1.

Содержание

Введение 4
1 Аналитический раздел 5
1.1 Описание предметной области 5
1.2 Анализ существующих решений 5
1.2.1 SAS Data Integration Studio 5
1.2.2 Informatica PowerCenter 7
1.2.3 Apache NiFi 8
1.2.4 Сравнение существующих решений 9
1.3 План выполнения запроса PostgreSQL 9
1.3.1 Построение дерева синтаксического анализа 9
1.3.2 Оптимизация 12
1.3.3 Создание плана 13
1.3.4 Выполнение 14
Заключение 15
Список использованных источников 16

Введение

Целью данной работы является разработка метода построения дерева синтаксического анализа языка SQL на основе графических примитивов.

Для достижения поставленной цели ставятся следующие задачи.

— Проанализировать предметную область.

— Рассмотреть и проанализировать существующие графические ETL-инструменты.

— Разработать метод построения дерева синтаксического анализа на основе графических примитивов.

— Разработать программное обеспечение, реализующее данный метод.

— Провести исследование построения дерева синтаксического анализа.

1 Аналитический раздел

1.1 Описание предметной области

Под деревом синтаксического анализа понимается набор хранимых в памяти взаимосвязанных между собой типов [1]. Дерево синтаксического анализа может быть создано из строки sql запроса [2]. На примере PostgreSQL [3] будет показано какие этапы ему нужны для выполнения запроса и какие структуры для этого он создает в памяти [4].

1.2 Анализ существующих решений

1.2.1 SAS Data Integration Studio

SAS Data Integration Studio – это инструмент визуального дизайна [5]. SAS DIS создает, реализует и управляет процессами интеграции данных независимо от источников данных, приложений или платформ.

Многопользовательская среда SAS DIS обеспечивает совместную работу над корпоративными проектами. Создание и управление данными и метаданными улучшаются за счет обширного анализа воздействия потенциальных изменений, внесенных во все процессы интеграции данных.

Встроенный в SAS DIS редактор jobs editor позволяет создавать задачи. На рисунке 1.1 продемонстрировано создание задачи.

