С.В. Минухин, В.В. Федько

РЕАЛИЗАЦИЯ ОПЕРАЦИЙ С РАСПРЕДЕЛЕННЫМИ БАЗАМИ ДАННЫХ СРЕДСТВАМИ POLYBASE

Аннотация. *Рассмотрены возможности доступа к данным значительных размеров, которые размещены на различных серверах и управляются СУБД разного типа. В качестве технологии хранения используется кластер Hadoop, а в качестве интерфейса с пользователем – технология PolyBase. Последняя предоставляет язык SQL для выполнения операций с данными, которые хранятся в Hadoop. Описаны основы функционирования PolyBase как компоненты SQL Server и особенности ее настройки на работу с Hadoop. Продемонстрировано на конкретных примерах выполнение основных операций с данными Hadoop посредством PolyBase, дана оценка перспективности применения этой технологии в распределенных вычислениях.*
Ключевые слова: *Big Data, Hadoop, PolyBase, SQL Server, T-SQL, Connection string, распределенные данные, распределенные вычисления, масштабирование, масштабируемые группы, MapReduce, YARN, HDFS, внешний источник данных, формат внешнего файла, внешняя таблица.*

Abstract. *Considered are the possibilities of access to data of considerable size, which are placed on different servers and managed by a different type of DBMS. As a storage technology, the Hadoop cluster is used, and PolyBase technology as the interface with the user. The latter provides the SQL language for performing operations with data stored in Hadoop. The bases of functioning of PolyBase as components of SQL Server and features of its adjustment for work with Hadoop are described. Demonstrated on specific examples of the implementation of basic operations with Hadoop data through PolyBase, an assessment of the promise of using this technology in distributed computing is given.*
Keywords: *Big Data, Hadoop, PolyBase, SQL Server, T-SQL, Connection string, distributed data, distributed computing, scaling, scalable groups, MapReduce, YARN, HDFS, external data source, external file format, external table.*

Введение и постановка задачи

Когда данные хранятся в различных базах данных одного типа, для обмена данными достаточно в коде указать библиотеку провайдера данных и для каждой базы данных задать строку соединения. Строки различаются лишь местом размещения базы данных. Например, для баз данных SQL Server

поддержка провайдера данных осуществляется указанием пространства имен System.Data.SqlClient, а строки соединения в простейшем случае имеют следующий вид [1, 2]:

```
Data Source=myServerAddress;Initial Catalog=myDataBase; User Id=myUserName; Password=myPassword;
```

Если данные хранятся в реляционных базах данных различного типа, ситуация незначительно усложняется – в коде указывают пространства имен для соответствующих провайдеров данных. Ниже представлено подключение пространств имен следующих провайдеров данных: SQL Server, Oracle и MySQL.

```
using System.Data.SqlClient;
using System.Data.OracleClient;
using MySql.Data.MySqlClient;
```

Такая комбинация часто встречается в растущих компаниях, когда первоначально данные в главном офисе хранились в Oracle, затем при появлении филиалов, для их данных закуплен SQL Server и, наконец, при выходе компании в Интернет, соответствующие данные стали размещать на серверах MySQL. Соответствующие строки соединения имеют следующий вид [1]:

```
Data Source=mySQLServerAddress; Initial Catalog=myDataBase; User Id=myUserName; Password=myPassword;

Provider=OraOLEDB.Oracle; Data Source=myOracleDB; User Id=myUsername;
Password=myPassword;

Server=myMySQLServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword;
```

В обоих рассмотренных выше случаях для операций с базами данных используется один язык SQL или его обертка – технология LINQ.

Ситуация усложняется, когда благодаря интенсивному применению Интернет-технологий понадобилось использование баз данных NoSQL. Здесь, помимо библиотек для специфических провайдеров данных и строк соединения, требуется отказаться от использования языка SQL. В качестве

средства для выполнения CRUD-операций с базой данных используют различные методы. Например, для одной из самых распространенных баз данных NoSQL MongoDB используют соответствующий драйвер (пространство имен MongoDB.Driver) [3]. Строка соединения имеет следующий синтаксис:

```
mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN
[:portN]]][/[database][?options]
```

В простейшем случае она имеет вид:

```
mongodb://localhost:27017
```

При работе с базой данных MongoDB, например, вместо оператора Select языка SQL используют метод find со своими параметрами.

Такое разнообразие средств, которые нужно одновременно использовать в приложениях, часто приводит к путанице и замедляет процесс разработки. Желательно иметь универсальные средства, которые давали бы возможность одинаковым образом обрабатывать данные, которые хранятся в базах данных различного типа.

Основная часть

Для решения проблемы обработки данных, хранящихся во множестве баз данных различного типа, разработана технология Hadoop. Она предназначена для обработки больших объемов данных (от нескольких терабайт до нескольких петабайт), которые хранятся в большом количестве файлов. Размер каждого из них достигает порядка 10 гигабайт и больше и расположены они на нескольких компьютерах (их может быть несколько десятков и более) [4].

Чаще всего Hadoop используется в качестве хранилища данных гибридного типа. В нем справочные данные хранятся в реляционной базе данных, а данные о повседневной работе компании загружаются в NoSQL-хранилище в преобразованном и очищенном виде из веб-сайтов. (рис. 1).

