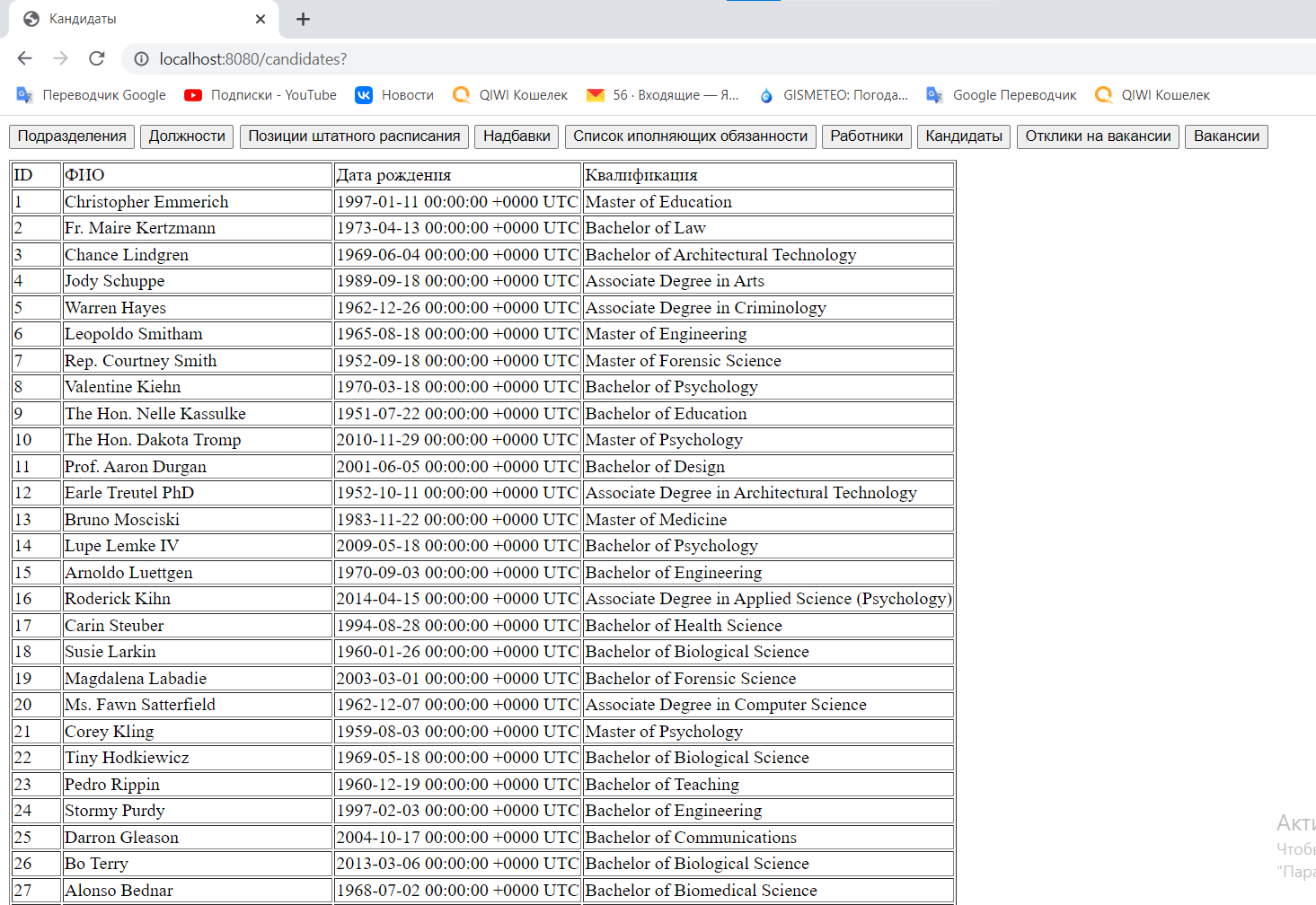


Рисунок 3 – Внешний вид интерфейса

**2.4 Запросы**

Для получения различной статистики, необходимой в рамках предметной области, был разработан ряд SQL-запросов, приведенных в листингах ниже. Целесообразность формирования именно такой статистики обусловлена спецификой предметной области. Ниже каждого из запросов расположен рисунок с примером результата.