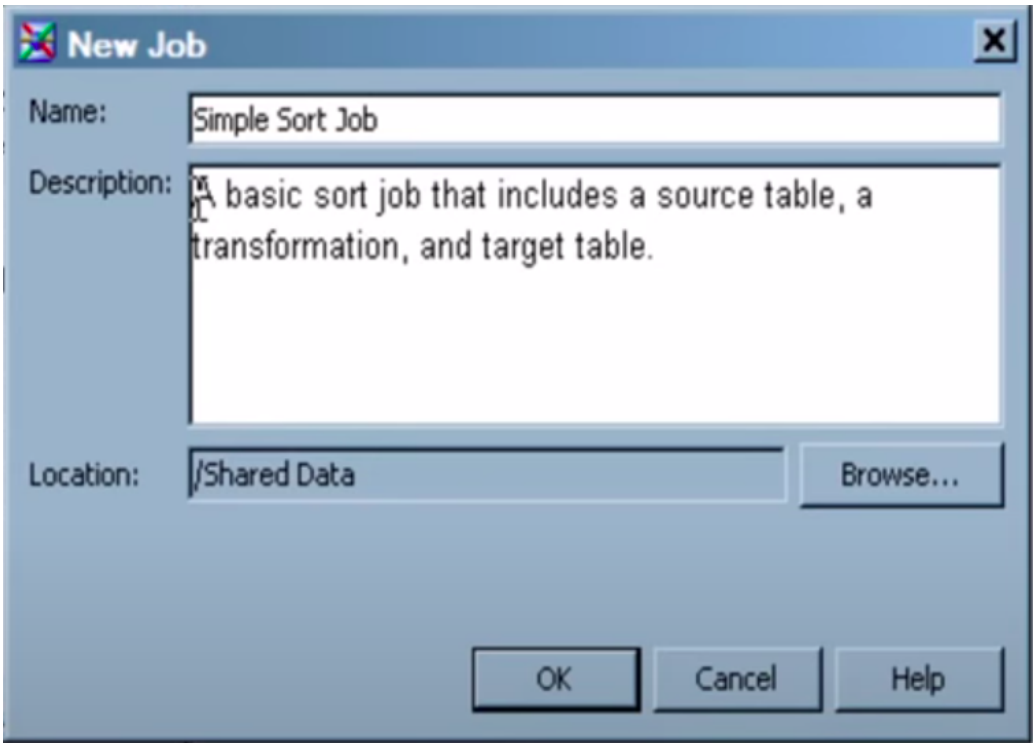

Рисунок 1.1 — Демонстрация создания задачи во встроенном редакторе jobs editor программного обеспечения SAS Data Integration Studio

После создания задачи открывается холст на котором возможно располагать графические примитивы, которые впоследствии будут преобразованы в запрос. Возможные преобразования представлены на рисунке 1.2. При расположении на холсте каждое преобразование превращается в графический примитив с данными о выполняемом преобразовании. На рисунке 1.3 продемонстрирована созданная задача.

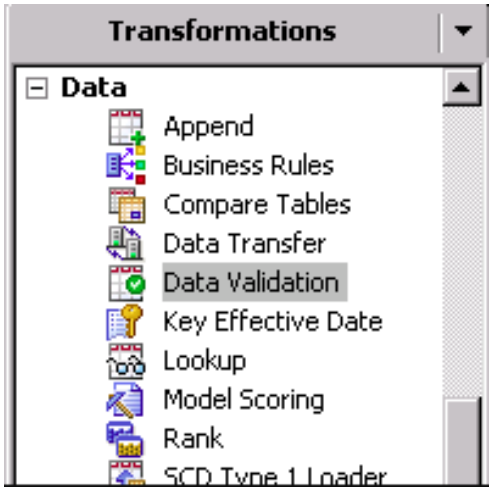

Рисунок 1.2 — Демонстрация панели трансформации во встроенном редакторе jobs editor программного обеспечения SAS Data Integration Studio

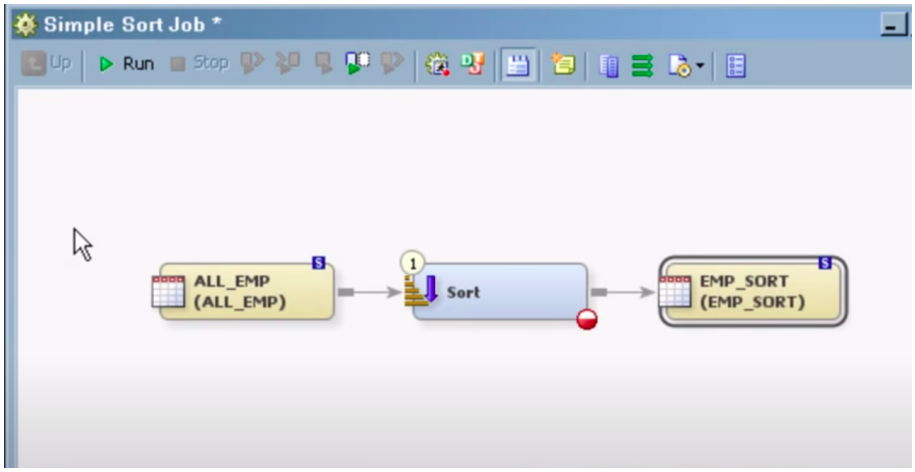

Рисунок 1.3 — Демонстрация созданной задачи во встроенном редакторе jobs editor программного обеспечения SAS Data Integration Studio

1.2.2 Informatica PowerCenter

Informatica PowerCenter – платформа интеграции данных, основанная на работе со структурами данных в визуальной среде [6]. Данная платформа предоставляет возможность работать со структурами данных без написания программного кода, что позваоляет не только ускорить получение данных бизнесом, но и облегчить взаимодействие с данными пользователям.