Такие хранилища могут содержать петабайты данных и легко масштабируются при применении технологии Hadoop. То обстоятельство, что Hadoop не поддерживает транзакции, не является в данном случае препятствием. Для ведения хранилища данных достаточно двух операций – поблокового добавления и чтения больших массивов данных. Часто файлы оперативных данных хранят слабоструктурированную информацию типа NoSQL. За систему хранения в технологии Hadoop отвечает файловая система

HDFS (от англ. Hadoop Distributed File System). В ней распределены блоки файлов между узлами кластера, состоящего из сотен и тысяч компьютеров, образующих вычислительный кластер.

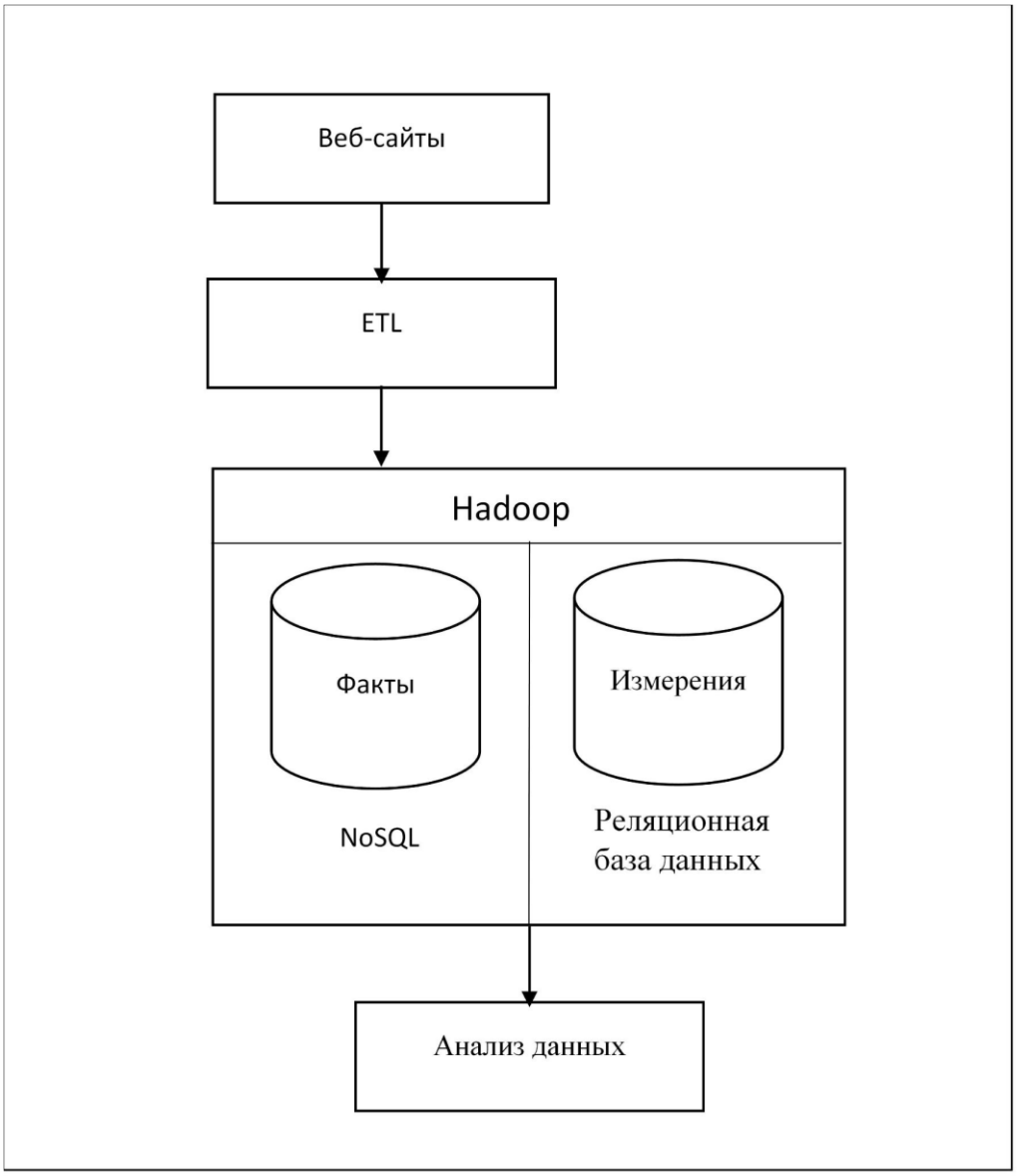

Рис. 1. Система анализа данных на основе Hadoop

Данные хранятся в узлах Data Node, которые представляют собой блоки. Обычно устанавливают размер блока в 64, 128 или 256 Мбайт. Если размер файла превышает выбранный размер блока, не поместившиеся данные попадают в следующий блок. В управляющем узле NameNode запоминаются адреса файлов в древовидной системе хранения данных и метаданные файлов и каталогов (рис. 2).

Чтение данных осуществляется также через узел NameNode. В нем выбирается ближайший блок, в котором хранятся нужные данные и затем из него выбираются данные. Поскольку данные в узлах дублируются,

одновременно с ними могут работать несколько приложений, что увеличивает масштабируемость системы.

