Вычислительные технологии, 2022, том 27, № 3, с. 46–65. © ФИЦ ИВТ, 2022
Computational Technologies, 2022, vol. 27, no. 3, pp. 46–65. © FRC ICT, 2022
ISSN 1560-7534
eISSN 2313-691X

DOI:10.25743/ICT.2022.27.3.005

Оценка эффективности обработки больших объемов данных в реляционных и колоночных форматах

В. А. Белов, Д. Ю. Ильин, Е. В. Никульчев*
МИРЭА — Российский технологический университет, 119454, Москва, Россия
*Контактный автор: Никульчев Евгений Витальевич, e-mail: nikulchev@mail.ru

Поступила 01 апреля 2022 г., доработана 12 апреля 2022 г., принята в печать 18 апреля 2022 г.

Эффективное хранение данных — одна из важнейших задач при проектировании любой информационной системы. Рост потребностей в обработке больших объемов данных спровоцировал появление большого количества средств для их хранения. В связи с этим возникает необходимость выбора форматов хранения на этапе проектирования. Выбор форматов влияет на параметры вычислительной среды (объем, время обработки данных), а также аппаратных ресурсов.

Статья посвящена разработке методики оценки эффективности обработки больших данных в зависимости от выбора реляционного или колоночного формата. Представлено исследование двух популярных способов хранения и обработки больших данных: реляционная база данных PostgreSQL и хранение в файлах колоночного формата Apache Parquet с обработкой с помощью фреймворка Apache Hive.

Ключевые слова: большие данные, форматы хранения данных, реляционные базы данных, PostgreSQL, Apache Hive.

Цитирование: Белов В.А., Ильин Д.Ю., Никульчев Е.В. Оценка эффективности обработки больших объемов данных в реляционных и колоночных форматах. Вычислительные технологии. 2022; 27(3):46–65. DOI:10.25743/ICT.2022.27.3.005.

Введение

Решение задачи выбора формы хранения данных является одним из обязательных этапов проектирования информационной системы — необходимо проанализировать данные, которые будут сохраняться в базы данных, типы и структуры данных, а также их объем. Увеличение объема данных привело к разработке новых способов и инструментов хранения данных, что требует сравнительного анализа этих средств.

При выборе инструмента хранения данных важную роль играет экспериментальная оценка имеющихся альтернатив [1, 2]. Результаты эксперимента могут продемонстрировать такие характеристики, как занимаемый объем, время обработки данных, а также использование ресурсов аппаратного обеспечения. При этом следует исходить из того факта, что объем данных со временем будет пополняться, поэтому целесообразным становится анализ перечисленных характеристик в динамике изменения объема данных, что, в свою очередь, также становится одной из характеристик в задаче выбора одной из альтернатив.

Целью данного исследования является сравнение особенностей обработки данных в реляционной базе данных и системах обработки больших данных. В качестве реляционной базы данных выбрана PostgreSQL [3] — один из наиболее популярных продуктов в системах хранения данных [4]. Для имитации системы обработки больших данных выбрана платформа Apache Hadoop [5] с предустановленным фреймворком Apache Hive [6]. Для хранения данных использован колоночный формат файлов Apache Parquet [7].

Наиболее распространенный способ хранения данных — использование реляционных баз данных [8]. Однако рост объема данных, изменение структур данных, а также потребность в хранении информации в ее оригинальном виде привели к появлению новых способов хранения. Все это вызвало появление новой концепции "больших данных" [9]. Точного определения этого понятия не существует, оно имеет такие характеристики, как объем (volume), скорость (velocity) и многообразие (variety), называемые также "тремя V" [9]. Отсутствие четкого понимания, в какой момент данные становятся такого объема, что речь может идти о "больших данных", создает необходимость проведения не только экспертного анализа предполагаемого к использованию программного обеспечения, но и экспериментальной оценки альтернативных средств.

Для хранения и обработки информации в системах больших данных широко применяется платформа Apache Hadoop [5], в которой используется файловая система Apache HDFS [5], хранящая данные в формате Apache Parquet. Сравнение разных способов хранения данных является одним из популярных направлений исследований.

Для работы с данными в формате Apache Parquet существует множество инструментов, например Apache Hive, представляющий собой реализацию алгоритма MapReduce [6] для чтения HDFS-данных. Также Apache Hive предлагает SQL-подобный язык запросов HiveQL.

Изначально большинство исследований в этой области связано с изучением возможностей реляционных баз данных. В [4] сравнивается производительность PostgreSQL и MySQL, в [10] — PostgreSQL и Oracle. Основная цель этих исследований — анализ производительности каждой из представленных баз данных, особенностей работы оптимизатора запросов и индексов. Кроме того, существуют исследования более узкой направленности. Так,в [11] рассмотрены возможности асинхронной репликации в базах данных PostgreSQL, MS SQL и MySQL, основные операции при работе с данными и скорость репликации обновленных данных.

Одним из наиболее развитых направлений в этой сфере исследований является сравнительная характеристика реляционных и NoSQL-баз данных. В [12] приведены результаты оценки временно́й эффективности реляционных и NoSQL-баз данных на основе нескольких запросов, включающих операции как на выборку, так и на модификацию данных. Исследованы четыре основных операции при работе с данными: создание, чтение, обновление и удаление, — чаще называемые CRUD (от англ. Create — Read — Update — Delete). Работа [13] представляет собой верхнеуровневое изучение наиболее популярных SQL- и NoSQL-баз данных, в ней приведены их основные возможности, однако отсутствуют результаты экспериментальных оценок. В [14, 15] приводится сравнительная оценка реляционной базы данных SQL Server и NoSQL-базы данных MongoDB. Основная цель исследования — получение временны́х характеристик при выполнении различных запросов к данным.

Существуют работы по исследованию способов манипулирования данными. Так, в [16] сравниваются возможности языков SQL и HiveQL, изучаются такие характе-

ристики языков, как представленные типы данных, возможные операции с данными, основные достоинства и недостатки каждой из альтернатив.

Исследуются разные формы хранения больших данных. Кроме традиционного хранения в SQL- и NoSQL-базах данных исследуются современные средства файлового хранения в так называемых озерах данных. В [2] рассмотрены различные файловые форматы хранения данных в крупных системах обработки и хранения больших данных, различные аспекты использования файлов при хранении данных, предложена математическая модель выбора альтернатив при проектировании файловых хранилищ данных. Авторы [17] изучают форматы файлов для хранения данных в сфере биоинформатики, что является узкоспециализированным исследованием под конкретные задачи. В работе [18] рассмотрены два формата хранения данных — Apache Parquet и Apache Avro, являющиеся одними из наиболее популярных форматов в системах больших данных.

Несмотря на большую изученность темы хранения данных, при проектировании информационных систем [19] могут возникать новые вопросы. В предлагаемом исследовании рассмотрено изменение основных характеристик программного и аппаратного обеспечения обработки и хранения данных при росте объемов данных.

1. Методика экспериментальной оценки

Основная цель исследования заключается в формировании методики для оценки основных характеристик эффективности использования разных способов хранения данных при проектировании конкретной информационной системы. В качестве критериев оценки эффективности выбраны:

- занимаемый объем данных;
- время обработки данных;
- использование ресурсов оперативной памяти;
- использование ресурсов процессора;
- динамика изменения вышеописанных характеристик в зависимости от объема данных.

Разработана методика сравнительной экспериментальной оценки эффективности форматов данных.