На рисунке 1.4 показано окно изменения данных. На рисунке 1.5 показан холст, на котором расположены графические элементы, связанные между собой.

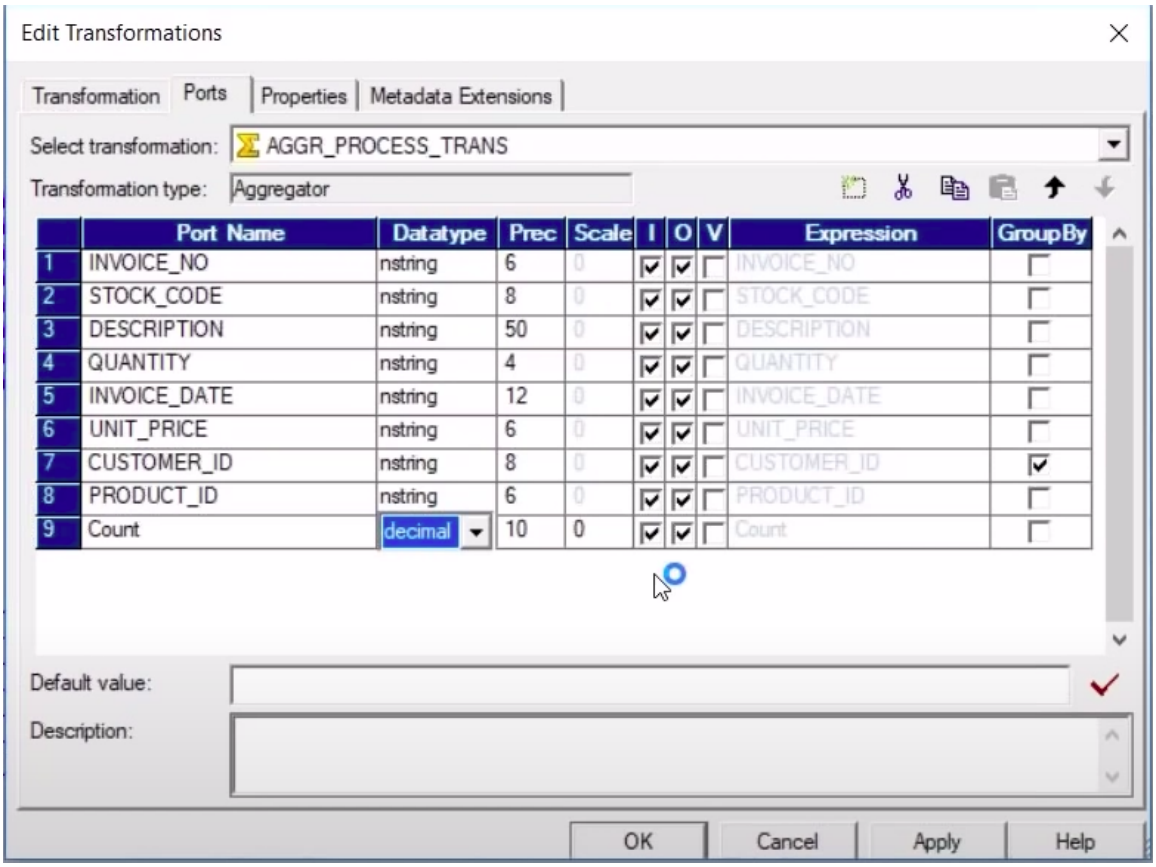

Рисунок 1.4 — Окно изменения данных в платформе Informatica PowerCenter

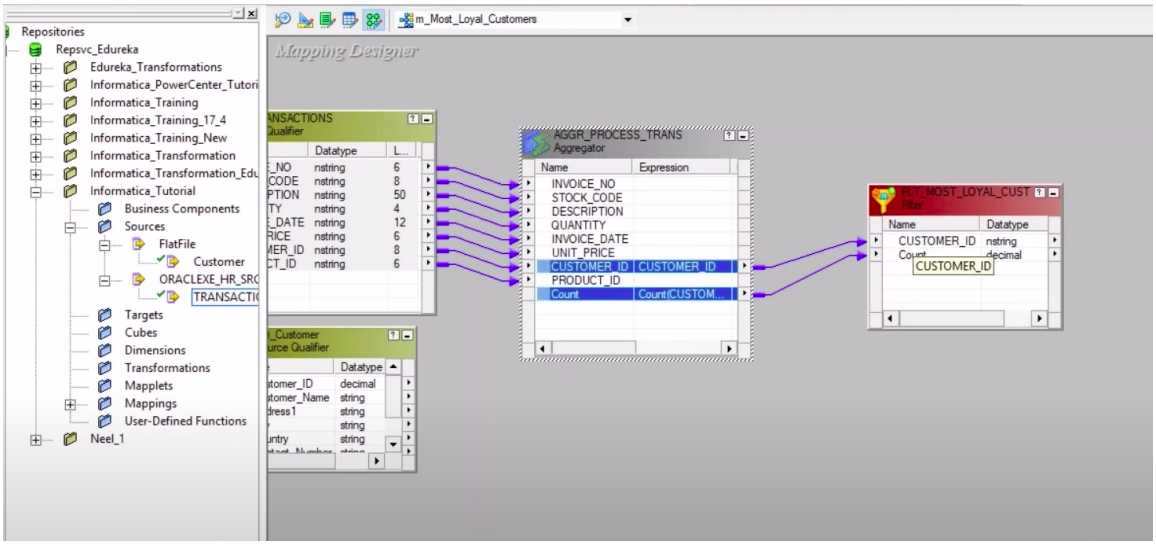

Рисунок 1.5 — Холст в платформе Informatica PowerCenter

1.2.3 Apache NiFi

Apache NiFi – это программный проект от Apache Software Foundation, предоставляющий возможности управления потоками данных из разнообразных источников в режиме реального времени с использованием графического интерфейса [7]. В NiFi используется веб-интерфейс для создания потоков данных. На рисунке 1.6 показан холст, на котором создан поток данных. NiFi позволяет парсить данные регулярными выражениями, выполнять по ним sql, фильтровать и добавлять поля, конвертировать один формат данных в другой [8].