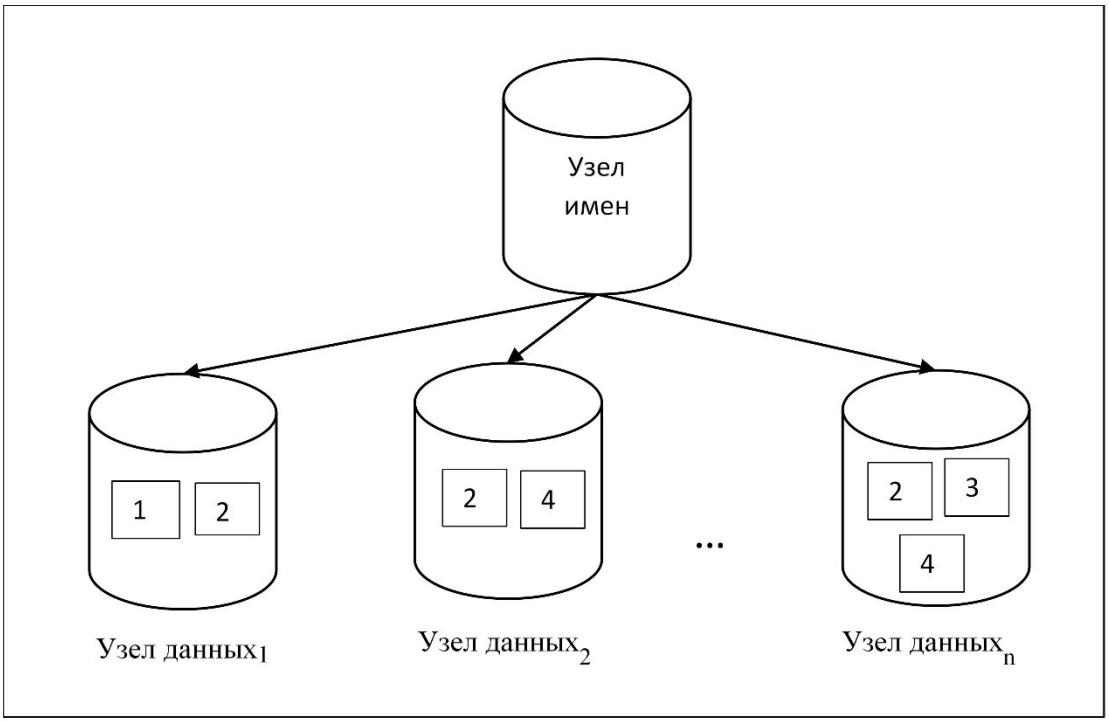

Рис. 2. Структура HDFS

Программный Framework MapReduce используется для организации распределенных вычислений. Они выполняются в два этапа. Сначала выбираются нужные данные (этап Map), а затем производятся вычисления над отобранными данными (этап агрегации данных Reduce). Поскольку данные разбросаны по разным блокам на различных серверах, операции первого этапа можно выполнять параллельно. За счет этого значительно сокращается время выполнения всего запроса (рис. 3).

Результатом этапа Map является множество пар ключ-значение. Перед тем, как они будут переданы на этап Reduce, эти пары группируются по совпадающим значением ключа, а если необходимо, то и сортировка. Такая группировка в терминах Hadoop называется Shuffle. Она является скрытым от пользователя этапом обработки.

Поскольку этап Map одновременно может выполняться на сотнях и большем количестве компьютеров, чтобы сократить количество передаваемых на следующий этап данных, на каждом компьютере, если возможно, производят предварительные вычисления (например, вычисление сумм).

Обработка данных может состоять из цепочки нескольких процессов MapReduce. Когда количество элементов в цепочке становится большим и количество компьютеров в кластере также достигает значительного числа,

возникают проблемы с масштабируемостью описанной выше архитектуры. Например, по результатам исследований Yahoo! ее пределы достигаются на кластере из 5000 узлов и 40 000 одновременно выполняющихся задач [4]. В качестве выхода из создавшейся ситуации был предложен Framework YARN (от англ. Yet Another Resource Negotiator — «ещё один ресурсный посредник»). Он предназначен для управления ресурсами кластеров и планирования заданий. YARN организует параллельное выполнение задач на одном кластере. При этом обеспечивает изоляцию этих задач (отдаленный аналог транзакций в реляционных базах данных). Распределенное приложение, работающее с YARN, должно иметь выделенный класс управления приложением, который отвечает за синхронизацию заданий. Они используют ресурсы, выделенные планировщиком ресурсов YARN. В целом структура Hadoop представлена на рис. 4.

