Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

**КУРСОВОЙ ПРОЕКТ**

По курсу **Проектирование и администрирование баз данных**

по направлению 09.03.01 − Информатика и вычислительная техника

Образовательная программа (профиль) «Системная и программная инженерия»

ТЕМА

**«Проектирование и разработка базы данных «Учет внутриофисных расходов» в СУБД PostgreSQL»**

Студент: \_\_\_\_\_\_\_\_\_ Никитин Олег Евгеньевич, 221-329

*ФИО, группа*

Проверил: \_\_\_\_\_\_\_\_ к.т.н., доцент Евдошенко О.И.

Москва, 2023

Федеральное государственное автономное

образовательное учреждение высшего образования

**«Московский политехнический университет»**

**Кафедра:** инфокогнитивных технологий

**Направление подготовки:** 09.03.01 Информатика и вычислительная техника

**Дисциплина:** Проектирование и администрирование баз данных

**Тема:** Проектирование и разработка базы данных «Учет внутриофисных расходов» в СУБД PostgreSQL

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

1. Необходимо выбрать и проанализировать предметную область.
2. Определить сущности (>5), атрибуты (>3), связи между сущностями. Привести отношения к 3 НФ.
3. Разработать базу данных средствами СУБД на основе спроектированной модели. Установить связи для поддержки ссылочной целостности.
4. Разработать SQL-запросы (DML) (не менее 20) различной степени сложности (агрегатные функции, группировка (GROUP BY, HAVING), подзапросы, CASE, оконные функции)
5. Разработать функции для расчетов (3 функции, используя язык SQL и plpgSQL)
6. Разработать триггеры (5 триггеров)

**Исполнитель**: студент гр. 221-329 Никитин О.Е.

подпись

**Руководитель**: к.т.н., доцент кафедры ИКТ Евдошенко О.И.

подпись

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc138845948)

[Актуальность проблемы 4](#_Toc138845949)

[Цель работы: 4](#_Toc138845950)

[Задачи работы: 4](#_Toc138845951)

[1 Описание предметной области 6](#_Toc138845952)

[2 Проектирование и разработка данных 7](#_Toc138845953)

[2.1 Концептуальное проектирование базы данных 7](#_Toc138845954)

[2.2 Логическое проектирование базы данных 8](#_Toc138845955)

[2.3 Выбор и описание СУБД 9](#_Toc138845956)

[2.4 Физическое проектирование базы данных 10](#_Toc138845957)

[3 Использование базы данных 21](#_Toc138845958)

[3.1 Создание DataSet 21](#_Toc138845959)

[3.2 Добавление данных в базу 22](#_Toc138845960)

[3.3 Запросы к базе данных 26](#_Toc138845961)

[3.4 Тестирование функций и триггеров 35](#_Toc138845962)

[Заключение 41](#_Toc138845963)

[Список использованных источников 42](#_Toc138845964)

# ВВЕДЕНИЕ

## Актуальность проблемы

Современные компании все больше придают значение модернизации структуры своей базы данных. Это становится неотъемлемой частью их деятельности по следующим причинам:

1) быстрое развитие технологий требует обновления и адаптации баз данных к новым возможностям и растущему объему информации;

2) изменения в бизнес-процессах подразумевают необходимость согласованной структуры баз данных, чтобы эффективно управлять информационными потоками;

3) приспособление к изменяющемуся законодательству требует своевременной модернизации баз данных для соответствия обязательствам по хранению и обработке данных пользователей;

Учет всех этих факторов позволяет компаниям быть гибкими, повышать эффективность и обеспечивать защиту интересов клиентов. В итоге, модернизация баз данных становится важной составляющей успешной бизнес-стратегии.

Курсовая работа выполнена в рамках учебного предмета «Проектирование и администрирование баз данных» и нацелена на закрепление навыков по разработке с использованием СУБД PostgreSQL.

## Цель работы:

Систематизация и закрепление полученных теоретических и практических умений по разработке баз данных с использованием СУБД PostgreSQL.

## Задачи работы:

1. изучить литературу по проектированию и разработке базы данных в СУБД PostgreSQL;
2. получить практический опыт разработки базы данных в СУБД PostgreSQL.

# 1 описание предметной области

Вы работаете в бухгалтерии частной фирмы. Сотрудники фирмы имеют возможность осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Вашей задачей является отслеживание внутриофисных расходов. Фирма состоит из отделов. Каждый отдел имеет название. В каждом отделе работает определенное количество сотрудников. Сотрудники могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел.

Нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Не использованные в текущем месяце деньги могут быть использованы позже.

При изучении предметной области был выявлен 1 бизнес-процесс Подробнее с ним можно ознакомиться в «Таблице 1 – Список основных бизнес-процессов».

Таблица 1 - Список основных бизнес-процессов

|  |  |
| --- | --- |
| Номер бизнес-процесса | Наименование бизнес процесса |
| 1 | Оформление сотрудником документа о расходах |

# 2 проектирование и разработка данных

## 2.1 Концептуальное проектирование базы данных

При анализе предметной области были выявлены следующие сущности: «сотрудник», «отдел», «документ о расходе».

Сущность «сотрудник» содержит следующие атрибуты:

1. фамилия;
2. имя;
3. отчество.

Сущность «отдел» содержит следующие атрибуты:

1. название.

Сущность «документ о расходе» содержит следующие атрибуты:

1. вид расхода;
2. дата;
3. сумма;
4. ФИО Сотрудника.

Связи между сущностями отображены в «Рисунке 1 – Концептуальная модель данных».

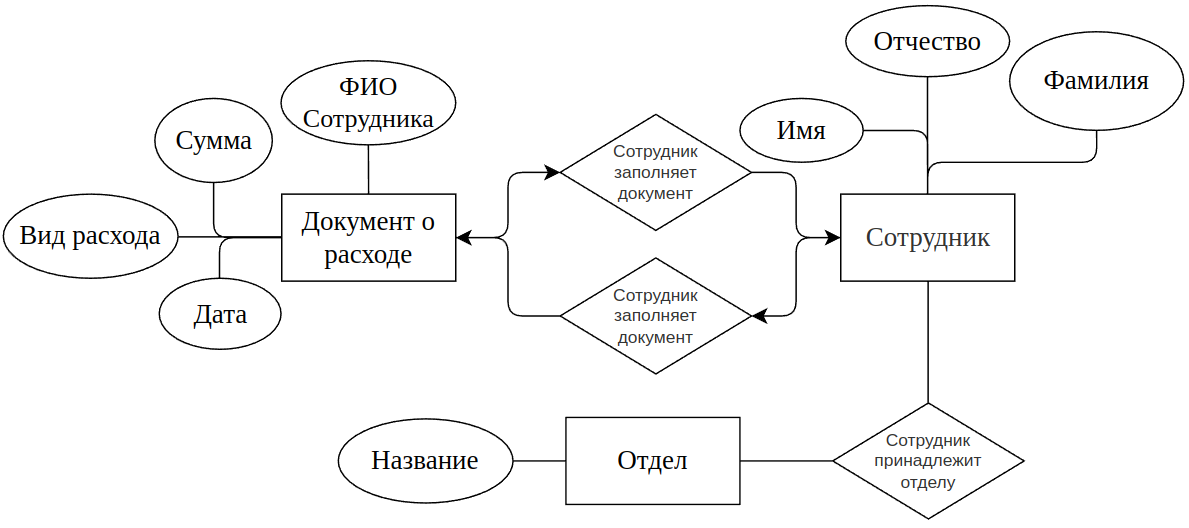


Рисунок 1 – Концептуальная модель данных

Структуры сущностей приведены в последующих таблицах: «Таблица 2 – Структура сущности сотрудник», «Таблица 3 – Структура сущности отдел», «Таблица 4 – Структура сущности документ о расходе».