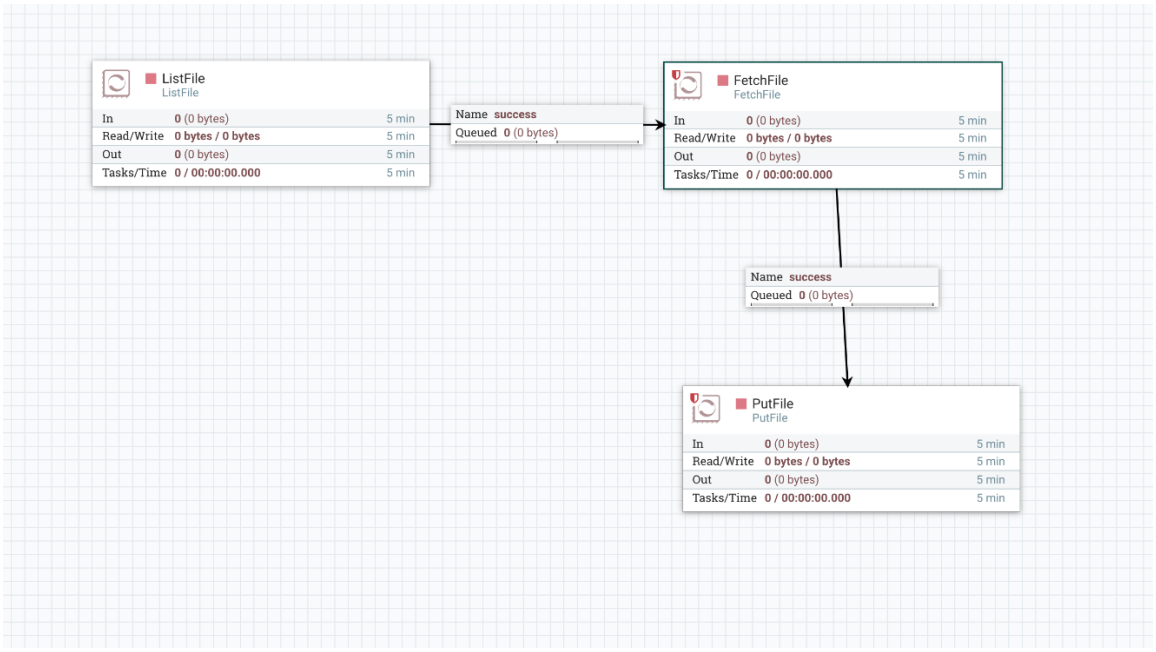

Рисунок 1.6 — Пример потока данных в программном продукте Apache NiFi

1.2.4 Сравнение существующих решений

На рисунке 1.7 приведено сравнение существующих решений.

ETL-инструмент/ критерий	Свободный доступ к ПО	Степень поддержки источников данных	Возможность мониторинга данных от начала до конца	Управление пользователями и ролями
SAS Data Integration Studio	Бесплатная демо версия	Практически все основные СУБД	Нет	Нет
Informatica PowerCenter	Бесплатная демо версия	Не имеет значения, откуда исходные данные	Нет	Да
Apache NiFi	Да	Широкий спектр источников	Да	Да

Рисунок 1.7 — Сравнение существующих решений

1.3 План выполнения запроса PostgreSQL

PostgreSQL получает на вход набор sql команд и обрабатывает каждую команду команду в четыре шага.

а) Синтаксический анализ sql-запроса и последующее создание дерева синтаксического анализа.

б) Оптимизация запроса.

в) Создание плана.

г) Выполнение.

В PostgreSQL функция exec_simple_query выполняет представленные шаги. Рассмотрим более подробно данные шаги.

1.3.1 Построение дерева синтаксического анализа

Дерево синтаксического анализа – набор хранимых в памяти взаимосвязанных типов. В случаи PostgreSQL этот набор хранимых структур данных языка Си [9].

С помощью синтаксического анализа PostgreSQL конвертирует sql-запрос во внутреннюю структуру данных, с которой в дальнейшем он может работать.

PostgreSQL использует генератор синтаксического анализа Bison [10] для синтаксического анализа.

Во время процесса сборки PostgreSQL Bison генерирует код парсера на основании ряда грамматических правил. Данный код работает внутри PostgreSQL, когда ему отправляются sql команды. Далее каждое грамматическое правило вызывается,

когда сгенерированный парсер находит соответствующий паттерн или синтаксис в строке sql и вставляет новую структуру памяти Си в дерево синтаксического анализа [11].

Для примера рассмотрим следующий запрос

Листинг 1.1 — Пример sql-запроса над таблицей foo

```
1 SELECT * FROM foo where bar = 42 ORDER BY id DESC LIMIT 23;
```

Данный запрос преобразуется в следующее дерево синтаксического анализа.