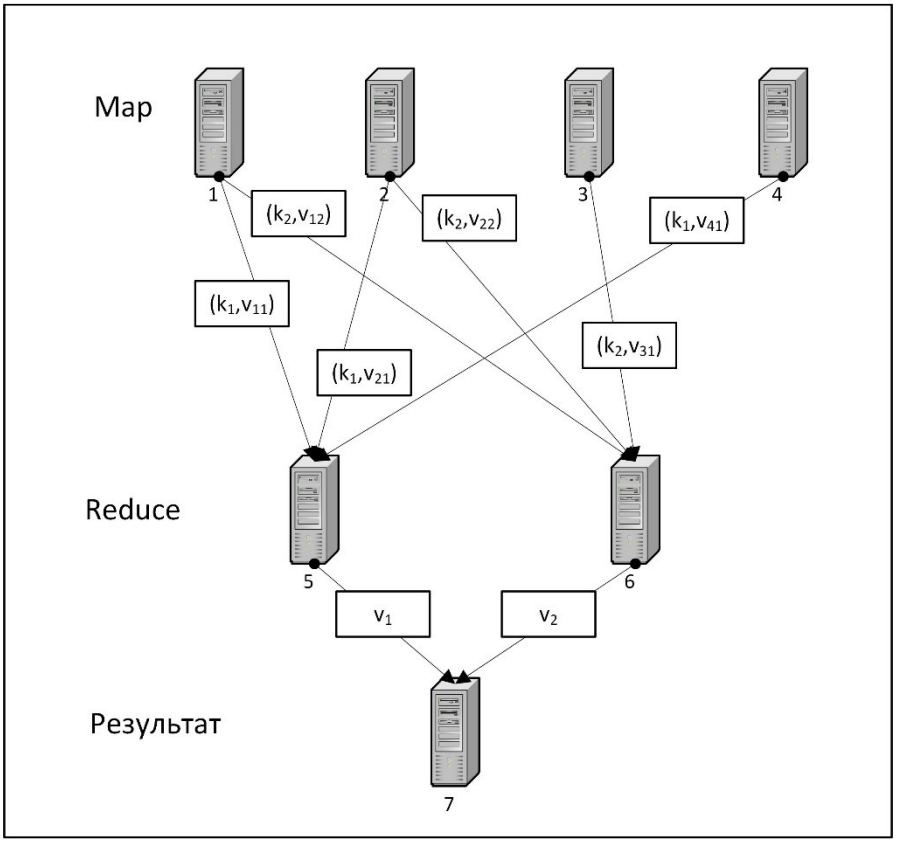

Рис. 3. Схема процесса MapReduce

При разработке Hadoop одной из важнейших целей была поставлена масштабируемость. Причем требовалась не вертикальная масштабируемость, когда увеличение мощности системы обработки данных достигается за счет увеличения мощности входящих в кластер компьютеров, а горизонтальная. В последней реакция на увеличение размера данных и количества пользователей реализуется за счет подключения дополнительного количества недорогих

компьютеров. Аналогичная проблема возникала и в базах данных NoSQL [3]. Там она разрешилась за счет шардинга и репликации баз данных. Подобные подходы были использованы в Hadoop. Здесь шардингу соответствует разделение файлов на блоки и, конечно, репликация данных, которая обеспечивает надежность (отказоустойчивость) обработки данных в условиях отсутствия транзакций. Масштабируемость еще больше улучшилась при добавлении фреймворка YARN в Hadoop. При этом удалось значительно превысить ограничение с 4 тыс. узлов при использовании 10 MapReduce-заданий на узел [4].

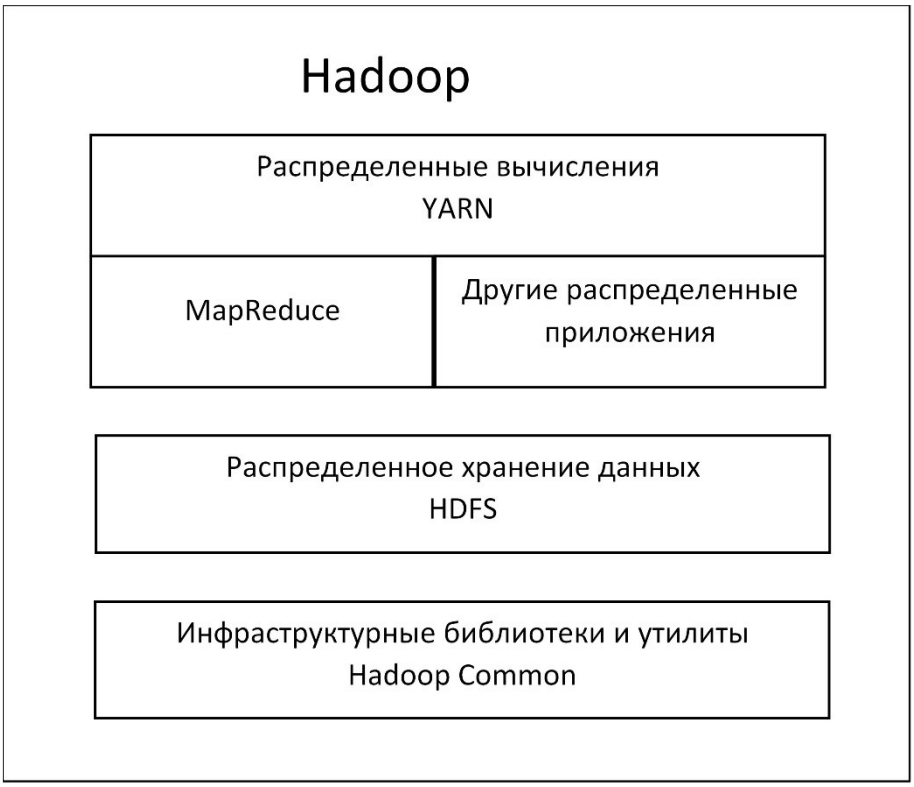

Рис. 4. Структура Hadoop

Задание запросов к хранилищам данных на уровне MapReduce часто является непростой задачей, которая требует профессионального владения соответствующими средствами. Для понижения порога использования технологии Hadoop были разработаны надстройки. Работы здесь велись в двух направлениях. В качестве интерфейса конечного пользователя предлагался язык программирования высокого уровня и SQL-подобный язык. Представителем первого направления считается система Pig [4], а второго – система Hive [4]. Обе системы помимо самого языка содержат трансляторы, преобразующие задание на соответствующем языке в последовательность команд для MapReduce. При создании этих систем они ориентировались на конечного пользователя, которым мог быть аналитик или Data Scientist. Но

практика показала, что этими системами активно пользуются только специалисты уровня разработчика.