1. Выбор двух или более альтернативных способов хранения данных [20].
2. Формирование экспериментального стенда для анализа альтернатив на виртуальной имитационной вычислительной инфраструктуре, соответствующей условию эксплуатации [1].
3. Подготовка экспериментальных запросов для выборки данных.
4. Экспериментальная оценка характеристик на запуске одного и нескольких одновременно запущенных запросов. Каждая из проверок проводится несколько раз. Среднее значение всех экспериментов используется в качестве результирующего. Такой способ выбран, чтобы избежать влияния сетевых задержек на результаты исследования.
5. Проведение исследований на разных объемах данных для оценки динамики характеристик эффективности.

Разработанная методика применена для сравнительной оценки реляционного формата хранения данных на примере базы PostgreSQL и колоночного формата файлов Apache Parquet. В ходе экспериментальной оценки используются одинаковые вычислительные узлы, характеристики которых приведены в табл. 1.

Для оценки реляционного формата данных развернут экспериментальный стенд, представляющий собой кластер из трех вычислительных узлов с установленной системой управления базами данных PostgreSQL версии 14.0. Детальные настройки PostgreSQL приведены в Приложении Б. На рис. 1 показана схема сформированного кластера. В качестве связующего элемента выбрано приложение Patroni [21]. Для хранения конфигурационных значений использовалась база данных etcd [22], которая требует отдельного узла, не участвующего в вычислительных процессах.

Для сравнения с файловым хранением подготовлен стенд с установленной системой Apache Hadoop и фреймворком Apache Hive версии 2.3.8 для доступа к данным. Стенд включает три взаимодействующих друг с другом узла HDFS, а также узел, хранящий метаинформацию о кластере. Детальные настройки Apache Hive приведены в Приложении Б. Схема экспериментального стенда для имитации системы обработки больших данных представлена на рис. 2. На трех узлах установлены интерпретатор Java версии 16.0.2, распределенная файловая система HDFSи необходимые зависимости для работы фреймворка Apache Hive. На других узлах располагается программное обеспечение для хранения метаинформации. Имеются настройки распределения задач по контейнерам, чтобы избежать ситуации попадания больших объемов информации в один контейнер, что приводит к бесконечному выполнению задач.

Для проведения эксперимента использованы подготовленные наборы данных разного объема, полученные во время проведения психологического тестирования в образовательных организациях, а также дополнительно сгенерированные данные, отвечающие схеме записи. В табл. 2 представлено описание одной записи данных. Также были подготовлены запросы к данным, на основании которых проводилось исследование. Далее (листинги 1 и 2) приведены эти запросы на языке SQL. Первый запрос представляет со-

Т а б л и ц а 1. Конфигурация вычислительного узла экспериментального стенда
Table 1. Configuration of the computational node of the experimental stand

Элемент конфигурации	Характеристика
Процессор	Intel Xeon 2.3 ГГц 4 ядра
Тип жесткого диска	HDD
Оперативная память	16 Гб
Операционная система	Ubuntu 20.04.3 LTS x64

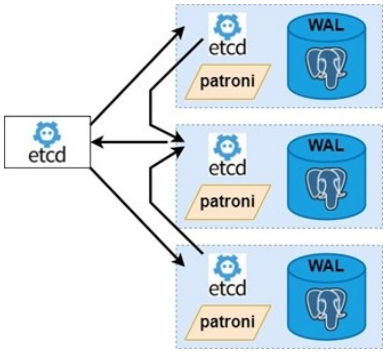


Рис. 1. Схема кластера PostgreSQL
Fig. 1. PostgreSQL cluster diagram

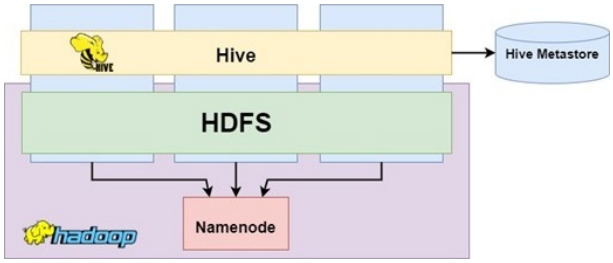


Рис. 2. Схема экспериментального стенда для имитации системы обработки больших данных
Fig. 2. Diagram of experimental environment for imitation of a big data system

Т а б л и ц а 2. Описание одной записи из экспериментального набора данных
Table 2. Description of an experimental dataset record

Название поля	Тип данных	Описание
name	varchar	Имя пользователя
surname	varchar	Фамилия пользователя
device	varchar	Тип используемого устройства
q1	double	Время ответа на вопрос 1
q2	double	Время ответа на вопрос 2
q3	double	Время ответа на вопрос 3
q4	double	Время ответа на вопрос 4
q5	double	Время ответа на вопрос 5
q6	double	Время ответа на вопрос 6
q7	double	Время ответа на вопрос 7
q8	double	Время ответа на вопрос 8
q9	double	Время ответа на вопрос 9
q10	double	Время ответа на вопрос 10

Листинг 1. SQL-запрос для эксперимента 1
Listing 1. SQL query for the experiment 1

```
select device,
count(*),
avg(q6),
percentile_cont(0.25) within group (order by q6 asc) as percentile_25,
percentile_cont(0.5) within group (order by q6 asc) as percentile_5,
percentile_cont(0.75) within group (order by q6 asc) as percentile_75,
stddev(q6)
from questionnaire
group by device;
```

Листинг 2. SQL-запрос для эксперимента 2
Listing 2. SQL query for the experiment 2

```
select name, surname
from questionnaire
where device = 'Mobile';
```

Листинг 3. Запрос Apache Hive для эксперимента 1
Listing 3. Apache Hive query for the experiment 1

```
select device,
count(*),
avg(q6),
percentile(q6, 0.25) over () as percentile_25,
percentile(q6, 0.5) over () as percentile_5,
percentile(q6, 0.75) over () as percentile_75,
stddev(q6)
from survey.questionnaire
group by device;
```

Листинг 4. Запрос Apache Hive для эксперимента 2
Listing 4. Apache Hive query for the experiment 2

```
select name, surname
from survey.questionnaire
where device = 'Mobile';
```

бой вычисление статистической информации на основании полученных значений. Такой тип запросов может применяться при разработке витрин данных [23] или подготовке отчетов.

Второй запрос более прост с точки зрения вычислительной сложности, поскольку представляет собой выборку части данных на основе условия фильтрации. Подобные запросы являются наиболее частыми при работе с базой данных, поскольку позволяют верхнеуровнево оценить данные в таблице.

Аналогичные с точки зрения результирующего набора данных запросы написаны с использованием фреймворка Apache Hive (листинги 3 и 4).

2. Результаты

На первом этапе исследования был изучен занимаемый объем данных. На рис. 3 представлены результаты при разном количестве записей. По оси x указано количество записей каждого набора данных в миллионах единиц, по оси y представлен объем данных в гигабайтах. Из графика ясно, что занимаемый объем при использовании Parquet-файлов растет значительно медленнее, чем в базе данных PostgreSQL. На втором этапе исследования были изучены такие характеристики, как время обработки запроса, а также потребление аппаратных ресурсов при выполнении описанных ранее запросов. На рис. 4 представлено время получения данных при одновременном запуске одного и пятнадцати запросов.

В Приложении А приведены графики полученных результатов по характеристикам аппаратного обеспечения для первого запроса как наиболее сложного с точки зрения вычислений. Поскольку разница между двумя альтернативами наблюдается на втором шаге эксперимента, в приложении представлены графики для наборов данных, состоящих из 50 млн записей для каждой альтернативы при запуске 15 запросов.