Листинг 1.2 — Дерево синтаксического анализа для sql-запроса над таблицей foo

```
1  (
2     {SELECT
3     :distinctClause <>
4     :intoClause <>
5     :targetList (
6        {RESTARGET
7        :name <>
8        :indirection <>
9        :val
10          {COLUMNREF
11          :fields (
12             {A_STAR
13             }
14          )
15          :location 7
16          }
17       :location 7
18       }
19    )
20    :fromClause (
21       {RANGEVAR
22       :schemaname <>
23       :relname foo
24       :inhOpt 2
25       :relpersistence p
26       :alias <>
27       :location 14
28       }
29    )
```

```
30      :whereClause
31         {AEXPR
32         :name ("=")
33         :lexpr
34            {COLUMNREF
35            :fields ("bar")
36            :location 24
37            }
38         :rexpr
39            {A_CONST
40            :val 42
41            :location 30
42            }
43         :location 28
44         }
45      :groupClause <>
46      :havingClause <>
47      :windowClause <>
48      :valuesLists <>
49      :sortClause (
50         {SORTBY
51         :node
52            {COLUMNREF
53            :fields ("id")
54            :location 42
55            }
56         :sortby_dir 2
57         :sortby_nulls 0
58         :useOp <>
59         :location -1
60         }
61      )
62      :limitOffset <>
63      :limitCount
64         {A_CONST
65         :val 23
66         :location 56
67         }
68      :lockingClause <>
```

```
69      :withClause <>
70      :op 0
71      :all false
72      :larg <>
73      :rarg <>
74      }
75  )
```

Рассмотрим еще один запрос (листинг 1.3). На рисунке 1.8 проиллюстрировано дерево синтаксического анализа, которое создал PostgreSQL из запроса представленного в листинге 1.3.

Листинг 1.3 — Пример sql-запроса над таблицей users

```
1 SELECT * FROM users
2 WHERE name = 'Captian Nemo'
3 ORDER BY id ASC
4 LIMIT 1;
```

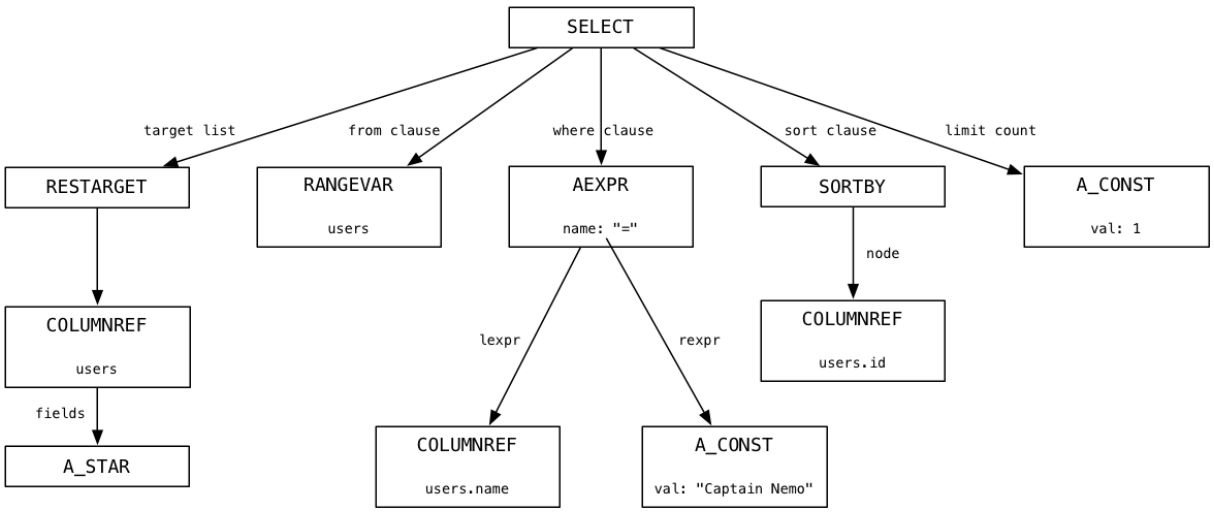

Рисунок 1.8 — Дерево синтаксического анализа для sql-запроса из таблицы users

1.3.2 Оптимизация

После того, как PostgreSQL создал дерево синтаксического анализа из переданного ему sql-запроса, он преобразовывает его, применяя оптимизации, в другое дерево, используя другой набор узлов. Полученное дерево называется деревом запроса. Процесс оптимизации применяет ряд сложных алгоритмов и эвристик в попытке оптимизировать и упростить sql-запрос.