Тенденция упрощения интерфейса пользователя с Hadoop была подхвачена многими производителями [4]. В частности, в Microsoft она реализовалась как технология PolyBase [5]. Она позволяет работать с Hadoop средствами языка T-SQL. Технология PolyBase представлена как новый компонент в SQL Server 2016 в качестве моста между Hadoop и SQL Server. PolyBase позволяет выполнять следующие операции:

1) производить запросы на получение данных из SQL Server к Hadoop;

2) импортировать данные из Hadoop в SQL Server;

3) экспортировать данные из SQL Server в Hadoop.

Две последние операции используются в качества средства ETL при формировании хранилища данных, а первая – непосредственно для анализа данных в гибридном хранилище. Экспорт данных также используется для пересылки в киоски данных для их анализа и построения отчетов [5]. Место PolyBase в схеме обработки данных представлено на рис. 5.

Использование PolyBase как моста между Hadoop и SQL Server позволяет легко применять все инструменты анализа данных, которые разработаны Microsoft, к данным, хранящимся в Hadoop. К ним относятся как более приближенные к конечному пользователю Excel и Power BI с их надстройками Power Pivot и Power Query, так и более профессиональные службы Integration Services, Analysis Services, Reporting Services и Machine Learning Services, ориентированные на разработчиков [5]. Помимо вычислительных средств Microsoft здесь могут использоваться эффективные распределенные вычисления Hadoop, что позволяет применять механизмы распараллеливания MapReduce и YARN в нагруженных вычислениях.

Для использования стека Hadoop – PolyBase вначале разворачивается и настраивается Hadoop. Для упрощения начала работы с Hadoop можно воспользоваться готовыми дистрибутивами, которые устанавливаются на виртуальную машину. Наиболее удачные дистрибутивы Hadoop разработаны следующими компаниями: Cloudera, Hortonworks, MapR, IBM и Pivotal [5]. Дистрибутивы можно найти на сайтах этих компаний. Если предполагается развернуть Hadoop в облаках, следует обратить внимание на Elastic MapReduce на платформе Amazon Web Services или HDInsight на платформе Microsoft Azure.

Отметим, что из перечисленных выше дистрибутивов пяти компаний в настоящее время PolyBase устанавливается только на HDP (Hortonworks Data Platform) и CDH (Cloudera Distributed Hadoop) [5].