Таблица 2 – структура сущности сотрудник

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Фамилия | Текстовый | 32 |  |
| Имя | Текстовый | 32 |  |
| Отчество | Текстовый | 32 |  |

Таблица 3 – структура сущности отдел

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Название | Текстовый | 32 |  |

Таблица 4 – структура сущности документ о расходе

|  |  |  |  |
| --- | --- | --- | --- |
| Название параметра | Тип данных | Размер | Диапазон значений |
| Вид расхода | Текстовый | 32 |  |
| Дата | Дата |  |  |
| Сумма | Числовой |  | От 0 |
| ФИО Сотрудника | Текстовый | 92 |  |

## 2.2 Логическое проектирование базы данных

Для выявленных сущностей была создана логическая схема базы данных (Рисунок 2 – Логическая схема базы данных).

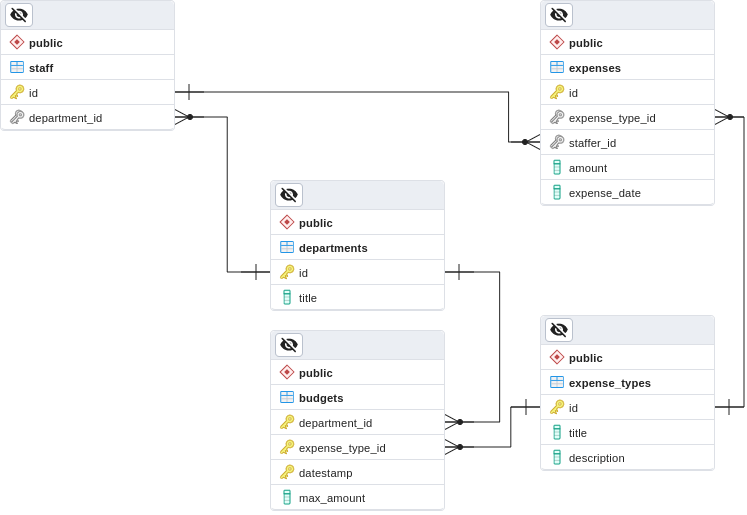


Рисунок 2 – Логическая модель данных

## 2.3 Выбор и описание СУБД

Для реализации курсовой работы по базам данных была выбрана СУБД PostgreSQL.

PostgreSQL является СУБД с открытым исходным кодом, что позволяет как можно глубже изучить внутреннее устройство СУБД. PostgreSQL также обладает широким набором функций, обеспечивающих гибкость при разработке баз данных, поддерживает различные типы данных. Надежность и стабильность PostgreSQL также являются важными преимуществами. СУБД активно поддерживается и разрабатывается сообществом разработчиков, что гарантирует безопасность и исправление ошибок. Встроенные механизмы резервного копирования, восстановления и системы журналирования обеспечивают сохранность данных и доступность базы данных. Кроме того, PostgreSQL обладает высокой производительностью и масштабируемостью. Он способен эффективно обрабатывать большие объемы данных и поддерживает параллельную обработку запросов, оптимизацию запросов и кэширование. Наконец, PostgreSQL имеет широкое сообщество разработчиков и пользователей, а также обширную документацию. Это облегчает изучение и работу с PostgreSQL, а также предоставляет возможность получить поддержку и советы от опытных пользователей. Все эти причины делают PostgreSQL отличным выбором для реализации курсовой работы.

## 2.4 Физическое проектирование базы данных

Используя функционал PostgreSQL была создана физическая схема базы данных (Рисунок 3 – Физическая модель данных).

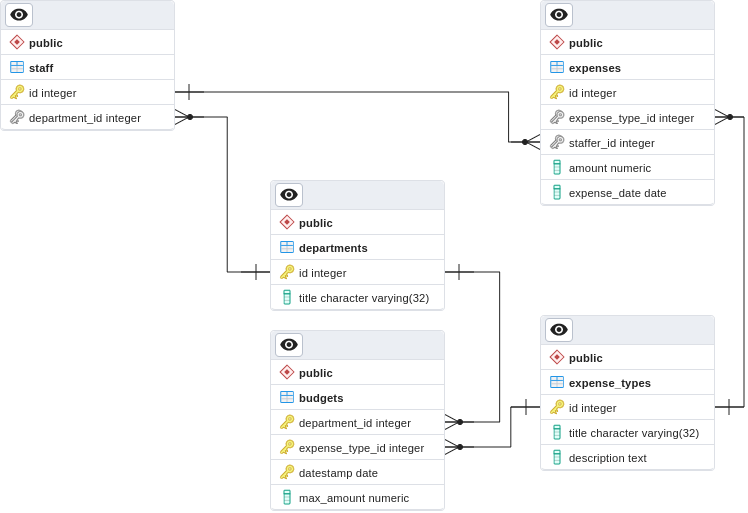


Рисунок 3 – Физическая модель данных

Далее модель была реализована при помощи PostgreSQL. Для создания таблиц и связей между ними были использованы SQL-запросы, которые указаны в «Листинге 1 – Инициализация базы данных».

Листинг 1 – Инициализация базы данных

------------

-- Отделы --

------------

DROP TABLE IF EXISTS departments CASCADE;

CREATE TABLE IF NOT EXISTS departments

(

id SERIAL PRIMARY KEY

,title VARCHAR(32) NOT NULL

);

----------------

-- Сотрудники --

----------------

DROP TABLE IF EXISTS staff CASCADE;

CREATE TABLE IF NOT EXISTS staff

(

id SERIAL PRIMARY KEY

,department\_id INT REFERENCES departments (id) NOT NULL

);

-------------------

-- Виды расходов --

-------------------

DROP TABLE IF EXISTS expense\_types CASCADE;

CREATE TABLE IF NOT EXISTS expense\_types

(

id SERIAL PRIMARY KEY

,title VARCHAR(32) NOT NULL

,description TEXT

);

-------------

-- Бюджеты --

-------------

DROP TABLE IF EXISTS budgets CASCADE;