Из полученных результатов видно, что на небольших объемах данных Apache Hive уступает реляционной базе данных. Однако при увеличении объема данных эффективность фреймворка для обработки больших данных увеличивается. Следует также обратить внимание, что изменения времени работы второго запроса для формата Parquet незначительны. Это объясняется колоночной структурой формата, которая, в отличие от строкового представления данных, позволяет читать только необходимые колонки без удаления части данных одной записи. Из этого следует, что выбор требуемых колонок в колоночной реализации хранения данных является наиболее “дорогой” операцией, в то время как их фильтрация происходит значительно быстрее.

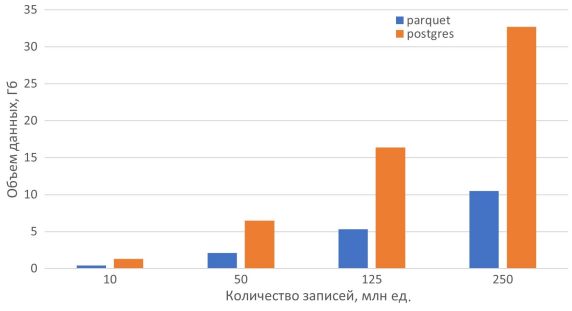


Рис. 3. Занимаемый объем данных
Fig. 3. Resulting data volume

Сравнение аппаратного обеспечения требует дополнительного анализа, поскольку представленные инструменты реализуют разную логику обработки запросов. СУБД PostgreSQL проводит параллельное выполнение запросов, поэтому большое количество ресурсов тратится на главном узле. При работе в системах обработки больших данных создаются контейнеры, располагающиеся на разных узлах системы, отчего загрузка каждого узла меньше.

Для более глубокого понимания работы каждого из инструментов проанализированы планы выполнения запросов (табл. 3–6). Для анализа взяты планы запросов, выполняемых к набору данных, состоящему из 10 млн записей.

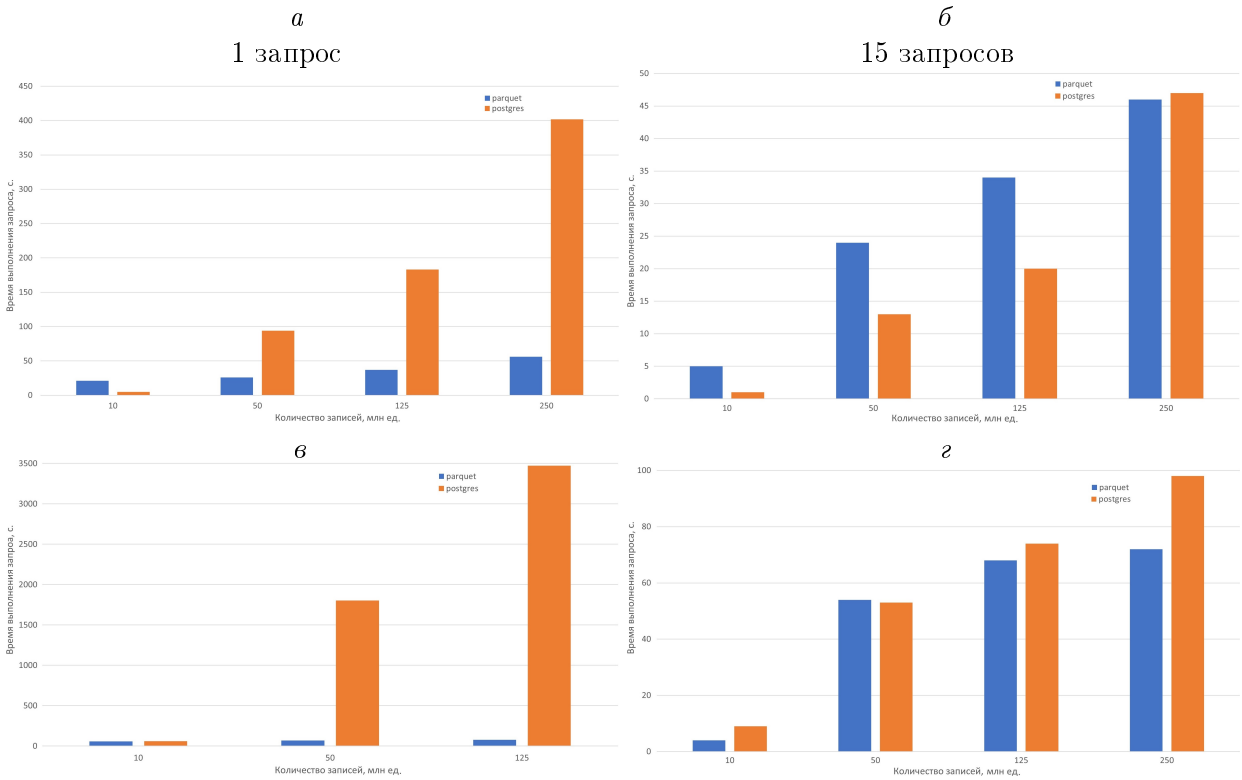


Рис. 4. Время обработки запроса 1 (*а*, *б*) и запроса 2 (*в*, *г*) при одновременной работе одного и 15 запросов
Fig. 4. Query 1 (*а*, *б*) and query 2 (*в*, *г*) processing time for a single and 15 simultaneous requests

Т а б л и ц а 3. План выполнения запроса 1 в PostgreSQL
Table 3. Query plan for the query 1 in ProstgreSQL

№	Query plan
0	GroupAggregate (cost=1621925.83..1746925.90 rows=3 width=54) (actual time=8028.698..12493.681 rows=3 loops=1)
1	Group Key: device
2	–> Sort (cost=1621925.83..1646925.83 rows=10000000 width=14) (actual time=5009.348..6160.366 rows=10000000 loops=1)
3	Sort Key: device
4	Sort Method: external merge Disk: 254472kB
5	–> Seq Scan on questionnaire (cost=0.00..263935.00 rows=10000000 width=14) (actual time=0.007..1229.218 rows=10000000 loops=1)

Т а б л и ц а 4. План выполнения запроса 1 в Apache Hive
Table 4. Query plan for the query 1 in Apache Hive

№	Query plan
1	Stage−0
2	Fetch Operator
3	limit:−1
4	Stage−1
5	Reducer 4
6	File Output Operator [FS_14]
7	Select Operator [SEL_12] (rows=368402/99852 width=182)
8	Output:["_col0","_col1","_col2","_col3","_col4","_col5","_col6"]
9	PTF Operator [PTF_11] (rows=368402/99852 width=182)
10	Function definitions :[{},{"name:":"windowingtablefunction","order by:":"0
11	ASC NULLS FIRST","partition by:":"0"}]
12	Select Operator [SEL_10] (rows=368402/99852 width=182)
13	Output:["_col0","_col1","_col2","_col3","_col4","_col5"]
14	<−Reducer 3 [SIMPLE_EDGE] vectorized
15	SHUFFLE [RS_21]
16	PartitionCols:0
17	Group By Operator [GBY_20] (rows=368402/99852 width=182)
18	Output:["_col0","_col1","_col2","_col3","_col4","_col5"],
19	aggregations:["count(VALUE._col0)","sum(VALUE._col1)", "count(VALUE._col2)","sum(VALUE._col3)"],keys:KEY._col0, KEY._col1
20	<−Reducer 2 [SIMPLE_EDGE] vectorized
21	SHUFFLE [RS_19]
22	PartitionCols: _col0, _col1
23	Group By Operator [GBY_18]
24	(rows=736805/99852 width=182)
25	Output:["_col0","_col1","_col2","_col3","_col4","_col5"],
26	aggregations:["count(VALUE._col0)","sum(VALUE._col1)", "count(VALUE._col2)","sum(VALUE._col3)"],
27	keys:KEY._col0, KEY._col1
28	<−Map 1 [SIMPLE_EDGE] vectorized
29	SHUFFLE [RS_17]
30	PartitionCols:rand()
31	Group By Operator [GBY_16] (rows=736805/99852 width=182)
32	Output:["_col0","_col1","_col2","_col3","_col4","_col5"],
33	aggregations:["count()","sum(_col1)","count(_col1)","sum(_col2)"],keys: _col0, _col1
34	Select Operator [SEL_15] (rows=736805/100000 width=182)
35	Output:["_col0","_col1","_col2"] TableScan [TS_0] (rows=736805/100000 width=182)
36	default@questionnaire1m,questionnaire1m, Tbl:COMPLETE,Col:NONE,Output:["device","q6"]