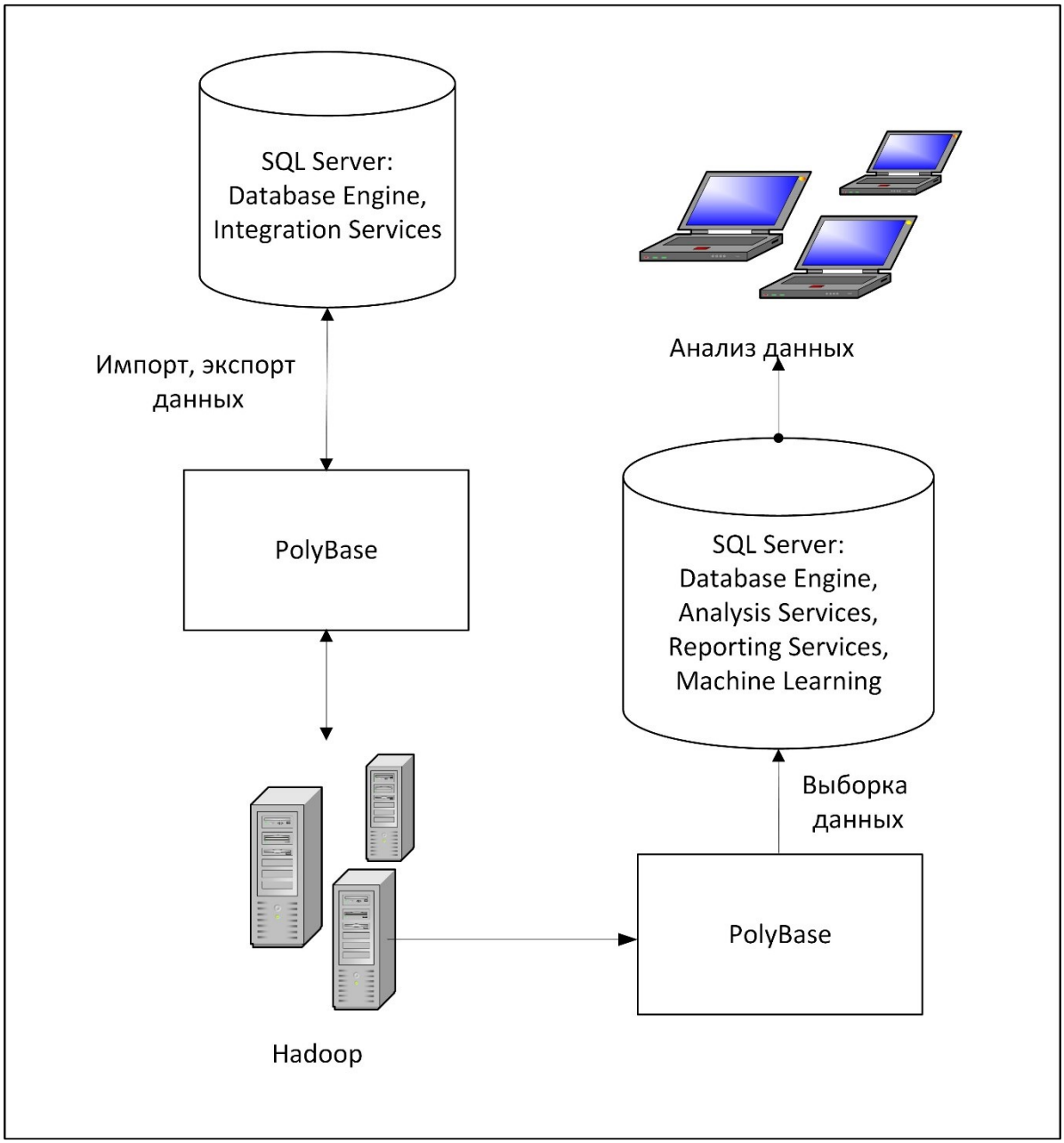

Рис. 5. Обработка данных с использованием PolyBase

После этого устанавливается SQL Server версии 2016 или выше [5]. При этом следует выбрать выпуск Enterprise или Developer. Последний из них бесплатный и наиболее приспособлен для освоения технологии.

Компонент PolyBase разворачивается во время установки SQL Server путем прямого выбора или добавляется позже при изменениях конфигурации SQL Server.

Подключение PolyBase к Hadoop как источнику данных производится сложнее, чем описанное выше задание строки соединения в ADO.NET. Здесь

подключение состоит из двух этапов – конфигурации сервера и создания внешнего источника данных. Изменение конфигурации сервера реализуется последовательным выполнением операторов sp_configure и RECONFIGURE. Первый из них представляет собой утилиту, которая имеет следующий формат:

```
sp_configure [ @configname = ] 'hadoop connectivity',
        [ @configvalue = ] { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 }
```

Значение первого параметра указывает на то, что подключение производится к Hadoop, а второй представляет собой целое число, значение которого определяет тип дистрибутива.

Второй оператор RECONFIGURE не имеет параметров. Он обновляет текущее значение, которое задается вторым параметром утилиты sp_configure (аналогично команде COMMIT в транзакциях).

Для повышения быстродействия вычислений добавляют технологии YARN и MapReduce. Первый компонент подключается путем присваивания значения ключа конфигурации yarn.application.classpath одноименному свойству SQL Server в xml-файле yarn.site.xml. Чтобы подключить компонент MapReduce, добавляют параметры конфигурации mapreduce.application.class-path в конец файла yarn.site.xml.

Для защищенного обмена данными используется SSL-протокол. Он обеспечивает аутентификацию и передачу зашифрованных данных между узлами кластера Hadoop. По умолчанию устанавливается конфигурация authenticate (проверка подлинности). Для более сильной защиты задается значение integrity (целостность) для свойства hadoop.rpc.protection. Наивысший уровень защиты обеспечивается, когда также шифруются данные, которые передаются между клиентом и сервером. Он достигается при значении privacy (конфиденциальность) этого свойства [5].

Для повышения эластичности реакции на увеличение объемов данных в кластере Hadoop к нему одновременно подключают несколько экземпляров PolyBase, расположенных на отдельных компьютерах, которые входят в один домен. Такое множество согласованно работающих экземпляров PolyBase называется масштабируемой группой PolyBase [5]. В нем один экземпляр считается головным узлом, а остальные – вычислительными (рис. 6).

Узлы PolyBase взаимодействуют следующим образом. SQL Server получает запрос и, если для его выполнения требуются данные из кластера Hadoop, передает задание на их поиск головному узлу. Находящееся в нем

ядро PolyBase распределяет работу между вычислительными узлами, распараллеливая таким образом выполнение запроса.