Для приведенного выше запроса (листинг 1.3) PostgreSQL преобразовал дерево синтаксического анализа в дерево, представленное на рисунке 1.9.

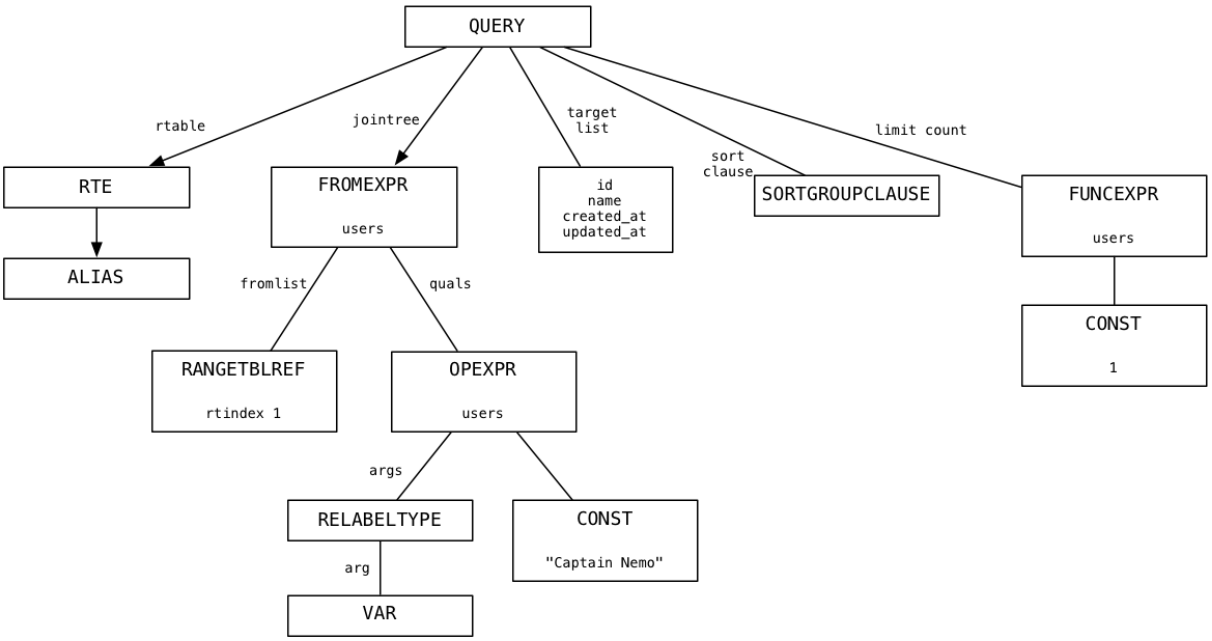

Рисунок 1.9 — Дерево запроса для sql-запроса из таблицы users

1.3.3 Создание плана

Перед тем, как начать выполнять запрос, PostgreSQL создает план. Данный процесс включает в себя создание третьего дерева узлов, которые представляют собой список инструкций для PostgreSQL. Для приведенного выше запроса (листинг 1.3) PostgreSQL построил дерево плана, которое показано на рисунке 1.10.

