**Федеральное государственное образовательное**

**учреждение высшего образования**

**«Чувашский государственный университет им. И.Н. Ульянова»**

Кафедра вычислительной техники

Лабораторная работа № 1

Реляционная СУБД PostgreSQL.

Выполнил: студент гр. ИВТ-42-21

Андреев Александр Всеволодович

Проверил: Ржавин В.В.

г. Чебоксары 2024

Создать и запустить скрипт создания таблиц (пример приведён в init\_db.sql)

**CREATE** **TABLE** **IF** **NOT** **EXISTS** clients(

id SERIAL **PRIMARY** **KEY**,

**label** **TEXT**

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** account\_statuses(

id SERIAL **PRIMARY** **KEY**,

**label** **TEXT**

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** accounts(

id SERIAL **PRIMARY** **KEY**,

balance **NUMERIC**(20,2),

is\_system **INT**,

id\_client **INT** **REFERENCES** clients (id)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** payments(

id SERIAL **PRIMARY** **KEY**,

dt **TIMESTAMP**,

amount **NUMERIC**(20,2),

id\_account\_from **INT** **REFERENCES** accounts (id),

id\_account\_to **INT** **REFERENCES** accounts (id)

);

**CREATE** **TABLE** **IF** **NOT** **EXISTS** archive\_payments(

id SERIAL **PRIMARY** **KEY**,

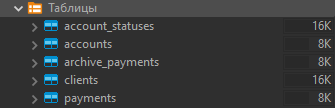
dt **TIMESTAMP**,

amount **NUMERIC**(20,2),

id\_account\_from **INT** **REFERENCES** accounts (id),

id\_account\_to **INT** **REFERENCES** accounts (id)

);



Создать и запустить скрипт заполнения данными таблиц(аналогичный fill\_db.sql),

удовлетворяющий следующим требованиям: 5 записей в таблице Клиенты, с

осмысленными названиями; 10 записей в таблице Счета, при этом должны быть два

клиента, у которых больше одного счёта; 20 записей в таблице Платежи, с

произвольными суммой и датой, каждый счёт должен поучаствовать хотя бы в одном

платеже.

-- Вставляем клиентов

**INSERT** **INTO** clients(id, **label**)

**VALUES**

(1, **'John Doe'**),

(2, **'Jane Smith'**),

(3, **'Acme Corp.'**),

(4, **'XYZ Inc.'**),

(5, **'Beta LLC'**);

-- Вставляем 10 счетов. У клиентов 1 и 2 больше одного счёта.

**INSERT** **INTO** accounts(id, balance, is\_system, id\_client)

**VALUES**

(1, 1000.0, 0, 1),

(2, 2000.0, 0, 1),

(3, 3000.0, 0, 2),

(4, 4000.0, 0, 2),

(5, 5000.0, 1, 3),

(6, 6000.0, 0, 3),

(7, 7000.0, 0, 4),

(8, 8000.0, 0, 5),

(9, 9000.0, 0, 5),

(10, 10000.0, 0, 5);

-- Вставляем 20 записей в таблицу Payments, все счета участвуют в хотя бы одном платеже