CREATE TABLE IF NOT EXISTS budgets

(

department\_id INT REFERENCES departments (id) NOT NULL

,expense\_type\_id INT REFERENCES expense\_types (id) NOT NULL

,datestamp DATE NOT NULL

,max\_amount DECIMAL NOT NULL CHECK (max\_amount >= 0)

,PRIMARY KEY (department\_id, expense\_type\_id, datestamp)

);

-------------

-- Расходы --

-------------

DROP TABLE IF EXISTS expenses CASCADE;

CREATE TABLE IF NOT EXISTS expenses

(

id SERIAL PRIMARY KEY

,expense\_type\_id INT REFERENCES expense\_types (id) NOT NULL

,staffer\_id INT REFERENCES staff (id) NOT NULL

,amount DECIMAL NOT NULL CHECK (amount >= 0)

,expense\_date DATE NOT NULL

);

Для работы с базой данных были разработаны следующие функции и триггеры:

1. функция department\_expenses(depart\_id INT), которая используется для вывода таблицы всех расходов конкретного отдела по годам, месяцам и типам расходов. Код функции и ее инициализация указаны в «Листинге 2 – Функция department\_expenses»;

Листинг 2 – Функция department\_expenses

CREATE OR REPLACE FUNCTION department\_expenses(depart\_id INT)

RETURNS TABLE (year DECIMAL, month DECIMAL, expense\_type\_id INT, expenses DECIMAL)

AS $$

BEGIN

IF NOT EXISTS (SELECT 1 FROM departments WHERE id = depart\_id) THEN

RAISE EXCEPTION 'Department does not exist';

END IF;

RETURN QUERY

SELECT

EXTRACT(YEAR FROM expenses.expense\_date) AS year

,EXTRACT(MONTH FROM expenses.expense\_date) AS month

,expenses.expense\_type\_id

,COALESCE(SUM(expenses.amount), 0) AS expenses

FROM expenses, staff

WHERE

expenses.staffer\_id = staff.id

AND staff.department\_id = depart\_id

GROUP BY year, month, expenses.expense\_type\_id

ORDER BY year, month, expenses.expense\_type\_id;

RETURN;

END;

$$ LANGUAGE plpgsql;

1. функция department\_expenses\_from\_to(depart\_id INT, start\_date DATE, end\_date DATE), которая используется для вывода суммы всех расходов конкретного отдела от даты до даты. Код функции и ее инициализация указаны в «Листинге 3 – Функция department\_expenses\_from\_to»;

Листинг 3 – Функция department\_expenses\_from\_to

CREATE OR REPLACE FUNCTION department\_expenses\_from\_to(depart\_id INT, start\_date DATE, end\_date DATE)

RETURNS DECIMAL

AS $$

DECLARE

total\_expenses DECIMAL;

BEGIN

IF NOT EXISTS (SELECT 1 FROM departments WHERE id = depart\_id) THEN

RAISE EXCEPTION 'Department does not exist';

END IF;

SELECT COALESCE(SUM(expenses.amount), 0) INTO total\_expenses

FROM expenses, staff

WHERE

staff.department\_id = depart\_id

AND expenses.staffer\_id = staff.id

AND expenses.expense\_date >= start\_date

AND expenses.expense\_date <= end\_date;

RETURN total\_expenses;

END;

$$ LANGUAGE plpgsql;

1. функция staff\_expenses(), которая используется для вывода таблицы всех расходов сотрудников. Код функции и ее инициализация указаны в «Листинге 4 – Функция staff\_expenses»;

Листинг 4 – Функция staff\_expenses

DROP FUNCTION IF EXISTS staff\_expenses();

CREATE OR REPLACE FUNCTION staff\_expenses()

RETURNS TABLE

(

staffer\_id INT

,expense\_date DATE

,amount DECIMAL

)

AS $$

BEGIN

RETURN QUERY

SELECT

staff.id

,expenses.expense\_date

,expenses.amount

FROM expenses, staff

WHERE expenses.staffer\_id = staff.id;

END;

$$ LANGUAGE plpgsql;

1. функция department\_remaining\_amount(depart\_id INT, type\_id INT), которая используется для вывода оставшейся не потраченной суммы отдела в конкретной категории. Код функции и ее инициализация указаны в «Листинге 5 – Функция department\_remaining\_amount»;

Листинг 5 – Функция department\_remaining\_amount

CREATE OR REPLACE FUNCTION department\_remaining\_amount(depart\_id INT, type\_id INT)

RETURNS DECIMAL

AS $$

DECLARE

remaining\_amount DECIMAL;

BEGIN

IF NOT EXISTS (SELECT 1 FROM departments WHERE id = depart\_id) THEN

RAISE EXCEPTION 'Department does not exist';

END IF;

IF NOT EXISTS (SELECT 1 FROM expense\_types WHERE id = type\_id) THEN

RAISE EXCEPTION 'Expense type does not exist';

END IF;

SELECT

COALESCE((

SELECT SUM(max\_amount)

FROM budgets

WHERE

department\_id = depart\_id

AND expense\_type\_id = type\_id

AND datestamp <= NOW()

), 0) -

COALESCE((

SELECT SUM(amount)

FROM expenses, staff

WHERE

staff.department\_id = depart\_id

AND expenses.staffer\_id = staff.id

AND expense\_type\_id = type\_id

AND expense\_date <= NOW()

), 0) INTO remaining\_amount;

RETURN remaining\_amount;

END;

$$ LANGUAGE plpgsql;

1. прощедура pay(staffer\_id INT, expense\_type VARCHAR(32), amount DECIMAL), которая используется для вывода оставшейся не потраченной суммы отдела в конкретной категории. Код функции и ее инициализация указаны в «Листинге 6 – Функция pay»;

Листинг 6 – Функция pay

CREATE OR REPLACE PROCEDURE pay(staffer\_id INT, expense\_type VARCHAR(32), amount DECIMAL)

AS $$

DECLARE

expense\_type\_id INT;

BEGIN

IF NOT EXISTS (SELECT 1 FROM staff WHERE id = staffer\_id) THEN

RAISE EXCEPTION 'Staffer does not exist';

END IF;

IF NOT EXISTS (SELECT 1 FROM expense\_types WHERE title = expense\_type) THEN

RAISE EXCEPTION 'Expense type does not exist';

END IF;

SELECT id INTO expense\_type\_id

FROM expense\_types

WHERE title = expense\_type;

INSERT INTO expenses

(expense\_type\_id, staffer\_id, amount, expense\_date) VALUES

(expense\_type\_id, staffer\_id, amount, NOW());

END;

$$ LANGUAGE plpgsql;

1. триггер check\_expense\_date\_trigger, который автоматически проверяет на правильность дату при внесении новой или обновлении старой записи в таблице expenses. Код триггера и его инициализация указаны в «Листинге 7 – Триггер check\_expense\_date\_trigger»;

Листинг 7 – Триггер check\_expense\_date\_trigger

CREATE OR REPLACE FUNCTION check\_expense\_date\_trigger()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.expense\_date > NOW() THEN