Листинг 4 – Получение штатного расписания в формате документа

SELECT subdiv.name,

subdiv.code,

pos.name,

staffing.staff\_units,

pos.salary,

(SELECT SUM(allw.value)

FROM allowances AS allw

WHERE allw.staffing\_id = staffing.staffing\_id

GROUP BY allw.allowance\_id) AS allowances,

staffing.info

FROM subdivisions AS subdiv JOIN staffing USING (subdivision\_id)

JOIN positions AS pos USING (position\_id);

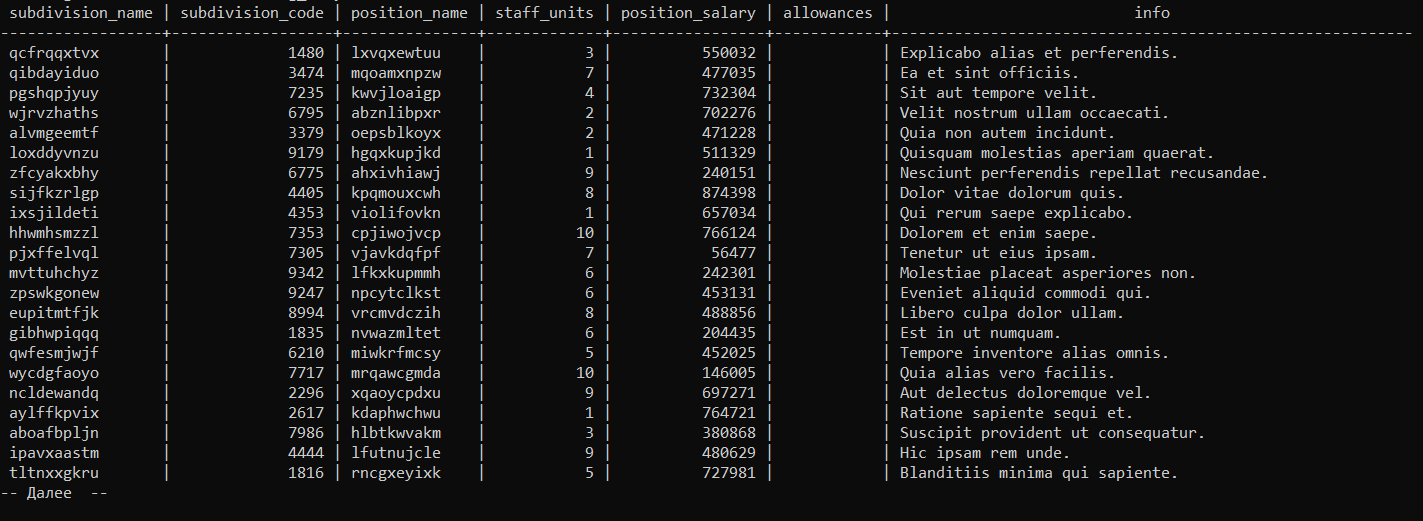


Рисунок 4 – Результат выполнения запроса

Листинг 5 – Неактуальные отклики на вакансии

SELECT cand.name AS candidate\_name, vac\_resp.responce\_date, vac.closing\_date, pos.name AS position\_name

FROM vacancies AS vac JOIN vacancy\_responces AS vac\_resp USING (vacancy\_id)

JOIN staffing USING (staffing\_id)

JOIN positions AS pos USING (position\_id)

JOIN candidates AS cand USING (candidate\_id)

WHERE vac\_resp.responce\_date > vac.closing\_date

ORDER BY candidate\_name;

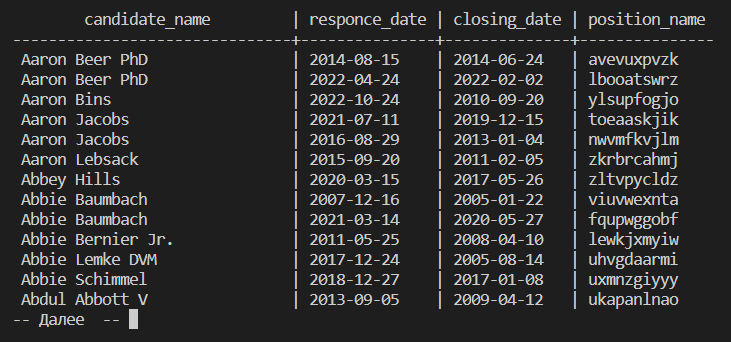


Рисунок 5 – Результат выполнения запроса

Листинг 6 – Средняя зарплата по отделам

SELECT subdiv.name,

(SELECT AVG(pos.salary + COALESCE((SELECT SUM(allw.value)