**INSERT** **INTO** payments(dt, amount, id\_account\_from, id\_account\_to)

**VALUES**

(**'2022-09-01'**, 100.0, 1, 2),

(**'2022-09-02'**, 200.0, 3, 2),

(**'2022-09-03'**, 150.0, 3, 4),

(**'2022-09-03'**, 150.0, 4, 5),

(**'2022-09-04'**, 300.0, 2, 5),

(**'2022-09-05'**, 100.0, 1, 3),

(**'2022-09-06'**, 200.0, 2, 6),

(**'2022-09-07'**, 150.0, 5, 7),

(**'2022-09-08'**, 300.0, 6, 8),

(**'2022-09-09'**, 250.0, 3, 9),

(**'2022-09-10'**, 150.0, 4, 10),

(**'2022-09-11'**, 500.0, 1, 10),

(**'2022-09-12'**, 250.0, 7, 9),

(**'2022-09-13'**, 150.0, 3, 10),

(**'2022-09-14'**, 300.0, 2, 8),

(**'2022-09-15'**, 200.0, 6, 9),

(**'2022-09-16'**, 150.0, 1, 7),

(**'2022-09-17'**, 100.0, 2, 4),

(**'2022-09-18'**, 400.0, 4, 1),

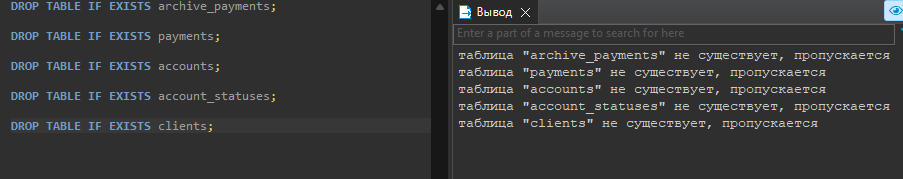
(**'2022-09-19'**, 300.0, 3, 2);

**INSERT** **INTO** account\_statuses(id,**label**)

**VALUES** (1, **'Активный'**), (2, **'Неактивный'**), (3, **'Системный'**);

(**'2022-09-19'**, 300.0, 3, 2);

Создать и запустить скрипт drop\_db.sql, убедиться что работает полный цикл запросов

init\_db.sql, fill\_db.sql, drop\_db.sql. 

**DROP** **TABLE** **IF** **EXISTS** archive\_payments;

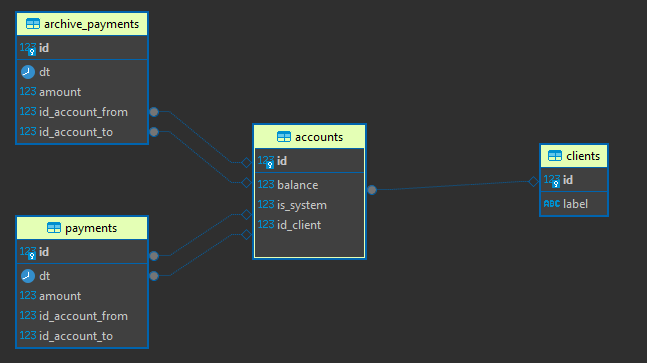
**DROP** **TABLE** **IF** **EXISTS** payments;

**DROP** **TABLE** **IF** **EXISTS** accounts;

**DROP** **TABLE** **IF** **EXISTS** account\_statuses;

**DROP** **TABLE** **IF** **EXISTS** clients;

Диаграмма на данном этапе



-- Добавляем поле "id\_status" в таблицу "accounts" с внешним ключом к "account\_statuses"

**ALTER** **TABLE** accounts

**ADD** **COLUMN** id\_status **INT**,

**ADD** **FOREIGN** **KEY** (id\_status) **REFERENCES** account\_statuses (id);

-- Удаляем поле "is\_system" из таблицы "accounts"

**ALTER** **TABLE** accounts

**DROP** **COLUMN** is\_system;

-- Добавляем поле "creation\_date" в таблицу "accounts" с значением по умолчанию

**ALTER** **TABLE** accounts

**ADD** **COLUMN** creation\_date **TIMESTAMP** **DEFAULT** **CURRENT\_TIMESTAMP**;

-- Устанавливаем значение по умолчанию для поля "balance"

**ALTER** **TABLE** accounts

**ALTER** **COLUMN** balance **SET** **DEFAULT** 0;

-- Устанавливаем значение по умолчанию для поля "dt" в таблице "payments"

**ALTER** **TABLE** payments

**ALTER** **COLUMN** dt **SET** **DEFAULT** **CURRENT\_TIMESTAMP**;

-- Устанавливаем свойство NOT NULL для полей "dt" и "amount" в таблице "payments"

**ALTER** **TABLE** payments

**ALTER** **COLUMN** dt **SET** **NOT** **NULL**,

**ALTER** **COLUMN** amount **SET** **NOT** **NULL**;

-- Вставляем данные в таблицу "account\_statuses"