RAISE EXCEPTION 'Expense cannot be committed in the future';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_expense\_date\_trigger

BEFORE INSERT OR UPDATE ON expenses

FOR EACH ROW

EXECUTE FUNCTION check\_expense\_date\_trigger();

1. триггер check\_pay\_trigger, который автоматически проверяет на покупательную способность при внесении новой или обновлении старой записи в таблице expenses. Код триггера и его инициализация указаны в «Листинге 8 – Триггер check\_pay\_trigger»;

Листинг 8 – Триггер check\_pay\_trigger

CREATE OR REPLACE FUNCTION check\_pay\_trigger()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.expense\_date <= NOW() THEN

IF NEW.amount > COALESCE(department\_remaining\_amount(

(

SELECT department\_id

FROM staff

WHERE id = NEW.staffer\_id

), NEW.expense\_type\_id

), 0) THEN

RAISE EXCEPTION 'Not enough money to make an expense';

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_pay\_trigger

BEFORE INSERT OR UPDATE ON expenses

FOR EACH ROW

EXECUTE FUNCTION check\_pay\_trigger();

1. триггер check\_budgets\_trigger, который не позволяет изменять предыдущие бюджеты. Код триггера и его инициализация указаны в «Листинге 9 – Триггер check\_budgets\_trigger»;

Листинг 9 – Триггер check\_budgets\_trigger

CREATE OR REPLACE FUNCTION check\_budgets\_trigger()

RETURNS TRIGGER AS $$

BEGIN

IF

EXTRACT(YEAR FROM NEW.datestamp) <= EXTRACT(YEAR FROM NOW())

AND EXTRACT(MONTH FROM NEW.datestamp) <= EXTRACT(MONTH FROM NOW())

THEN

RAISE EXCEPTION 'You cant change budgets in the previous or current month';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_budgets\_trigger

BEFORE INSERT OR UPDATE ON budgets

FOR EACH ROW

EXECUTE FUNCTION check\_budgets\_trigger();

1. триггер check\_budgets\_date\_trigger, который Не позволяет вносить изменения в datestamp, а также преобразует дату в дату с первым числом месяца. Код триггера и его инициализация указаны в «Листинге 10 – Триггер check\_budgets\_date\_trigger»;

Листинг 10 – Триггер check\_budgets\_date\_trigger

CREATE OR REPLACE FUNCTION check\_budgets\_date\_trigger()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.datestamp <> OLD.datestamp THEN

RAISE EXCEPTION 'You cant update the datestamp column';

END IF;

NEW.datestamp = DATE(

'01.'

|| EXTRACT(MONTH FROM NEW.datestamp)

|| '.'

|| EXTRACT(YEAR FROM NEW.datestamp)

);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER check\_budgets\_date\_trigger

BEFORE INSERT OR UPDATE ON budgets

FOR EACH ROW

EXECUTE FUNCTION check\_budgets\_date\_trigger();

# 3 использование базы данных

## 3.1 Создание DataSet

Для дальнейшей работы с базой данных был разработан dataset, указанный в «Таблице 5 – Dataset таблицы departments», «Таблице 6 – Dataset таблицы expense\_types», «Таблице 7 – Dataset таблицы staff», «Таблице 8 – Dataset таблицы budgets», «Таблице 9 – Dataset таблицы expenses».

Таблица 5 – Dataset таблицы departments

| id (идентификатор отдела в базе данных) | title  (название отдела) |
| --- | --- |
| 1 | Разработка |
| 2 | Маркетинг |
| 3 | Продажи |

Таблица 6 – Dataset таблицы expense\_types

| id  (иденти-  фикатор типа расхода в базе данных) | title  (название типа расхода) | description  (описание расхода) |
| --- | --- | --- |
| 1 | Реклама | Затраты на рекламные кампании |
| 2 | Офисные расходы | Затраты на покупку офисных расходных материалов |
| 3 | IT-расходы | Расходы на приобретение и обслуживание информационных технологий |
| 4 | Мероприятия и конференции | Затраты на организацию и участие в мероприятиях и конференциях |
| 5 | Обучение и развитие | Расходы на обучение сотрудников и развитие профессиональных навыков |
| 6 | Транспортные расходы | Затраты перемещение сотрудников |

Таблица 7 – Dataset таблицы staff

| id (идентификатор сотрудника в базе данных) | department\_id  (идентификатор отдела в базе данных) |
| --- | --- |
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |
| ... | ... |

Таблица 8 - Dataset таблицы budgets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id  (идентификатор бюджета в базе данных) | department\_id (идентификатор отдела в базе данных) | expense\_type\_id  (идентификатор типа расхода в базе данных) | datestamp  (первое число, год и месяц бюджета) | max\_amount (максимальная сумма всех расходов) |
| 1 | 1 | 1 | 01.5.2023 | 41400 |
| 2 | 1 | 2 | 01.5.2023 | 42400 |
| 3 | 1 | 3 | 01.5.2023 | 23400 |
| 4 | 1 | 4 | 01.5.2023 | 33400 |
| 5 | 1 | 5 | 01.5.2023 | 23400 |
| ... | ... | ... | ... | ... |

Таблица 9 – Dataset таблицы expenses

| expense\_type\_id  (идентификатор типа расхода в базе данных) | staffer\_id  (идентификатор категории в базе данных) | amount  (сумма расхода) | expense\_date  (дата расхода) |
| --- | --- | --- | --- |
| 1 | 1 | 41400 | 01.5.2023 |
| 2 | 1 | 42400 | 11.5.2023 |
| 3 | 2 | 23400 | 21.5.2023 |
| 4 | 3 | 33400 | 24.5.2023 |
| 5 | 4 | 23400 | 25.5.2023 |
| ... | ... | ... | ... |

## 3.2 Добавление данных в базу

Разработанный dataset был добавлен в таблицу при помощи SQL-запросов, указанных в «Листинге 11 – внесение данных в систему»

Листинг 11 – Внесение данных в систему

------------

-- Отделы --

------------

INSERT INTO departments

(id, title) VALUES

(1, 'Разработка'),

(2, 'Маркетинг'),

(3, 'Продажи');

-------------------

-- Виды расходов --

-------------------

INSERT INTO expense\_types

(id, title, description) VALUES

(1, 'Реклама', 'Затраты на рекламные кампании'),

(2, 'Офисные расходы', 'Затраты на покупку офисных расходных материалов'),

(3, 'IT-расходы', 'Расходы на приобретение и обслуживание информационных технологий'),

(4, 'Мероприятия и конференции', 'Затраты на организацию и участие в мероприятиях и конференциях'),

(5, 'Обучение и развитие', 'Расходы на обучение сотрудников и развитие профессиональных навыков'),

(6, 'Транспортные расходы', 'Затраты перемещение сотрудников');

----------------

-- Сотрудники --

----------------

INSERT INTO staff

(department\_id) VALUES

(1),

(1),

(1),

(1),

(1),

(2),

(2),

(2),

(2),

(2),

(2),

(2),

(3),

(3),

(3),

(3),

(3),

(3),

(3),

(3);