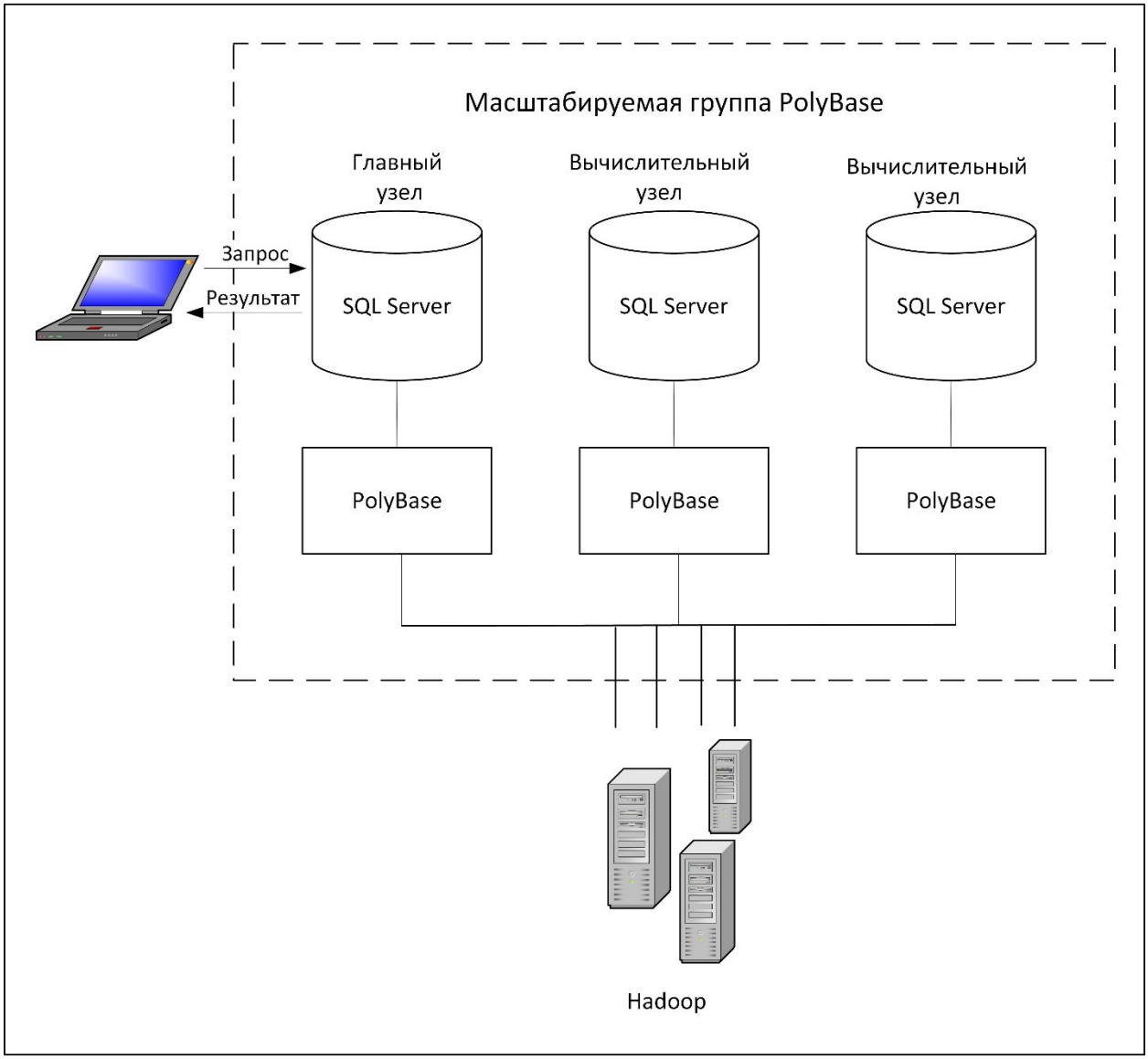

Рис. 6. Масштабируемая группа PolyBase

Для непосредственного использования PolyBase необходимо создать следующие объекты:

внешний источник данных;

формат внешнего файла;

внешняя таблица.

Внешний источник данных определяет расположение кластера Hadoop и, возможно, учетную информацию для проверки подлинности на внешнем источнике данных. Скрипт на создание источника данных имеет следующий синтаксис:

```
CREATE EXTERNAL DATA SOURCE data_source_name
   WITH (
      TYPE = HADOOP,
      LOCATION = 'hdfs://NameNode_URI[:port]'
      [, RESOURCE_MANAGER_LOCATION =
'ResourceManager_URI[:port]' ]
      [, CREDENTIAL = credential_name ]
   )
```

В простейшем случае источник данных HadoopCluster1 на кластере, который подключен к порту 8050 компьютера с IP-адресом 10.10.10.10 создается следующим скриптом:

```
CREATE EXTERNAL DATA SOURCE HadoopCluster1
WITH (
   TYPE = HADOOP,
   LOCATION = 'hdfs://10.10.10.10:8050'
)
```

В формате внешнего файла указывают тип файлов в определенном выше источнике данных и его особенности (признак конца поля для текстового файла, разделитель строк в текстовом файле, формат даты, метод сжатия файлов и т. п.). Его синтаксис имеет следующий вид:

```
CREATE EXTERNAL FILE FORMAT file_format_name
WITH (
   FORMAT_TYPE = external_file_type
   [ , FORMAT_OPTIONS ( <format_options> [ ,...n  ] ) ]  ]
    [ , DATA_COMPRESSION = data_compression_method ])

<format_options> ::=
{
   FIELD_TERMINATOR = field_terminator
   | STRING_DELIMITER = string_delimiter
   | DATE_FORMAT = datetime_format
   | USE_TYPE_DEFAULT = { TRUE | FALSE }
   | Encoding = {'UTF8' | 'UTF16'}
}
```

При этом поддерживаются следующие типы файлов: Delimited text, Hive RCFile, Hive ORC, Parquet.

Например, если в кластере Hadoop содержатся Parquet-файлы сжатые по методу Gzip, то описание такого формата имеет следующий вид:

```
CREATE EXTERNAL FILE FORMAT ParquetGzip1
WITH (
   FORMAT_TYPE = PARQUET,
   FORMAT_OPTIONS (DATE_FORMAT = 'dd/MM/yyyy'),
   DATA_COMPRESSION = 'org.apache.hadoop.io.compress.GzipCodec'
   );
```

Внешняя таблица используется для отображения в PolyBase объекта хранения данных из кластера Hadoop. В частности, таким объектом может быть таблица реляционной базы данных или коллекция документоориентированой базы данных, или их представление. Создание внешней таблицы данных имеет следующий синтаксис:

```
CREATE EXTERNAL TABLE [ database_name.[ schema_name ].|
schema_name. ] table_name
   ( <column_definition> [ ,...n ] )
   WITH (
      LOCATION = 'folder_or_filepath',
      DATA_SOURCE = external_data_source_name,
      FILE_FORMAT = external_file_format_name
      [ , <reject_options> [ ,...n ] ]
   )
[;]

<reject_options> ::=
{
   | REJECT_TYPE = value | percentage
   | REJECT_VALUE = reject_value
   | REJECT_SAMPLE_VALUE = reject_sample_value
}
```

При описании внешней таблицы количество ее столбцов и их типы должны совпадать с соответствующими параметрами схемы таблицы внутри кластера Hadoop. При несовпадении выдается сообщение об ошибке. Выдачу сообщений можно отложить, задав в параметре reject_options допустимое

количество «сбойных» строк или их процент. Такой подход важен в случае, когда данные из кластера Hadoop содержатся в документоориентированой базе данных, особенностью которых является гибкая схема коллекций.