FROM allowances AS allw

WHERE allw.staffing\_id = staffing.staffing\_id

GROUP BY allw.allowance\_id),

0)

)

FROM staffing JOIN positions AS pos USING (position\_id)

WHERE staffing.subdivision\_id = subdiv.subdivision\_id

GROUP BY subdiv.subdivision\_id) AS avg\_salary

FROM subdivisions AS subdiv

ORDER BY avg\_salary;

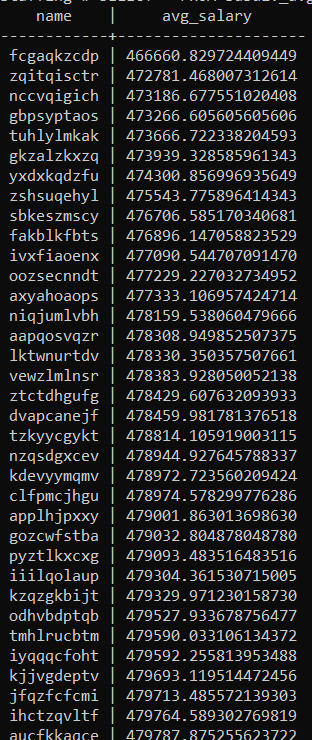


Рисунок 6 – Результат выполнения запроса

Листинг 7 – Вакансии по числу откликов

SELECT pos.name, pos.salary, COUNT(vac\_resp.vacancy\_responce\_id)

FROM vacancies AS vac JOIN vacancy\_responces AS vac\_resp USING (vacancy\_id)

JOIN staffing USING (staffing\_id)

JOIN positions AS pos USING (position\_id)

GROUP BY pos.name, pos.salary;

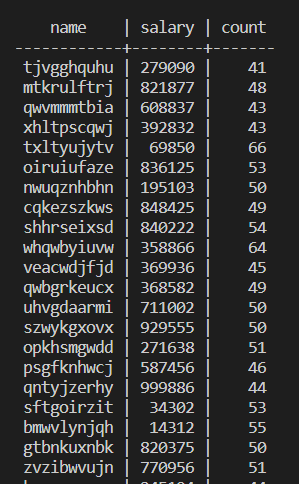


Рисунок 7 – Результат выполнения запроса

Листинг 8 – На скольких должностях работает каждый работник

SELECT cand.name, COUNT(acting.acting\_id) AS postions\_count

FROM staffing JOIN acting USING (staffing\_id)

JOIN employees AS emp USING (employee\_id)

JOIN candidates AS cand USING (candidate\_id)

WHERE emp.termination\_date IS NULL OR emp.termination\_date > CURRENT\_DATE

GROUP BY cand.name;

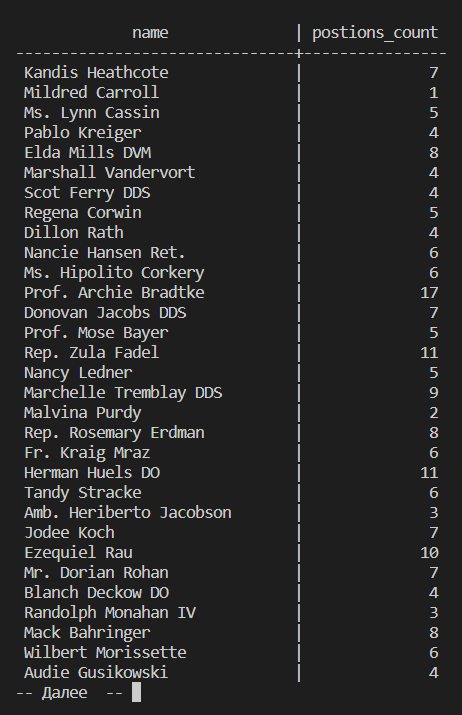


Рисунок 8 – Результат выполнения запроса

Листинг 9 – Уволенные работники и должности, на которых они работали

SELECT cand.name AS candidate\_name,

emp.employment\_date,

emp.termination\_date,

pos.name AS position\_name

FROM employees AS emp JOIN candidates AS cand USING (candidate\_id)