------------------------------

-- Виды расходов по отделам --

------------------------------

INSERT INTO budgets

(department\_id, expense\_type\_id, datestamp, max\_amount) VALUES

(1, 1, '01.5.2023', 41400),

(1, 2, '01.5.2023', 42400),

(1, 3, '01.5.2023', 23400),

(1, 4, '01.5.2023', 33400),

(1, 5, '01.5.2023', 23400),

(1, 6, '01.5.2023', 36400),

(2, 1, '01.5.2023', 29400),

(2, 2, '01.5.2023', 37400),

(2, 3, '01.5.2023', 28400),

(2, 4, '01.5.2023', 49400),

(2, 5, '01.5.2023', 48400),

(2, 6, '01.5.2023', 32400),

(3, 1, '01.5.2023', 51400),

(3, 2, '01.5.2023', 39400),

(3, 3, '01.5.2023', 26400),

(3, 4, '01.5.2023', 39400),

(3, 5, '01.5.2023', 38400),

(3, 6, '01.5.2023', 49400),

(1, 1, '01.6.2023', 40500),

(1, 2, '01.6.2023', 28500),

(1, 3, '01.6.2023', 40500),

(1, 4, '01.6.2023', 44500),

(1, 5, '01.6.2023', 45500),

(1, 6, '01.6.2023', 32500),

(2, 1, '01.6.2023', 32500),

(2, 2, '01.6.2023', 37500),

(2, 3, '01.6.2023', 37500),

(2, 4, '01.6.2023', 24500),

(2, 5, '01.6.2023', 30500),

(2, 6, '01.6.2023', 30500),

(3, 1, '01.6.2023', 37500),

(3, 2, '01.6.2023', 47500),

(3, 3, '01.6.2023', 31500),

(3, 4, '01.6.2023', 37500),

(3, 5, '01.6.2023', 37500),

(3, 6, '01.6.2023', 29500),

(1, 1, '01.7.2023', 25600),

(1, 2, '01.7.2023', 25600),

(1, 3, '01.7.2023', 43600),

(1, 4, '01.7.2023', 41600),

(1, 5, '01.7.2023', 31600),

(1, 6, '01.7.2023', 15600),

(2, 1, '01.7.2023', 20600),

(2, 2, '01.7.2023', 38600),

(2, 3, '01.7.2023', 22599),

(2, 4, '01.7.2023', 21600),

(2, 5, '01.7.2023', 38600),

(2, 6, '01.7.2023', 26600),

(3, 1, '01.7.2023', 46600),

(3, 2, '01.7.2023', 31600),

(3, 3, '01.7.2023', 46600),

(3, 4, '01.7.2023', 39600),

(3, 5, '01.7.2023', 31600),

(3, 6, '01.7.2023', 35600);

-------------

-- Расходы --

-------------

INSERT INTO expenses

(expense\_type\_id, staffer\_id, amount, expense\_date) VALUES

-- Отдел 1 / 5 мес

(1, 1, 41400, '01.5.2023'),

(2, 1, 42400, '11.5.2023'),

(3, 2, 23400, '21.5.2023'),

(4, 3, 33400, '24.5.2023'),

(5, 4, 23400, '25.5.2023'),

(6, 5, 36400, '26.5.2023'),

-- Отдел 2 / 5 мес

(1, 6, 29400, '01.5.2023'),

(2, 6, 37400, '02.5.2023'),

(3, 6, 28400, '03.5.2023'),

(4, 8, 49400, '04.5.2023'),

(5, 8, 48400, '05.5.2023'),

(6, 9, 32400, '06.5.2023'),

-- Отдел 3 / 5 мес

(1, 20, 51400, '21.5.2023'),

(2, 20, 39400, '22.5.2023'),

(3, 20, 26400, '23.5.2023'),

(4, 20, 39400, '24.5.2023'),

(5, 20, 38400, '25.5.2023'),

(6, 19, 49400, '26.5.2023');

## 3.3 Запросы к базе данных

Для работы с базой данных были разработаны следующие запросы:

1. Выборка всех отделов. Код запроса указан в «Листинге 12 – Запрос 1»;

Листинг 12 – Запрос 1

SELECT \* FROM departments;

Вывод запроса указан на «Рисунке 4 – Вывод запроса 1».

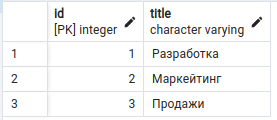


Рисунок 4 – Вывод запроса 1

1. Выборка сотрудников определенного отдела. Код запроса указан в «Листинге 13 – Запрос 2»;

Листинг 13 – Запрос 2

SELECT \* FROM staff WHERE department\_id = 1;

Вывод запроса указан на «Рисунке 5 – Вывод запроса 2».

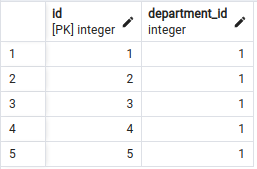


Рисунок 5 – Вывод запроса 2

1. Сортировка отделов по названию. Код запроса указан в «Листинге 14 – Запрос 3»;

Листинг 14 – Запрос 3

SELECT \* FROM departments ORDER BY title;

Вывод запроса указан на «Рисунке 6 – Вывод запроса 3».

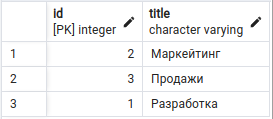


Рисунок 6 – Вывод запроса 3

1. Общая сумма расходов. Код запроса указан в «Листинге 15 – Запрос 4»;

Листинг 15 – Запрос 4

SELECT SUM(amount) FROM expenses;

Вывод запроса указан на «Рисунке 7 - Вывод запроса 4».

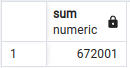


Рисунок 7 – Вывод запроса 4

1. Общая сумма расходов по каждому отделу. Код запроса указан в «Листинге 16 – Запрос 5»;

Листинг 16 – Запрос 5

SELECT department\_id, SUM(amount)

FROM expenses

JOIN staff ON expenses.staffer\_id = staff.id

GROUP BY department\_id;

Вывод запроса указан на «Рисунке 8 – Вывод запроса 5»

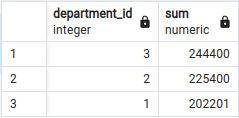


Рисунок 8 – Вывод запроса 5

1. Самый большой бюджет по отделам. Код запроса указан в «Листинге 17 – Запрос 6»;

Листинг 17 – Запрос 6

SELECT department\_id, MAX(max\_amount)

FROM budgets

GROUP BY department\_id;

Вывод запроса указан на «Рисунке 9 – Вывод запроса 6».