Например, для получения данных о продажах из таблицы фактов factSales, которая хранится в кластере Hadoop (источник данных HadoopCluster1) и имеет формат внешнего файла ParquetGzip1, создают внешнюю таблицу HDP_factSales с помощью следующего скрипта:

```
CREATE EXTERNAL TABLE HDP_factSales
(
   [DateKey] INT,
   [ManufacturerKey] INT,
   [ProductKey] INT,
   [Quantity] SMALLINT,
   [Cost] MONEY
)
WITH (
   LOCATION = '/apps/hive/DW/factSales',
   DATA_SOURCE = HadoopCluster1,
   FILE_FORMAT = ParquetGzip1,
   REJECT_TYPE = value,
   REJECT_VALUE=0
)
```

После создания внешней таблицы она не заполняется данными. Такая таблица выполняет только роль ссылки на данные, которые хранятся в кластере Hadoop. Но с ней можно выполнять различные операции – Insert, Select, Join и т. п. То есть, по своим функциональным возможностям она близка к представлению в реляционной базе данных.

Например, для вывода данных о стоимости продаж товаров по месяцам можно использовать следующий скрипт:

```
SELECT d.Year*100+d.MonthNumberOfYear as YearMonth, SUM(s.Cost) as
MonthCost
   FROM DimDate as d INNER JOIN HDP_factSales as s
      ON d.DateKey = s.DateKey
   WHERE s.Cost > 200
   Group By (d.Year*100+d.MonthNumberOfYear)
```

В приведенном выше скрипте применяется соединение внешней таблицы HDP_factSales с локальной DimDate и группировка данных по месяцам, а также фильтрация данных по полю Cost внешней таблицы. Последние две операции выполняются с подключением заданий MapReduce, что значительно ускоряет выполнение запроса.

Импорт данных в локальную таблицу продемонстрируем на примере передачи данных о продажах за 2018 год из внешней таблицы HDP_factSales в локальную таблицу factSales2018. Он производится следующим скриптом:

```
SELECT *
        INTO factSales2018
        FROM  HDP_factSales
        WHERE HDP_factSales.DateKey BETWEEN 20180101 AND 20181231
```

Для архивирования устаревших данных за 2016 год производится экспорт из локальной таблицы factSales во внешнюю таблицу HDP_factSales2016. Он описывается следующим скриптом:

```
-- Создание внешней таблицы, в которую будут переданы данные
CREATE EXTERNAL TABLE [HDP_factSales2016]
(
   [DateKey] INT,
   [ManufacturerKey] INT,
   [ProductKey] INT,
   [Quantity] SMALLINT,
   [Cost] MONEY
)
WITH (
   LOCATION='/OldData/2016/SalesData',
   DATA_SOURCE = HadoopCluster2,
   FILE_FORMAT = TextFileFormat,
   REJECT_TYPE = VALUE,
   REJECT_VALUE = 0
);
-- Экспорт данных
INSERT INTO HDP_factSales2016
SELECT *
        FROM  factSales
        WHERE factSales.DateKey BETWEEN 20160101 AND 20161231
```

Таким образом, используя язык T-SQL можно читать данные из Hadoop и объединять их с данными, которые хранятся в локальной базе данных SQL Server для дальнейшего бизнес-анализа, а также импортировать и экспортировать данные между локальной базой данных и хранилищем Hadoop.

Заключение

Рассмотренная технология работы с распределенными базами данных позволяет обрабатывать значительные объемы данных в приемлемое время за счет средств двухэтапного распараллеливания вычислений, которые предоставляют технологии Hadoop и PolyBase. Последняя технология обеспечивает также удобный интерфейс пользователя через посредство языка T-SQL. Технология PolyBase включена уже в два выпуска SQL Server (2016 и 2017), но до сих пор здесь не поддерживаются операции Insert, Delete и Update, а также не разработана технология подсоединения к хранилищу данных на основе аппарата Connection string. Несмотря на эти недостатки развивающаяся технология PolyBase представляется перспективной для работы с большими данными благодаря широкому использованию средств горизонтального масштабирования и языку T-SQL, а также поддержке операций записи данных большими блоками и эффективному поиску данных в значительных объемах данных. Именно эти операции чаще всего используются при работе с хранилищами данных.

Литература

1. Федько В. В. Організація баз даних та знань. / В. В. Федько, О. В. Тарасов, М. Ю. Лосєв. – Х. : Вид. ХНЕУ, 2013. – 200 с.

2. Федько В. В. Сучасні засоби доступу до даних. / В. В. Федько, О. В. Тарасов, М. Ю. Лосєв. – Х. : Вид. ХНЕУ ім. С. Кузнеця, 2014. – 328 с.

3. Бэнкер К. MongoDB в действии = MongoDB in Action. — ДМК Пресс, 2014. — 394 с.

4. Shvachko K. Apache Hadoop. The Scalability Update. – 2011. – Vol. 36, no. 3.

5. О новых функциях SQL Server 2016. [Электронный ресурс] Режим доступа: https://habrahabr.ru/company/microsoft/blog/302770/