Т а б л и ц а 5. План выполнения запроса 2 в PostgreSQL
Table 5. Query plan for the query 2 in ProstgreSQL

№	Query plan
0	Seq Scan on questionnaire (cost=0.00..288935.00 rows=3341100 width=14) (actual time=0.027..1321.974 rows=3333439 loops=1)
1	Filter: ((device)::text = 'Mobile'::text)
2	Rows Removed by Filter: 6666561

Т а б л и ц а 6. План выполнения запроса 2 в Apache Hive
Table 6. Query plan for the query 2 in Apache Hive

№	Query plan
1	Stage–0
2	Fetch Operator
3	limit:–1
4	Stage–1
5	Map 1 vectorized
6	File Output Operator [FS_7]
7	Select Operator [SEL_6] (rows=18419603/16670622 width=564)
8	Output:["_col0","_col1"]
9	Filter Operator [FIL_5] (rows=18419603/16670622 width=564) predicate:(device = 'Mobile')
10	TableScan [TS_0] (rows=36839206/50000010 width=564) default@questionnaire,questionnaire,
11	Tbl:COMPLETE,Col:NONE,Output:["name","surname","device"]

Запрос, выполняемый СУБД PostgreSQL, проходит следующие этапы реализации (см. табл. 3). На этапе GroupAggregate формируется таблица соответствия ключа, которым выступает параметр агрегации (в данном случае поле device), и текущей строки. Особенностью этой операции является необходимость сортировки по данному ключу. Затем выполняется расчет всех необходимых статистик.

В запросе, выполняемом фреймворком Apache Hive, более сложная логика (см. табл. 4). Связано это с необходимостью Shuffle-операции, т. е. перемешивания данных между рабочими контейнерами. Это наиболее "дорогая" операция при работе с большими данными, поскольку требует передачи больших объемов информации по сети.

Второй запрос более простой. В данном случае СУБД PostgreSQL (см. табл. 5) проходит по всему набору данных. Единственной выполняемой при проходе набора операцией является фильтрация по указанному параметру. При использовании файлов Apache Parquet алгоритм выборки изменен. По приведенному плану (см. табл. 6) можно заметить, что изначально выбираются лишь нужные колонки, а затем происходит сканирование, при котором уже нет необходимости удалять лишние колонки, а фильтрация выполняется на основе одного параметра.

Важно также отметить разницу в обработке данных. При использовании кластера СУБД PostgreSQL бóльшая часть запросов попадает на главный сервер, в то время как Apache Hive распределяет весь набор на несколько контейнеров, каждый из которых обрабатывает лишь часть общего набора.

3. Дискуссия

Особенностью настоящего исследования является тот факт, что оценка инструментов хранения формировалась на основе запросов по выборке данных с дополнительными условиями по их обработке и вычислению статистических величин. Такой подход вызван несколькими причинами. Во-первых, системы обработки больших данных поддерживают принцип WORM (Write Once — Read Many), предполагающий однократную запись данных без последующих изменений. Отсюда следует, что описанные системы имеют абсолютно разные подходы к записи и обновлению данных. Однако они имеют схожий способ выборки данных, что позволяет проводить исследование на запросах

выборки данных. Во-вторых, описанная схема представляет собой первый уровень обработки данных — так называемые сырые данные, которые требуют следующих этапов обработки. Часто задачи сводятся к построению таких структур, в которых данные хранятся в заранее обработанном виде. К таким задачам можно отнести разработку витрин данных [23], отчетов, OLAP-кубов [24].

Следует обратить внимание на то, что структура выбранного набора данных статична, тогда как структура опросников часто изменяется, что говорит о необходимости рассмотреть возможность анализа эффективности инструментов обработки слабоструктурированных данных, таких как документные базы данных [25].

Важно также отметить, что инструменты для обработки больших данных зависимы от правильно выставленных настроек, в то время как реляционные базы данных чаще всего имеют автоматизированный оптимизатор запросов [26]. Это означает, что проведение аналогичных исследований может иметь отличающийся результат в случае выставления иных настроек.

Одной из главных проблем инструментов распределенной обработки больших данных является так называемая ошибка (data skew), приводящая к загрузке одного из контейнеров выполнения задач, в то время как другие контейнеры не получают или получают минимальные наборы данных для обработки.

Заключение

Предложенная методика сравнительной экспериментальной оценки эффективности форматов данных состоит из пяти шагов:

1) выбор двух или более альтернативных способов хранения данных;
2) формирование экспериментального стенда на виртуальной имитационной вычислительной инфраструктуре;
3) подготовка экспериментальных запросов;
4) экспериментальная оценка характеристик на запуске одного и нескольких одновременно запущенных запросов;
5) проведение исследований на разных объемах данных для оценки динамики характеристик эффективности.

С использованием предложенной методики проведена оценка двух способов хранения данных: в реляционной базе данных и в виде файлов колоночного формата. В качестве реляционной базы данных выбрана система управления базами данных PostgreSQL. В качестве альтернативы выбрана связка Apache Hive и Apache Parquet. В последние годы эти два инструмента стали де-факто стандартом для хранения и обработки больших данных.

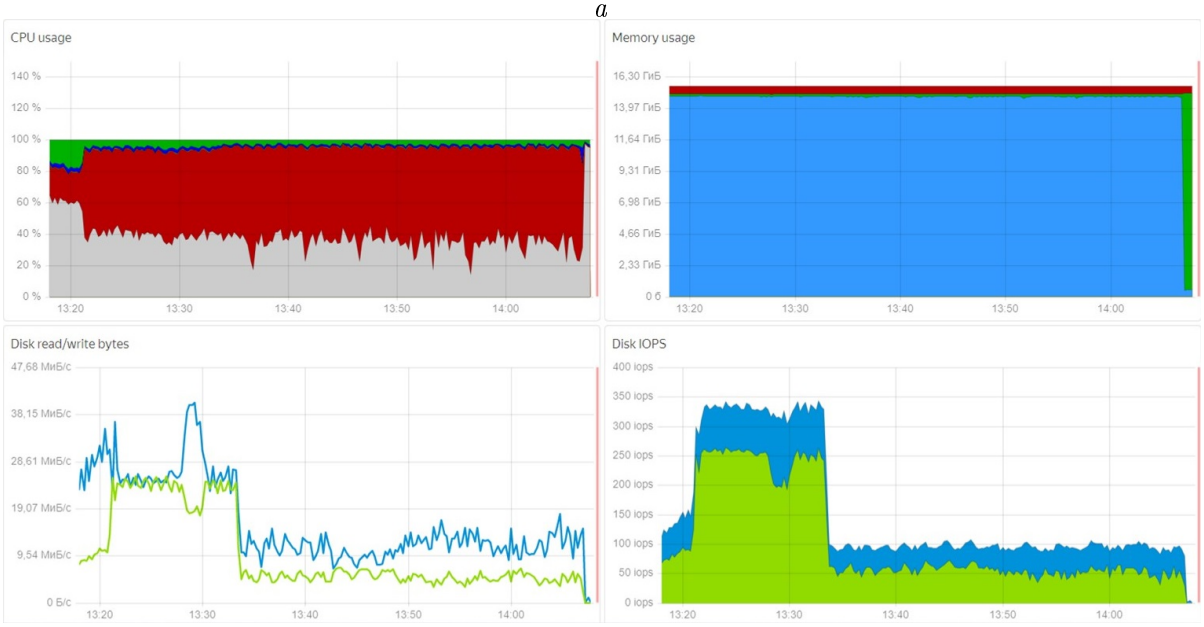
Основными исследуемыми характеристиками эффективности выбраны занимаемый объем данных, время обработки данных, а также используемые аппаратные ресурсы. Для проведения эксперимента со временем обработки данных подготовлены два SQL-запроса для PostgreSQL и аналогичные для Apache Hive.