**INSERT** **INTO** account\_statuses(id, **label**)

**VALUES**

(1, **'Активный'**),

(2, **'Неактивный'**),

(3, **'Закрыт'**);

-- Вставляем данные в таблицу "clients"

**INSERT** **INTO** clients(id, **label**)

**VALUES**

(1, **'John Doe'**),

(2, **'Jane Smith'**),

(3, **'Acme Corp.'**),

(4, **'XYZ Inc.'**),

(5, **'Beta LLC'**);

-- Вставляем данные в таблицу "accounts", добавляя "id\_status" и "creation\_date"

**INSERT** **INTO** accounts(id, balance, id\_client, ***id\_status***, ***creation\_date***)

**VALUES**

(1, 1000.0, 1, 1, **'2022-01-01'**),

(2, 2000.0, 1, 1, **'2022-01-02'**),

(3, 3000.0, 2, 1, **'2022-01-03'**),

(4, 4000.0, 2, 1, **'2022-01-04'**),

(5, 5000.0, 3, 2, **'2022-01-05'**),

(6, 6000.0, 3, 2, **'2022-01-06'**),

(7, 7000.0, 4, 1, **'2022-01-07'**),

(8, 8000.0, 5, 3, **'2022-01-08'**),

(9, 9000.0, 5, 3, **'2022-01-09'**),

(10, 10000.0, 5, 3, **'2022-01-10'**);

-- Вставляем данные в таблицу "payments", учитывая новые ограничения

**INSERT** **INTO** payments(dt, amount, id\_account\_from, id\_account\_to)

**VALUES**

(**'2022-09-01'**, 100.0, 1, 2),

(**'2022-09-02'**, 200.0, 3, 2),

(**'2022-09-03'**, 150.0, 3, 4),

(**'2022-09-03'**, 150.0, 4, 5),

(**'2022-09-04'**, 300.0, 2, 5),

(**'2022-09-05'**, 100.0, 1, 3),

(**'2022-09-06'**, 200.0, 2, 6),

(**'2022-09-07'**, 150.0, 5, 7),

(**'2022-09-08'**, 300.0, 6, 8),

(**'2022-09-09'**, 250.0, 3, 9),

(**'2022-09-10'**, 150.0, 4, 10),

(**'2022-09-11'**, 500.0, 1, 10),

(**'2022-09-12'**, 250.0, 7, 9),

(**'2022-09-13'**, 150.0, 3, 10),

(**'2022-09-14'**, 300.0, 2, 8),

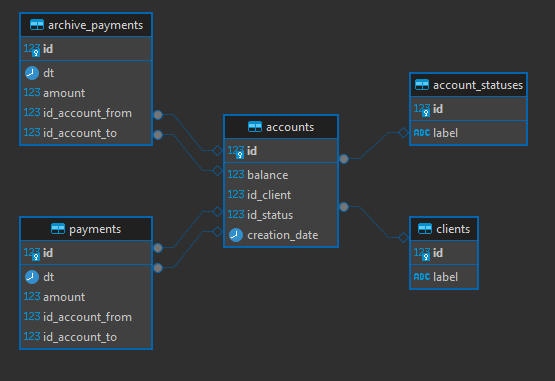
(**'2022-09-15'**, 200.0, 6, 9),

(**'2022-09-16'**, 150.0, 1, 7),

(**'2022-09-17'**, 100.0, 2, 4),

(**'2022-09-18'**, 400.0, 4, 1),

(**'2022-09-19'**, 300.0, 3, 2);



Создать и запустить скрипт select\_from\_db.sql, который содержит следующие запросы:

- список клиентов, являющихся плательщиками хотя бы по одному платежу(есть в

select\_from\_db.sql);

- список клиентов, являющихся получателями хотя бы по одному платежу;

- id платежа с самой большой суммой;

- id трёх последних по дате платежей;

- общая сумма всех проведённых платежей;

- количество исходящих платежей по каждому счёту;

- количество входящих платежей по каждому счёту;

- все платежи с указанием даты, суммы, id счёта получателя, названия клиента-

получателя, id счёта плательщика, названия клиента-плательщика, упорядоченные по

убыванию даты.