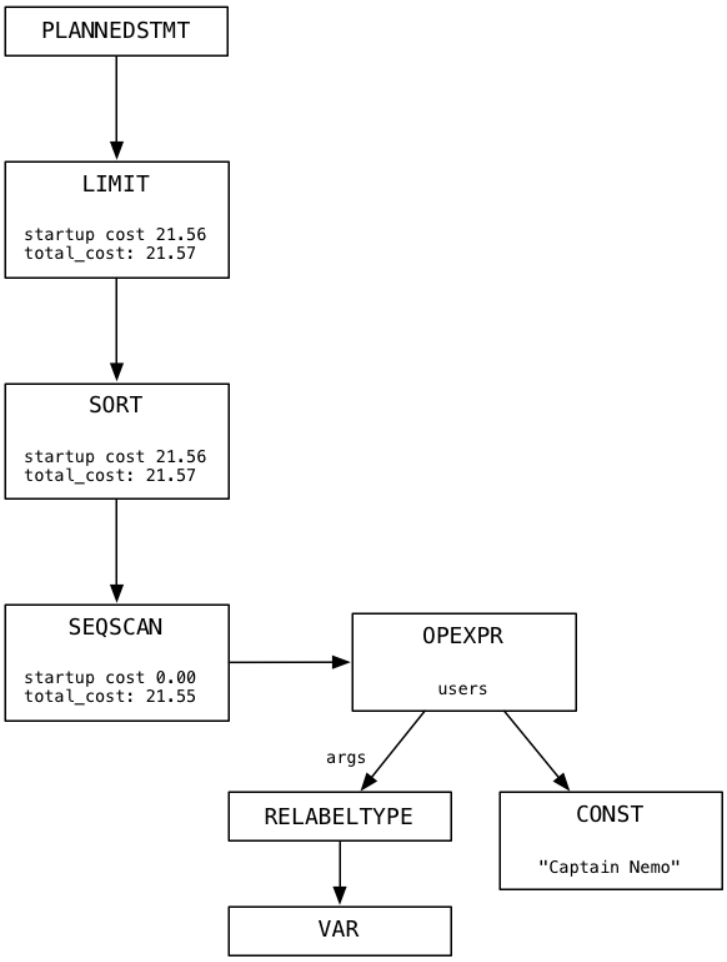

Рисунок 1.10 — План для sql-запроса из таблицы users

1.3.4 Выполнение

На шаге выполнения запроса PostgreSQL уже преобразовал переданный ему в с самом начале sql-запрос в синтаксическое дерево анализа, оптимизировал и построил план запроса. PostgreSQL имеет план, которому будет следовать, чтобы получить результат. PostgreSQL для каждого нижележащего узла берет данные и возвращает их в качестве исходных данных для узла выше. Таким образом данный итеративный процесс повторяется, пока что не будут получены данные из самого верхнего узла. Это и будет являться результатом sql-запроса.

Заключение

В рамках выполнения работы решены следующие задачи.

— Описана предметная область.
— Рассмотрены существующие решения.
— Проанализированы существующие решения
— Описан план выполнения PostgreSQL запроса.

Список использованных источников

1. *Соломатин Д.И. Копытин А.В., Другалев А.И.* Основы синтаксического разбора, построение синтаксических анализаторов / Другалев А.И. Соломатин Д.И., Копытин А.В. — Министерство образования и науки РФ федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «воронежский государственный университет», 2014. — P. 57.

2. Анализ и переписывание в дерево запросов postgresql. — 2011. https://russianblogs.com/article/67421448978/.

3. postgresql. — 1996. https://www.postgresql.org/.

4. Основные структуры данных. — 2018. — 9. https://habr.com/ru/post/422259/.

5. SAS Data Integration Studio. — SAS Institute, 1976. https://support.sas.com/en/software/data-integration-studio-support.html.

6. Informatica PowerCenter. — Data Integration, 1993. https://dis-group.ru/technologies/data-integration/powercenter/.

7. Apache NiFi. — Apache Software Foundation, 2006. https://nifi.apache.org/.

8. Apache NiFi: краткий обзор возможностей. — 2018. — 12. https://habr.com/ru/company/rostelecom/blog/432166/.

9. *Керниган Брайан У., Ритчи Деннис М.* Язык программирования C / Ритчи Деннис М. Керниган Брайан У. — Вильямс, 2019. — P. 288.

10. GNU Bison. — Free Software Foundation, 2014. https://www.gnu.org/software/bison/.

11. *Е., Андросова Е.* Синтаксически управляемый анализатор выражений / Андросова Е. Е. — Тринадцатая научная конференция молодых исследователей, 2010. — P. 30.