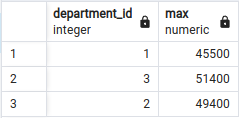


Рисунок 9 – Вывод запроса 6

1. Отделы, которые превысили заданный бюджет. Код запроса указан в «Листинге 9 – Запрос 7»;

Листинг 18 – Запрос 7

SELECT department\_id

FROM (

SELECT department\_id, AVG(amount) as avg\_expenses

FROM expenses

JOIN staff ON expenses.staffer\_id = staff.id

GROUP BY department\_id

) AS dept\_expenses

WHERE avg\_expenses > 35000;

Вывод запроса указан на «Рисунке 10 – Вывод запроса 7».

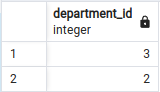


Рисунок 10 – Вывод запроса 7

1. Использование CASE для категоризации расходов. Код запроса указан в «Листинге 19 – Запрос 8»;

Листинг 19 – Запрос 8

SELECT id, amount,

CASE

WHEN amount <= 20000 THEN 'low'

WHEN amount <= 40000 THEN 'medium'

ELSE 'high'

END as expense\_category

FROM expenses;

Вывод запроса указан на «Рисунке 11 – Вывод запроса 8».

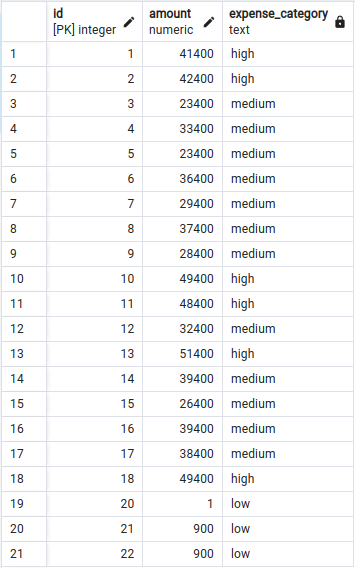


Рисунок 11 – Вывод запроса 8

1. Отделы, в которых работает более 6 сотрудников. Код запроса указан в «Листинге 20 – Запрос 9»;

Листинг 20 – Запрос 9

SELECT department\_id

FROM staff

GROUP BY department\_id

HAVING COUNT(DISTINCT id) > 6;

Вывод запроса указан на «Рисунке 12 – Вывод запроса 9».

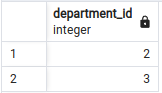


Рисунок 12 – Вывод запроса 9

1. Оконная функция для ранжирования расходов внутри каждого отдела. Код запроса указан в «Листинге 21 – Запрос 10»;

Листинг 21 – Запрос 10

SELECT

department\_id

,staffer\_id

,amount

,RANK() OVER(

PARTITION BY department\_id

ORDER BY amount DESC

) as rank

FROM expenses

JOIN staff ON expenses.staffer\_id = staff.id;

Вывод запроса указан на «Рисунке 13 – Вывод запроса 10».

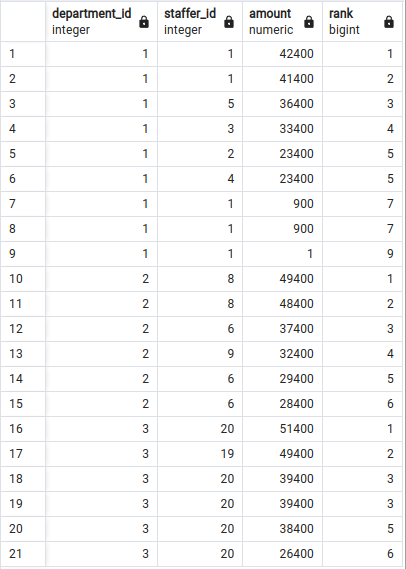


Рисунок 13 – Вывод запроса 10

1. Общая сумма расходов по каждому виду расходов. Код запроса указан в «Листинге 22 – Запрос 11»;

Листинг 22 – Запрос 11

SELECT expense\_type\_id, SUM(amount)

FROM expenses

GROUP BY expense\_type\_id;

Вывод запроса указан на «Рисунке 14 – Вывод запроса 11».

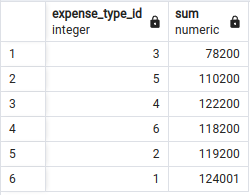


Рисунок 14 – Вывод запроса 11

1. Средний бюджет по отделам. Код запроса указан в «Листинге 23 – Запрос 12»;

Листинг 23 – Запрос 12

SELECT department\_id, AVG(max\_amount)

FROM budgets

GROUP BY department\_id;

Вывод запроса указан на «Рисунке 15 – Вывод запроса 12».

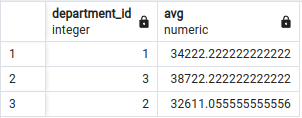


Рисунок 15 – Вывод запроса 12

1. Отделы, в которых есть сотрудники, чьи общие расходы превышают 100000. Код запроса указан в «Листинге 24 – Запрос 13»;

Листинг 24 – Запрос 13

SELECT DISTINCT department\_id

FROM (

SELECT s.department\_id, SUM(e.amount) as total\_expenses

FROM staff s

JOIN expenses e ON s.id = e.staffer\_id

GROUP BY s.id, s.department\_id

) AS staff\_expenses

WHERE total\_expenses > 100000;

Вывод запроса указан на «Рисунке 16 – Вывод запроса 13».

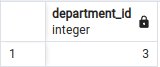


Рисунок 16 – Вывод запроса 13

1. Вид расходов с наибольшей общей суммой. Код запроса указан в «Листинге 25 – Запрос 14»;

Листинг 25 – Запрос 14

SELECT title, SUM(amount) as total\_expense

FROM expenses, expense\_types

WHERE expense\_type\_id = expense\_types.id

GROUP BY title

ORDER BY total\_expense DESC LIMIT 1;

Вывод запроса указан на «Рисунке 17 – Вывод запроса 14».

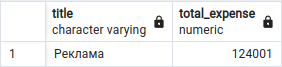


Рисунок 17 – Вывод запроса 14

1. Общая сумма расходов по месяцам. Код запроса указан в «Листинге 26 – Запрос 15»;

Листинг 26 – Запрос 15

SELECT

DATE\_TRUNC('month', expense\_date) as month

,SUM(amount)

FROM expenses

GROUP BY month;

Вывод запроса указан на «Рисунке 18 – Вывод запроса 15».

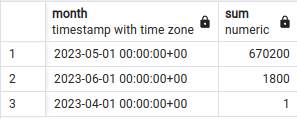


Рисунок 18 – Вывод запроса 15

1. Процентный расход каждого сотрудника внутри его отдела. Код запроса указан в «Листинге 27 – Запрос 16»;

Листинг 27 – Запрос 16

SELECT e.staffer\_id, SUM(e.amount) / SUM(SUM(e.amount)) OVER (PARTITION BY s.department\_id) \* 100 as percent

FROM expenses e

JOIN staff s ON e.staffer\_id = s.id

GROUP BY e.staffer\_id, s.department\_id;

Вывод запроса указан на «Рисунке 19 – Вывод запроса 16».