Результат подтвердил целесообразность использования реляционной базы данных для относительно небольших объемов данных и необходимость рассмотрения альтернативных способов хранения данных при значительном увеличении объема. Однако следует отметить, что полученные результаты не являются эталонными. Системы обработки больших данных зависимы от выставленных настроек, отчего результаты аналогичных испытаний могут различаться.

Благодарности. Работа выполнена при финансовой поддержке гранта РТУ МИРЭА “Инновации в реализации приоритетных направлений развития науки и технологий” (проект НИЧ 28/22).

Приложение А

а



б

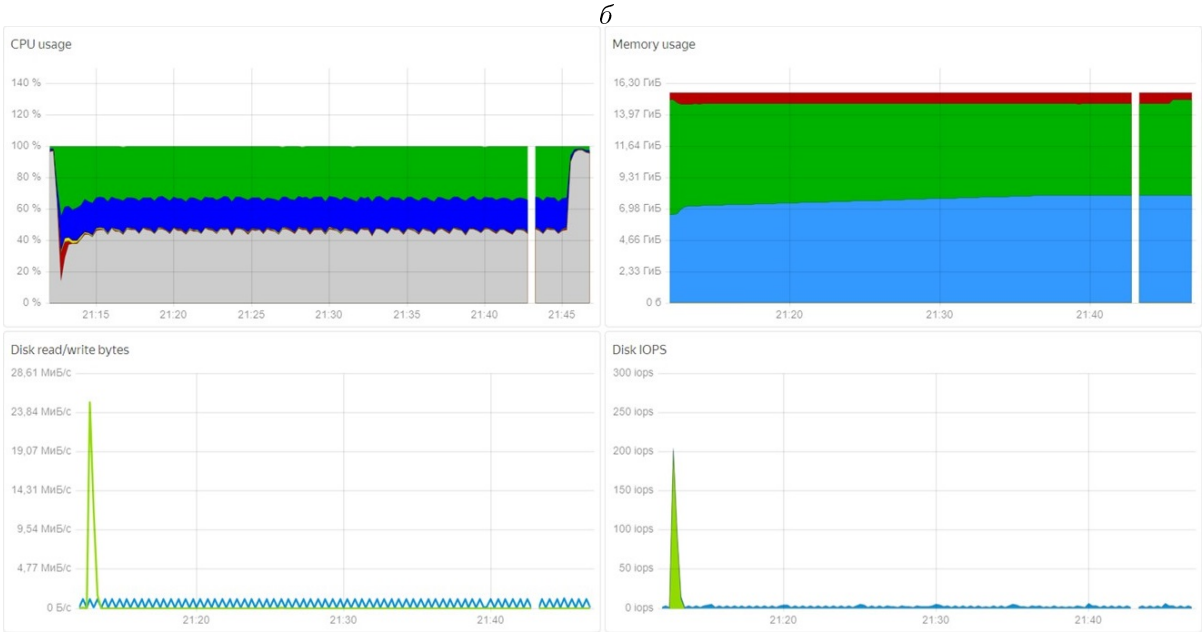
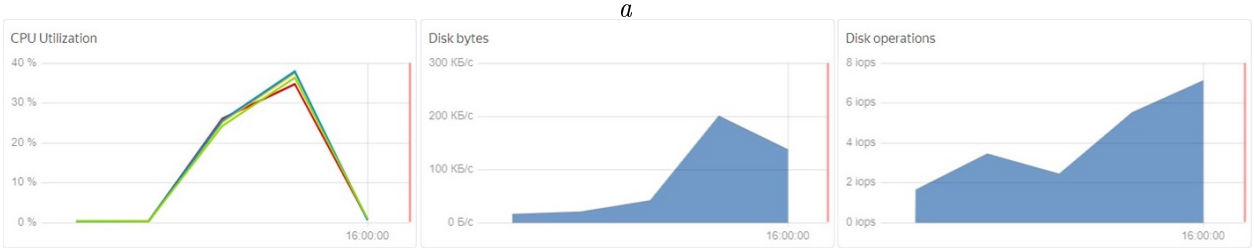


Рис. ПА1. Использование аппаратного обеспечения на главном узле PostgreSQL для запросов 1 (*а*) и 2 (*б*) при 15 работающих потоках

Fig. ПА1. PostgreSQL main node hardware resource utilization for processing query 1 for 15 simultaneous requests

а



б

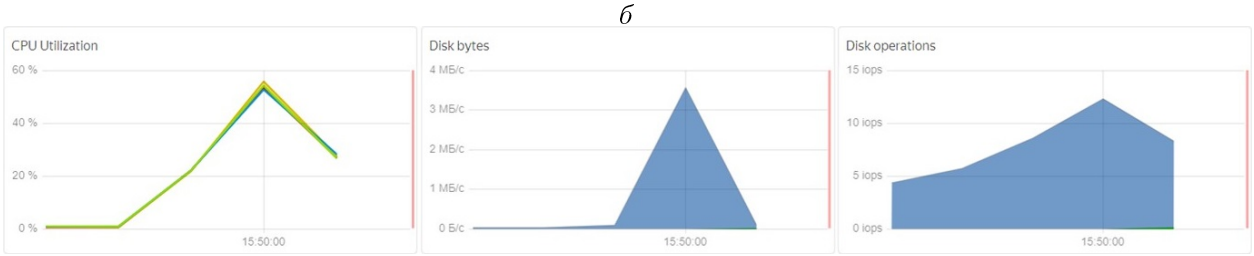


Рис. ПА2. Использование аппаратного обеспечения на главном узле Apache Hive для запросов 1 и 2 (*а* и *б* соответственно) при 15 работающих потоках
Fig. ПА2. Apache Hive main node hardware resource utilization for processing queries 1 and 2 (*а* and *б* respectively) for 15 simultaneous requests

Приложение Б

Для проведения экспериментов с базой данных PostgreSQL использованы настройки по умолчанию. В табл ПБ1 представлены основные свойства, влияющие на оптимизацию выполнения запросов при проведении экспериментальных замеров.

Для проведения исследования на кластере Apache Hive применены настройки по умолчанию. Однако некоторые из них изменены во избежание проблемы переполнения контейнеров обработки данных, известной как data skew. В табл. ПБ2 приведены основные настройки, оказывающие влияние на оптимизацию экспериментальных запросов.