JOIN acting USING (employee\_id)

JOIN staffing USING (staffing\_id)

JOIN positions AS pos USING (position\_id)

WHERE emp.termination\_date IS NOT NULL AND emp.termination\_date < CURRENT\_DATE;

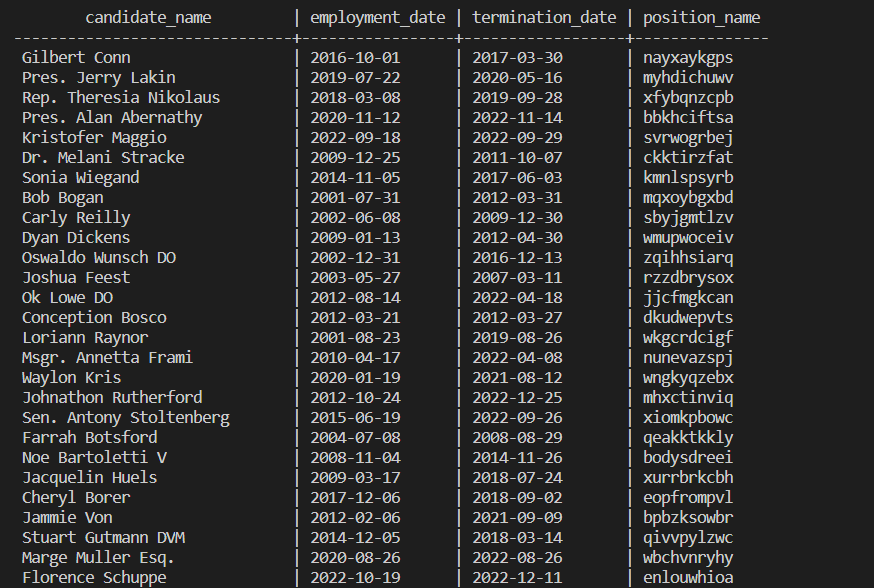


Рисунок 9 – Результат выполнения запроса

**ЗАКЛЮЧЕНИЕ**

Во время выполнения данной работы, выполнено следующее:

* Проведен анализ предметной области, выделены основные сущности и процессы;
* Спроектирована база данных, разработаны инфологическая даталогическая модель базы данных;
* Написан скрипт создания таблиц базы данных, роли директора и триггеров, обеспечивающих согласованность данных;
* Разработан пользовательский интерфейс;
* Написаны SQL-запросы для получения выборок данных.

В результате курсовой работы, была реализована система учета работников на предприятии. Система с помощью интерфейса предоставляет доступ к необходимой информации, а также позволяет создавать и изменять данные, что повышает эффективность предприятия, на котором данная система будет внедрена.

**СПИСОК ЛИТЕРАТУРЫ**

1. Скворцова М. А., Замула М. И. Методические указания по выполнению курсовой работы по курсу «Базы данных» [Электронный ресурс] URL: [https://e-learning.bmstu.ru/iu6/pluginfile.php/19147/mod\_resource/content/1/%D0%9C%D0%A0\_%D0%9A%D0%A0\_%D0%91%D0%94\_2%20%D0%BA%D1%83%D1%80%D1%81.pdf](https://e-learning.bmstu.ru/iu6/pluginfile.php/19147/mod_resource/content/1/МР_КР_БД_2 курс.pdf)

2. Документация модуль psycopg для Python [Электронный ресурс] URL: <https://github.com/faker-ruby/faker>

3. А.В. Бобин, С. А. Булгаков Правила оформления отчетов к лабораторным и курсовым работам [Электронный ресурс] URL: <https://publications.hse.ru/pubs/share/direct/227003831.pdf>

4. Документация Microsoft.PowerShell.Management [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/powershell/module/microsoft.powershell.management/?view=powershell-7.3>

5. Документация к PostgreSQL 9.6.24 [Электронный ресурс] URL: <https://postgrespro.ru/docs/postgresql/9.6/index>

6. Формы первичных учетных документов [Электронный ресурс] URL: <https://www.consultant.ru/document/cons_doc_LAW_32449/1d769d91ace06658134e81c925cc070bc3e63147/>

7. Руководство по веб-программированию языку Go [Электронный ресурс] URL: <https://metanit.com/go/web/>