-- Список клиентов, являющихся плательщиками хотя бы по одному платежу

**SELECT**

**DISTINCT** clients.**label** **AS** *payers*

**FROM**

payments

**INNER** **JOIN**

accounts **ON** accounts.id = payments.id\_account\_from

**INNER** **JOIN**

clients **ON** clients.id = accounts.id\_client;

-- Список клиентов, являющихся получателями хотя бы по одному платежу

**SELECT**

**DISTINCT** clients.**label** **AS** *receivers*

**FROM**

payments

**INNER** **JOIN**

accounts **ON** accounts.id = payments.id\_account\_to

**INNER** **JOIN**

clients **ON** clients.id = accounts.id\_client;

-- id платежа с самой большой суммой

**SELECT**

id

**FROM**

payments

**ORDER** **BY**

amount **DESC**

**LIMIT** 1;

-- id трех последних по дате платежей

**SELECT**

id

**FROM**

payments

**ORDER** **BY**

dt **DESC**

**LIMIT** 3;

-- Общая сумма всех проведенных платежей

**SELECT**

**SUM**(amount) **AS** *total\_payments*

**FROM**

payments;

-- Количество исходящих платежей по каждому счету

**SELECT**

id\_account\_from **AS** *account\_id*,

**COUNT**(\*) **AS** *outgoing\_count*

**FROM**

payments

**GROUP** **BY**

id\_account\_from;

-- Количество входящих платежей по каждому счету

**SELECT**

id\_account\_to **AS** *account\_id*,

**COUNT**(\*) **AS** *incoming\_count*

**FROM**

payments

**GROUP** **BY**

id\_account\_to;

-- Все платежи с указанием даты, суммы, id счета получателя, названия клиента-получателя,

-- id счета плательщика, названия клиента-плательщика, упорядоченные по убыванию даты

**SELECT**

payments.dt,

payments.amount,

payments.id\_account\_to,

clients\_receiver.**label** **AS** *receiver\_client*,

payments.id\_account\_from,

clients\_payer.**label** **AS** *payer\_client*

**FROM**

payments

**INNER** **JOIN**

accounts *accounts\_to* **ON** *accounts\_to*.id = payments.id\_account\_to

**INNER** **JOIN**

clients *clients\_receiver* **ON** *clients\_receiver*.id = *accounts\_to*.id\_client

**INNER** **JOIN**

accounts *accounts\_from* **ON** *accounts\_from*.id = payments.id\_account\_from

**INNER** **JOIN**

clients *clients\_payer* **ON** *clients\_payer*.id = *accounts\_from*.id\_client

**ORDER** **BY**

payments.dt **DESC**;

**Лексические правила**

1. **Именование таблиц и столбцов**:
   * Имена таблиц обычно пишутся в единственном числе (client, account, payment), что отражает сущность, которую они представляют.
   * Столбцы именуются так, чтобы четко указывать их содержимое. Часто используется нотация в стиле snake\_case (account\_balance, creation\_date).
2. **Имена внешних ключей**:
   * Если столбец является внешним ключом, он должен отражать, к какой таблице он относится. Например, id\_client означает, что это ссылка на таблицу clients.

**Синтаксические правила**

1. **Организация SQL-запросов**:
   * Ключевые слова (SELECT, FROM, WHERE, JOIN, ORDER BY, и т.д.) обычно пишутся в верхнем регистре, чтобы легко их выделять.
   * Запросы часто разбиваются на несколько строк для улучшения читаемости.

 **Использование комментариев**:

* Комментарии используются для пояснения сложных или нестандартных частей кода. Обычно комментарии предшествуют коду, который они объясняют, и начинаются с --.
* Для многострочных комментариев используются /\* ... \*/.

 **Использование пробелов и отступов**:

* Пробелы используются для разделения ключевых частей запроса.
* Отступы используются для вложенных запросов или условий.