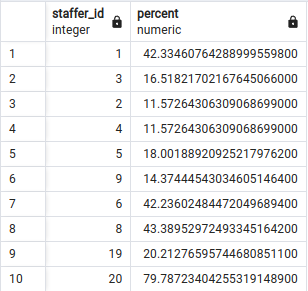


Рисунок 19 – Вывод запроса 16

1. Количество сотрудников в каждом отделе. Код запроса указан в «Листинге 28 – Запрос 17»;

Листинг 28 – Запрос 17

SELECT title, COUNT(\*)

FROM staff, departments

WHERE department\_id = departments.id

GROUP BY title;

Вывод запроса указан на «Рисунке 20 – Вывод запроса 17».

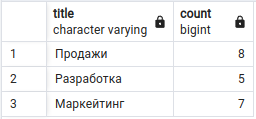


Рисунок 20 – Вывод запроса 17

1. Бюджет по видам расходов, превышающий определенную сумму. Код запроса указан в «Листинге 29 – Запрос 18»;

Листинг 29 – Запрос 18

SELECT DISTINCT title

FROM budgets, expense\_types

WHERE

expense\_type\_id = expense\_types.id

AND max\_amount > 49000;

Вывод запроса указан на «Рисунке 21 – Вывод запроса 18».

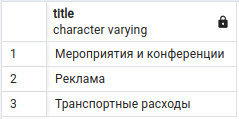


Рисунок 21 – Вывод запроса 18

1. Количество расходов каждого сотрудника. Код запроса указан в «Листинге 30 – Запрос 19»;

Листинг 30 – Запрос 19

SELECT staffer\_id, COUNT(\*)

FROM expenses

GROUP BY staffer\_id

ORDER BY staffer\_id;

Вывод запроса указан на «Рисунке 22 – Вывод запроса 19».

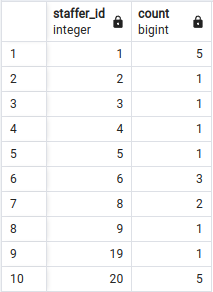


Рисунок 22 – Вывод запроса 19

1. Сотрудники, чьи расходы превысили 1000 за последний месяц. Код запроса указан в «Листинге 31 – Запрос 20»;

Листинг 31 – Запрос 20

SELECT staffer\_id

FROM expenses

WHERE expense\_date > (CURRENT\_DATE - INTERVAL '1 month')

GROUP BY staffer\_id

HAVING SUM(amount) > 1000;

Вывод запроса указан на «Рисунке 23 – Вывод запроса 20».



Рисунок 23 – Вывод запроса 20

## 3.4 Тестирование функций и триггеров

Было проведено тестирование функций на новых наборах данных, был описан ожидаемый результат и фактический результат и сделан вывод о работоспособности функции.

1. функция department\_expenses(depart\_id INT). Выводит таблицу всех расходов конкретного отдела по годам, месяцам и типам расходов.

Результат тестирования при корректных данных приведен на «Рисунке 24 – Тестирование department\_expenses 1»

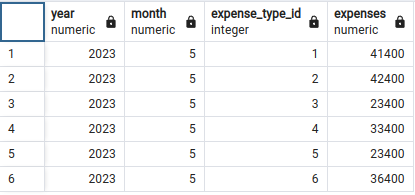


Рисунок 24 – Тестирование department\_expenses 1

Результат тестирования при не корректных данных приведен на «Рисунке 25 – Тестирование department\_expenses 2»

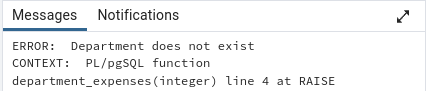


Рисунок 25 – Тестирование department\_expenses 2

1. функция department\_expenses\_from\_to(depart\_id INT, start\_date DATE, end\_date DATE). Выводит cумму всех расходов конкретного отдела от даты до даты

Результат тестирования при корректных данных приведен на «Рисунке 26 – Тестирование department\_expenses\_from\_to 1».

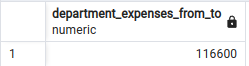


Рисунок 26 – Тестирование department\_expenses\_from\_to 1

Результат тестирования при не корректных данных приведен на «Рисунке 27 – Тестирование department\_expenses\_from\_to 2».

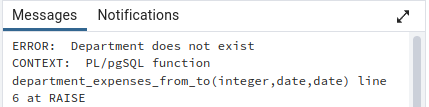


Рисунок 27 – Тестирование department\_expenses\_from\_to 2

1. функция staff\_expenses(). Выводит таблицу всех расходов сотрудников.

Результат тестирования приведен на «Рисунке 28 – Тестирование staff\_expenses»

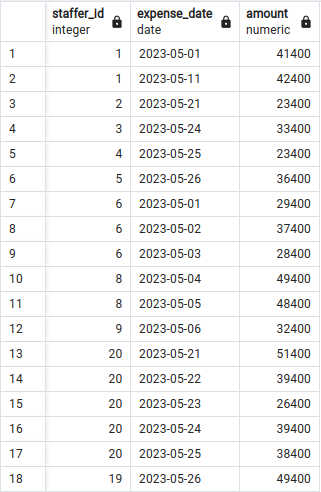


Рисунок 28 – Тестирование staff\_expenses

1. функция department\_remaining\_amount(depart\_id INT, type\_id INT). Выводит оставшуюся не потраченную сумму отдела в конкретной категории.

Результат тестирования при корректных данных приведен на «Рисунке 29 – Тестирование department\_remaining\_amount 1»

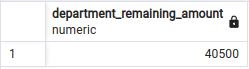


Рисунок 29 – Тестирование department\_remaining\_amount 1

Результат тестирования при не корректных данных приведен на «Рисунке 30 – Тестирование department\_remaining\_amount 2», «Рисунке 31 – Тестирование department\_remaining\_amount 3»

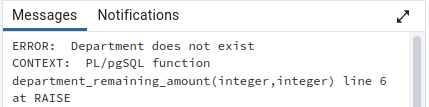


Рисунок 30 – Тестирование department\_remaining\_amount 2

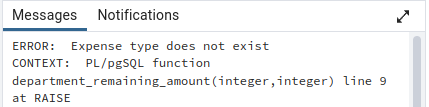


Рисунок 31 – Тестирование department\_remaining\_amount 3

1. процедура pay(staffer\_id INT, expense\_type VARCHAR(32), amount DECIMAL). Совершает расход.

Результат тестирования при корректных данных приведен на «Рисунке 32 – Тестирование pay 1»

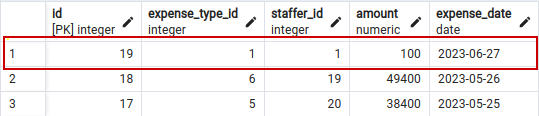


Рисунок 32 – Тестирование pay 1

Результат тестирования при не корректных данных приведен на «Рисунке 33 – Тестирование pay 2», «Рисунке 34 – Тестирование pay 3»

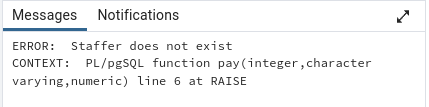


Рисунок 33 – Тестирование pay 2

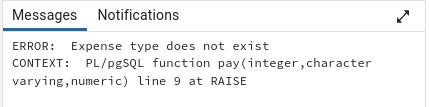


Рисунок 34 – Тестирование pay 3

1. триггер check\_expense\_date\_trigger. Проверяет на правильность дату при каждом расходе.