Т а б л и ц а ПБ1. Свойства базы данных PostgreSQL
Table ПБ1. PostgreSQL database properties

Свойство	Значение
autovacuum	on
autovacuum_analyze_scale_factor	0.0001
autovacuum_analyze_threshold	50
autovacuum_freeze_max_age	200000000
autovacuum_max_workers	4
autovacuum_multixact_freeze_max_age	200000000
autovacuum_naptime	15s
autovacuum_vacuum_cost_delay	35ms
autovacuum_vacuum_cost_limit	1000
autovacuum_vacuum_insert_scale_factor	0.2
autovacuum_vacuum_insert_threshold	1000
autovacuum_vacuum_scale_factor	1e−05
autovacuum_vacuum_threshold	200
backend_flush_after	0
bgwriter_delay	200ms

Продолжение табл. ПБ1

Свойство	Значение
bgwriter_flush_after	512kB
bgwriter_lru_maxpages	100
block_size	8192
commit_delay	0
commit_siblings	5
compute_query_id	auto
cpu_index_tuple_cost	0.005
cpu_operator_cost	0.0025
cpu_tuple_cost	0.01
cursor_tuple_fraction	0.1
data_checksums	on
deadlock_timeout	1s
enable_async_append	on
enable_bitmapscan	on
enable_gathermerge	on
enable_hashagg	on
enable_hashjoin	on
enable_incremental_sort	on
enable_indexonlyscan	on
enable_indexscan	on
enable_material	on
enable_memoize	on
enable_mergejoin	on
enable_nestloop	on
enable_parallel_append	on
enable_parallel_hash	on
enable_partition_pruning	on
enable_seqscan	on
enable_sort	on
enable_tidscan	on
escape_string_warning	on
extra_float_digits	3
geqo	on
geqo_effort	5
geqo_generations	0
geqo_pool_size	0
geqo_seed	0
geqo_selection_bias	2
geqo_threshold	12
hot_standby	on
hot_standby_feedback	on
huge_page_size	0
jit_above_cost	100000
jit_expressions	on
jit_inline_above_cost	500000

Продолжение табл. ПБ1

Свойство	Значение
jit_optimize_above_cost	500000
jit_tuple_deforming	on
lc_collate	C
lc_ctype	C
maintenance_work_mem	64MB
max_index_keys	32
min_dynamic_shared_memory	0
min_parallel_index_scan_size	512kB
min_parallel_table_scan_size	8MB
parallel_leader_participation	on
parallel_setup_cost	1000
parallel_tuple_cost	0.1
pg_stat_kcache.linux_hz	1000000
pg_stat_statements.max	10000
pg_stat_statements.save	on
pg_stat_statements.track	all
plan_cache_mode	auto
post_auth_delay	0
pre_auth_delay	0
update_process_title	on
vacuum_cost_delay	0
vacuum_cost_limit	200
vacuum_cost_page_dirty	20
vacuum_defer_cleanup_age	0
vacuum_failsafe_age	1600000000
vacuum_freeze_min_age	50000000
vacuum_freeze_table_age	150000000
vacuum_multixact_failsafe_age	1600000000
vacuum_multixact_freeze_min_age	5000000
vacuum_multixact_freeze_table_age	75000000
work_mem	4MB

Т а б л и ц а ПБ2. Свойства Apache Hive
Table ПБ2. Apache Hive properties

Свойство	Значение
hive.combine.equivalent.work.optimization	true
hive.compute.query.using.stats	true
hive.compute.splits.in.am	true
hive.constraint.notnull.enforce	true
hive.driver.parallel.compilation	false
hive.groupby.limit.extrastep	true
hive.groupby.mapaggr.checkinterval	100000
hive.groupby.orderby.position.alias	false
hive.groupby.position.alias	false

Продолжение табл. ПБ2

Свойство	Значение
hive.groupby.skewindata	true
hive.hashtable. initialCapacity	100000
hive.hashtable.key.count.adjustment	2.0
hive.hashtable.loadfactor	0.75
hive.heap.memory.monitor.usage.threshold	0.7
hive.limit.optimize.enable	false
hive.limit.optimize.fetch.max	50000
hive.limit.optimize.limit. file	10
hive.limit.pushdown.memory.usage	0.1
hive.limit.row.max.size	100000
hive.load.dynamic.partitions.thread	15
hive.map.aggr	true
hive.map.aggr.hash.force.flush.memory.threshold	0.9
hive.map.aggr.hash.min.reduction	0.5
hive.map.aggr.hash.percentmemory	0.5
hive.map.groupby.sorted	true
hive.mapper.cannot.span.multiple.partitions	false
hive.mapred.local.mem	0
hive.mapred.partitioner	org.apache.hadoop.hive.ql.io. DefaultHivePartitioner
hive.mapred.reduce.tasks.speculative.execution	true
hive.max.open.txns	100000
hive.merge.cardinality.check	true
hive.merge.mapfiles	true
hive.merge.mapredfiles	false
hive.merge.nway.joins	true
hive.merge.size.per.task	2.56E+08
hive.metadata.move.exported.metadata.to.trash	true
hive.multigroupby.singlereducer	true
hive.new.job.grouping.set. cardinality	30
hive.optimize.bucketingsorting	true
hive.optimize.bucketmapjoin	false
hive.optimize.bucketmapjoin.sortedmerge	false
hive.optimize.constant.propagation	true
hive.optimize.correlation	false
hive.optimize.countdistinct	true
hive.optimize.cte.materialize.threshold	−1
hive.optimize.distinct.rewrite	true
hive.optimize.dynamic.partition.hashjoin	false
hive.optimize. filter .stats.reduction	false
hive.optimize.groupby	true
hive.optimize.index. filter	false
hive.optimize.joinreducededuplication	true
hive.optimize.limittranspose	false
hive.optimize.limittranspose.reductionpercentage	1.0

Продолжение табл. ПБ2

Свойство	Значение
hive.optimize.limittranspose.reductiontuples	0
hive.optimize.listbucketing	false
hive.optimize.metadataonly	false
hive.optimize.null.scan	true
hive.optimize.partition.columns.separate	true
hive.optimize.point.lookup	true
hive.optimize.point.lookup.min	31
hive.optimize.ppd	true
hive.optimize.ppd.storage	true
hive.optimize.ppd.windowing	true
hive.optimize.reducededuplication	true
hive.optimize.reducededuplication.min.reducer	4
hive.optimize.remove.identity.project	true
hive.optimize.remove.sq_count_check	false
hive.optimize.sampling.orderby	false
hive.optimize.sampling.orderby.number	1000
hive.optimize.sampling.orderby.percent	0.1
hive.optimize.semijoin.conversion	true
hive.optimize.shared.work	true
hive.optimize.shared.work.extended	true
hive.optimize.skewjoin	true
hive.optimize.skewjoin.compiletime	true
hive.optimize.sort.dynamic.partition	true
hive.optimize.union.remove	false
hive.optimize.update.table.properties.from.serde	false
hive.optimize.update.table.properties.from.serde.list	org.apache.hadoop.hive.serde2.avro.AvroSerDe
hive.parquet.timestamp.skip.conversion	false
hive.support.concurrency	false
hive.vectorized.adaptor.suppress.evaluate.exceptions	false
hive.vectorized.adaptor.usage.mode	all
hive.vectorized.complex.types.enabled	true
hive.vectorized.execution.enabled	true
hive.vectorized.execution.mapjoin.minmax.enabled	false
hive.vectorized.execution.mapjoin.native.enabled	true
hive.vectorized.execution.mapjoin.native.fast.hashtable.enabled	false
hive.vectorized.execution.mapjoin.native.multikey.only.enabled	false
hive.vectorized.execution.mapjoin.overflow.repeated.threshold	−1
hive.vectorized.execution.ptf.enabled	true
hive.vectorized.execution.reduce.enabled	true
hive.vectorized.execution.reduce.groupby.enabled	true

Продолжение табл. ПБ2

Свойство	Значение
hive.vectorized.execution.reducesink.new.enabled	true
hive.vectorized.groupby.checkinterval	100000
hive.vectorized.groupby.complex.types.enabled	true
hive.vectorized.groupby.flush.percent	0.1
hive.vectorized.groupby.maxentries	1000000
hive.vectorized.input.format.supports.enabled	decimal_64
hive.vectorized.ptf.max.memory.buffering.batch.count	25
hive.vectorized.reuse.scratch.columns	true
hive.vectorized.row. identifier .enabled	true
hive.vectorized.row.serde.inputformat.excludes	org.apache.parquet.hadoop. ParquetInputFormat, org.apache.hadoop.hive .ql.io.parquet.MapredParquetInputFormat

Список литературы

[1] **Gusev A., Ilin D., Kolyasnikov P., Nikulchev E.** Effective selection of software components based on experimental evaluations of quality of operation. Engineering Letters. 2020; 28(2):420–427.

[2] **Belov V., Tatarintsev A., Nikulchev E.** Choosing a data storage format in the Apache Hadoop system based on experimental evaluation using Apache Spark. Symmetry. 2021; 13(2):195. DOI:10.3390/sym13020195.

[3] PostgreSQL. Адрес доступа: `http://www.postgresql.org` (дата обращения 31.01.2022).

[4] **Andjelic S., Obradovic S., Gacesa B.** A performance analysis of the DBMS — MySQL vs PostgreSQL. Communications — Scientific Letters of the University of Zilina. 2008; 10(4):53–57.

[5] **Lee S., Jo J.Y., Kim Y.** Survey of data locality in Apache Hadoop. 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD). Honolulu, USA; 2019: 46–53.

[6] **Camacho-Rodriguez J., Chauhan A., Gates A., Koifman E., O'Malley O., Garg V., Haindrich Z., Shelukhin S., Jayachandran P., Seth S., Jaiswal D., Bouguerra S., Bangarwa N., Hariappan S., Agarwal A., Dere J., Dai D., Nair T., Dembla N., Vijayaraghavan G., Hagleitner G.** Apache Hive: from MapReduce to enterprise-grade Big Data warehousing. Proceedings of the 2019 International Conference on Management of Data (SIGMOD'19). N.Y., USA: Association for Computing Machinery; 2019: 1773–1786. DOI:10.1145/3299869.3314045.

[7] Apache Parquet. Адрес доступа: `https://parquet.apache.org/documentation/latest` (дата обращения 31.01.2022).

[8] **Alasta A.F., Enaba M.A.** Data warehouse on manpower employment for decision support system. International Journal of Computing, Communication and Instrumentation Engineering. 2014; (1):48–53.

[9] **Cappa F., Oriani R., Peruffo E., McCarthy I.P.** Big Data for creating and capturing value in the digitalized environment: unpacking the effects of volume, variety and veracity on firm performance. Journal of Product Innovation Management. 2021; 38(1):49–67.

[10] **Martins P., Tome P., Wanzeller C., Sa F., Abbasi M.** Comparing Oracle and PostgreSQL, performance and optimization. Trends and Applications in Information Systems and Technologies. Springer, Cham; 2021: 1366. DOI:10.1007/978-3-030-72651-5_46.

[11] **Truica C.-O., Boicea A., Radulescu F.** Asynchronous replication in Microsoft SQL Server, PostgreSQL and MySQL. International Conference on Cyber Science and Engineering. China, Guangzhou; 2013: 50–55.

[12] **Li Y., Manoharan S.** A performance comparison of SQL and NoSQL databases. 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM). 2013: 15–19. DOI:10.1109/PACRIM.2013.6625441.

[13] **Cattell R.** Scalable SQL and NoSQL data stores. SIGMOD Record. 2010; 39(4):12–27. DOI:10.1145/1978915.1978919.

[14] **Parker Z., Poe S., Vrbsky S.** Comparing NoSQL MongoDB to an SQL DB. Proceedings of the 51st ACM Southeast Conference (ACMSE'13). N.Y., USA: Association for Computing Machinery; 2013: 1–6. DOI:10.1145/2498328.2500047.

[15] **Jung M., Youn S., Bae J., Choi Y.** A study on data input and output performance comparison of MongoDB and PostgreSQL in the Big Data environment. 2015 8th International Conference on Database Theory and Application (DTA). Jeju, Korea (South): IEEE; 2015: 14–17. DOI:10.1109/DTA.2015.14.

[16] **Kumar R., Gupta N., Charu Sh., Bansal S., Yadav K.** Comparison of SQL with HiveQL. International Journal for Research in Technological Studies. 2014; 1(9):28–30.

[17] **Ahmed S., Ali M.U., Ferzund J., Sarwar M.A., Rehman A., Mehmood A.** Modern data formats for Big Bioinformatics Data analytics. International Journal of Advanced Computer Science and Applications. 2017; 8(4):366–377. DOI:10.14569/IJACSA.2017.080450.

[18] **Plase D., Niedrite L., Taranovs R.** A comparison of HDFS compact data formats: Avro Versus Parquet. Mokslas–Lietuvos Ateitis/Science-Future of Lithuania. 2017; 9(3):267–276.

[19] **Изергин Д.А., Еремеев М.А., Магомедов Ш.Г., Смирнов С.И.** Оценка уровня информационной безопасности мобильной операционной системы Android. Russian Technological Journal. 2019; 7(6):44–55. DOI:10.32362/2500-316X-2019-7-6-44-55.

[20] **Nikulchev E., Ilin D., Gusev A.** Technology stack selection model for software design of digital platforms. Mathematics. 2021; 9(4):308. DOI:10.3390/math9040308.

[21] Patroni. Адрес доступа: `https://patroni.readthedocs.io/en/latest` (дата обращения 01.03.2022).

[22] etcd. Адрес доступа: `https://etcd.io` (дата обращения 01.03.2022).

[23] **Belov V., Kosenkov A.N., Nikulchev E.** Experimental characteristics study of data storage formats for data marts development within Data Lakes. Applied Sciences. 2021; 11(18):8651. DOI:10.3390/app11188651.

[24] **Chaudhuri S., Dayal U.** An overview of data warehousing and OLAP technology. SIGMOD Record. 1997; 26(1):65–74. DOI:10.1145/248603.248616.

[25] **Popescul A., Flake G.W., Lawrence S., Ungar L.H., Giles C.L.** Clustering and identifying temporal trends in document databases. Proceedings IEEE Advances in Digital Libraries 2000. Washington, USA; 2000: 173–182. DOI:10.1109/ADL.2000.848380.

[26] **Li D., Han L., Ding Y.** SQL query optimization methods of relational database system. 2010 Second International Conference on Computer Engineering and Applications. Bali, Island; 2010: 557–560. DOI:10.1109/ICCEA.2010.113.

Вычислительные технологии, 2022, том 27, № 3, с. 46–65. © ФИЦ ИВТ, 2022 ISSN 1560-7534
Computational Technologies, 2022, vol. 27, no. 3, pp. 46–65. © FRC ICT, 2022 eISSN 2313-691X

INFORMATION TECHNOLOGIES

DOI:10.25743/ICT.2022.27.3.005

Comparative evaluation of the efficiency of data processing by storing the data in a relational database and column format files

Belov Vladimir A., Ilin Dmitry Yu., Nikulchev Evgeny V.*

MIREA — Russian Technological University, 119454, Moscow, Russia
*Corresponding author: Nikulchev Evgeny V., e-mail: nikulchev@mail.ru

Received April 01, 2022, revised April 12, 2022, accepted April 18, 2022.