Результат тестирования при не корректных данных приведен на «Рисунке 35 – Тестирование check\_expense\_date\_trigger»

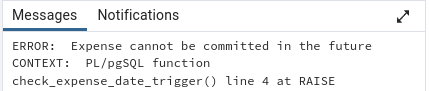


Рисунок 35 – Тестирование check\_expense\_date\_trigger

1. функция check\_pay\_trigger. Проверяет на покупательную способность при каждом расходе.

Результат тестирования при не корректных данных приведен на «Рисунке 36 – Тестирование check\_pay\_trigger»

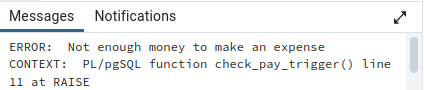


Рисунок 36 – Тестирование check\_pay\_trigger

1. функция check\_budgets\_trigger. Не позволяет изменять предыдущие бюджеты.

Результат тестирования при не корректных данных приведен на «Рисунке 37 – Тестирование check\_budgets\_trigger»

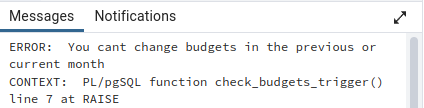


Рисунок 37 – Тестирование check\_budgets\_trigger

1. функция check\_budgets\_date\_trigger. Не позволяет в таблице budgets вносить изменения в datestamp, а также преобразует дату в дату с первым числом месяца.

Результат тестирования при корректных данных приведен на «Рисунке 38 – Тестирование check\_budgets\_date\_trigger 1»

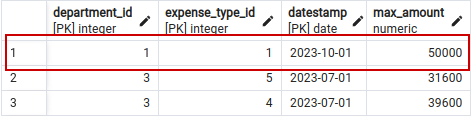


Рисунок 38 – Тестирование check\_budgets\_date\_trigger 1

Результат тестирования при не корректных данных приведен на «Рисунке 39 – Тестирование check\_budgets\_date\_trigger 2»

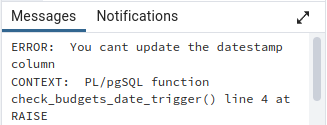


Рисунок 39 – Тестирование check\_budgets\_date\_trigger 2

# заключение

В рамках этой курсовой работы мною была разработана база данных, используя СУБД PostgreSQL. Этот проект стал платформой для приобретения мною важных навыков в области проектирования и создания баз данных, а также изучения SQL-запросов и работы с определённой СУБД.

В процессе выполнения задания я освоил основные аспекты моделирования данных, определенную структуру базы данных, создал таблицы, установил связи между ними и заполнил их информацией. Я также изучил различные функциональные возможности PostgreSQL, включая создание и применение функций, триггеров и процедур.

Разработка этой базы данных подтолкнула меня к более глубокому пониманию важности эффективной организации и сохранения данных для успешной работы информационной системы. Я осознал важность правильного проектирования базы данных для обеспечения производительности и масштабируемости системы, а также для обеспечения надежности и безопасности данных.

В ходе работы с PostgreSQL я столкнулся с определёнными сложностями и проблемами, но, глубоко изучив документацию, я смог успешно преодолеть их и получить ценный опыт.

В общем и целом, эта курсовая работа дала мне возможность углубить мои знания и навыки в сфере баз данных и в частности PostgreSQL. Я получил опыт в области проектирования, создания и оптимизации баз данных, который будет полезен для моего будущего профессионального развития в сфере информационных технологий. PostgreSQL – мощная и гибкая СУБД, и я уверен, что приобретенные мной знания и навыки будут использоваться мной в последующих проектах для создания надежных и эффективных баз данных.

# список использованных источников

1. ГОСТ 19.201-78. Режим доступа: https://www.swrit.ru/doc/espd/19.201-78.pdf (дата обращения: 18.04.2023).

2. Курс в ЛМС Московского политеха «Разработка технических текстов и документации». Режим доступа: https://online.mospolytech.ru/course/view.php?id=1689 (дата обращения: 18.04.2023).

3. Презентация «Модель «сущность-связь». Режим доступа: <https://foreva.susu.ru/courses/db/lecture3.pdf> (дата обращения 29.05.2023)

4. Триггерные процедуры в PostgreSQL. Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/plpgsql-trigger> (дата обращения 29.05.2023)

5. Форум по PosgreSQL Wiki.postgresql. Режим доступа: <https://wiki.postgresql.org/wiki/Main_Page/ru> (дата обращения 29.05.2023).

6. «PostgreSQL изнутри» — Е. Рогов. Режим доступа: <https://postgrespro.ru/education/books/internals> (дата обращения 29.05.2023).

7. «PostgreSQL. Основы языка SQL» — Е. Моргунов. Режим доступа: <https://postgrespro.ru/education/books/sqlprimer> (дата обращения 29.05.2023).

8. Руководство по подготовке курсовых работ (проектов) и выпускных квалификационных работ. Режим доступа: <http://www.skf-mtusi.ru/files/vkr/MU_DP_KP_MTUSI.pdf> (дата обращения 29.05.2023)

9. Правила оформления курсовых и дипломных работ. Режим доступа: <https://studfile.net/preview/3009968/page:5/> (дата обращения 29.05.2023).

10. «Оптимизация запросов PostgreSQL» — Г. Домбровская, Б. Новиков, А. Бейликова. Режим доступа: <https://dmkpress.com/files/PDF/978-5-97060-963-7.pdf> (дата обращения 29.05.2023).

11. «Основы технологий баз данных: учебное пособие» — Б. Новиков, Е. Горшкова, Н. Графеева. Режим доступа: <https://postgrespro.ru/education/books/dbtech> (дата обращения: 29.05.2023).

12. Правила оформления отчета к лабораторным и курсовым работам. Режим доступа: <https://publications.hse.ru/mirror/pubs/share/direct/227003831> (дата обращения 29.05.2023).

13. Статья «Моделирование данных: обзор». Режим доступа <https://habr.com/ru/articles/556790/> (дата обращения 29.05.2023).

14. Нотация Питера Чена. Режим доступа: <https://studme.org/77222/informatika/notatsiya_pitera_chena> (дата обращения 29.05.2023).

15. Официальная документация разработчика PostgreSQL. Режим доступа: <https://www.postgresql.org/> (дата обращения: 29.05.2023).

16. ГОСТ 7.32-2017. Режим доступа: <https://files.stroyinf.ru/Data2/1/4293742/4293742537.pdf> (дата обращения: 29.05.2023).