Abstract

Purpose. In the process of developing information and analytical systems, the choice of the most effective tool for data storage is important. The purpose of the presented study is to compare the data processing features for various data storage tools. Analysis of these features in the dynamics of the growth of the data volume is an important issue.

Methodology. Stands were prepared for experimental evaluation of the two presented alternatives. As evaluation criteria, the data volume, processing time, the use of RAM and processor resources as well as the dynamics of changes in the characteristics with a change in the data volume was chosen. Two data queries were prepared that contain different requirements for obtaining results: filtering and data aggregation. For evaluation, both one and several simultaneously running queries were launched.

Findings. Numerical characteristics of the examined criteria were obtained. The processing speed when using a relational database was several times higher than the results obtained when using a big data processing system. As the volume of data grows, big data processing systems perform better. Regarding characteristics such as the data volume, the use of column formats is more efficient for any amount of data.

Value. The results showed the feasibility of using a relational database with small amounts of data. As the volume of data grows, it is necessary to use alternative ways of storing and processing data, which suggests that when designing a system, not only the analysis of the data structure is required, but also the estimated volume.

Keywords: big data, data storage formats, relational databases, PostgreSQL, Apache Hive.

Citation: Belov V.A., Ilin D.Yu., Nikulchev E.V. Comparative evaluation of the efficiency of data processing by storing the data in a relational database and column format files. Computational Technologies. 2022; 27(3):46–65. DOI:10.25743/ICT.2022.27.3.005. (In Russ.)

Acknowledgements. The work is supported by RTU MIREA, grant "Innovations in the realization of priority areas for the development of science and technology" (project NICh 28/22).

References

1. **Gusev A., Ilin D., Kolyasnikov P., Nikulchev E.** Effective selection of software components based on experimental evaluations of quality of operation. Engineering Letters. 2020; 28(2):420–427.
2. **Belov V., Tatarintsev A., Nikulchev E.** Choosing a data storage format in the Apache Hadoop system based on experimental evaluation using Apache Spark. Symmetry. 2021; 13(2):195. DOI:10.3390/sym13020195.
3. PostgreSQL. Available at: http://www.postgresql.org (accessed January 31, 2022).
4. **Andjelic S., Obradovic S., Gacesa B.** A performance analysis of the DBMS — MySQL vs PostgreSQL. Communications — Scientific Letters of the University of Zilina. 2008; 10(4):53–57.
5. **Lee S., Jo J.Y., Kim Y.** Survey of data locality in Apache Hadoop. 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD). Honolulu, USA; 2019: 46–53.

6. **Camacho-Rodriguez J., Chauhan A., Gates A., Koifman E., O'Malley O., Garg V., Haindrich Z., Shelukhin S., Jayachandran P., Seth S., Jaiswal D., Bouguerra S., Bangarwa N., Hariappan S., Agarwal A., Dere J., Dai D., Nair T., Dembla N., Vijayaraghavan G., Hagleitner G.** Apache Hive: from MapReduce to enterprise-grade Big Data warehousing. Proceedings of the 2019 International Conference on Management of Data (SIGMOD'19). N.Y., USA: Association for Computing Machinery; 2019: 1773–1786. DOI:10.1145/3299869.3314045.
7. Apache Parquet. Available at: `https://parquet.apache.org/documentation/latest`.
8. **Alasta A.F., Enaba M.A.** Data warehouse on manpower employment for decision support system. International Journal of Computing, Communication and Instrumentation Engineering. 2014; (1):48–53.
9. **Cappa F., Oriani R., Peruffo E., McCarthy I.P.** Big Data for creating and capturing value in the digitalized environment: unpacking the effects of volume, variety and veracity on firm performance. Journal of Product Innovation Management. 2021; 38(1):49–67.
10. **Martins P., Tome P., Wanzeller C., Sa F., Abbasi M.** Comparing Oracle and PostgreSQL, performance and optimization. Trends and Applications in Information Systems and Technologies. Springer, Cham; 2021: 1366. DOI:10.1007/978-3-030-72651-5_46.
11. **Truica C.-O., Boicea A., Radulescu F.** Asynchronous replication in Microsoft SQL Server, PostgreSQL and MySQL. International Conference on Cyber Science and Engineering. China, Guangzhou; 2013: 50–55.
12. **Li Y., Manoharan S.** A performance comparison of SQL and NoSQL databases. 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM). 2013: 15–19. DOI:10.1109/PACRIM.2013.6625441.
13. **Cattell R.** Scalable SQL and NoSQL data stores. SIGMOD Record. 2010; 39(4):12–27. DOI:10.1145/1978915.1978919.
14. **Parker Z., Poe S., Vrbsky S.** Comparing NoSQL MongoDB to an SQL DB. Proceedings of the 51st ACM Southeast Conference (ACMSE'13). N.Y., USA: Association for Computing Machinery; 2013: 1–6. DOI:10.1145/2498328.2500047.
15. **Jung M., Youn S., Bae J., Choi Y.** A study on data input and output performance comparison of MongoDB and PostgreSQL in the Big Data environment. 2015 8th International Conference on Database Theory and Application (DTA). Jeju, Korea (South): IEEE; 2015: 14–17. DOI:10.1109/DTA.2015.14.
16. **Kumar R., Gupta N., Charu Sh., Bansal S., Yadav K.** Comparison of SQL with HiveQL. International Journal for Research in Technological Studies. 2014; 1(9):28–30.
17. **Ahmed S., Ali M.U., Ferzund J., Sarwar M.A., Rehman A., Mehmood A.** Modern data formats for Big Bioinformatics Data analytics. International Journal of Advanced Computer Science and Applications. 2017; 8(4):366–377. DOI:10.14569/IJACSA.2017.080450.
18. **Plase D., Niedrite L., Taranovs R.** A comparison of HDFS compact data formats: Avro Versus Parquet. Mokslas–Lietuvos Ateitis/Science-Future of Lithuania. 2017; 9(3):267–276.
19. **Izergin D.A., Eremeev M.A., Magomedov S.G., Smirnov S.I.** Information security evaluation for Android mobile operating system. Russian Technological Journal. 2019; 7(6):44–55. DOI:10.32362/2500-316X-2019-7-6-44-55. (In Russ.)
20. **Nikulchev E., Ilin D., Gusev A.** Technology stack selection model for software design of digital platforms. Mathematics. 2021; 9(4):308. DOI:10.3390/math9040308.
21. Patroni. Available at: `https://patroni.readthedocs.io/en/latest` (accessed March 01, 2022).
22. etcd. Available at: `https://etcd.io` (accessed March 01, 2022).
23. **Belov V., Kosenkov A.N., Nikulchev E.** Experimental characteristics study of data storage formats for data marts development within Data Lakes. Applied Sciences. 2021; 11(18):8651. DOI:10.3390/app11188651.
24. **Chaudhuri S., Dayal U.** An overview of data warehousing and OLAP technology. SIGMOD Record. 1997; 26(1):65–74. DOI:10.1145/248603.248616.
25. **Popescul A., Flake G.W., Lawrence S., Ungar L.H., Giles C.L.** Clustering and identifying temporal trends in document databases. Proceedings IEEE Advances in Digital Libraries 2000. Washington, USA; 2000: 173–182. DOI:10.1109/ADL.2000.848380.
26. **Li D., Han L., Ding Y.** SQL query optimization methods of relational database system. 2010 Second International Conference on Computer Engineering and Applications. Bali, Island; 2010: 557–560. DOI:10.1109/ICCEA.2010.113.