JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 28 (2023) 204–209
Received: 24 May 2023
Accepted: 25 June 2023

Analyze the effectiveness of ETL processes implemented using SQL and Apache HiveQL languages

Analiza efektywności procesów ETL realizowanych z użyciem języków SQL i Apache HiveQL

Krzysztof Damian Litka*

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

In the era of digitization, where data is collected in ever-increasing quantities, efficient processing is required. The article analyzes the performance of SQL and HiveQL, for scenarios of varying complexity, focusing on the execution time of individual queries. The tools used in the study are also discussed. The results of the study for each language are summarized and compared, highlighting their strengths and weaknesses, as well as identifying their possible areas of application.

Keywords: ETL; SQL; HiveQL

Streszczenie

W dobie cyfryzacji, gdzie dane gromadzone są w coraz większych ilościach, wymagane jest ich efektywne przetwarzanie. W artykule dokonano analizy wydajności języka SQL i HiveQL, dla scenariuszy o zróżnicowanym stopniu złożoności, skupiając się na czasie wykonania poszczególnych zapytań. Omówiono także wykorzystane w badaniu narzędzia. Wyniki badań dla poszczególnych języków zostały zestawione i porównane, podkreślając ich mocne i słabe strony, a akże określając ich możliwe obszary zastosowań.

Słowa kluczowe: ETL; SQL; HiveQL

*Corresponding author

Email address: s99174@pollub.edu.pl (K. D. Litka)

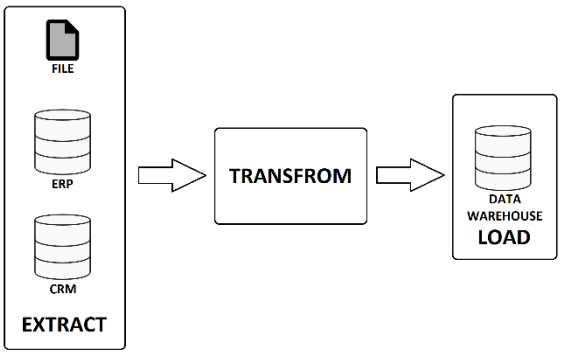
1. Wstęp

W dobie gwałtownego wzrostu ilości danych generowanych przez różne systemy informatyczne, coraz bardziej kluczowe staje się efektywne zarządzanie tymi danymi, ich analiza i przetwarzanie. To powoduje, że procesy ETL (Extract, Transform, Load), które służą do pobierania, transformacji i ładowania danych z różnych źródeł do magazynu danych, stają się coraz bardziej istotne. Kluczowym aspektem procesów ETL jest nie tylko skuteczność, ale także zdolność do szybkiego przetwarzania dużych ilości danych, przy minimalnym wykorzystaniu zasobów. SQL, będący standardowym językiem do zarządzania danymi przechowywanymi w relacyjnych bazach danych, jest powszechnie stosowany w tradycyjnych procesach ETL. Z drugiej strony, Apache HiveQL, będący częścią ekosystemu Hadoop i zapewniający interfejs zapytań podobnych do SQL, jest coraz częściej wykorzystywany w procesach ETL związanych z Big Data [2, 9].

Celem tego artykułu jest szczegółowe zbadanie i porównanie efektywności dwóch popularnych języków zapytań - SQL i HiveQL – w kontekście różnorodnych scenariuszy procesów ETL. Te scenariusze zostały zaprojektowane tak, aby odzwierciedlać typowe przypadki użycia w biznesie, obejmujące różne stopnie złożoności. Przez przeprowadzenie serii eksperymentów, porównana została wydajność tych dwóch języków, ale także przeanalizowano różne funkcje i ograniczenia związane z ich użyciem. W pracy tej zwrócono szczególną uwagę na wpływ różnych czynników, takich jak rozmiar danych wejściowych i złożoność zapytań, na efektywność procesów ETL. W tym kontekście, badania te stanowią istotny wkład do zrozumienia, jak te czynniki mogą wpływać na wybór odpowiedniego języka zapytań dla konkretnego zadania czy projektu. Ponadto, praca ta jest zakorzeniona w istniejącej literaturze teoretycznej dotyczącej procesów ETL oraz w literaturze dotyczącej wydajności narzędzi, które są wykorzystywane w tych procesach. Zdobyta wiedza teoretyczna posłużyła jako fundament do projektowania i przeprowadzania eksperymentów. Praca ta ma na celu nie tylko poszerzenie zrozumienia związku między teoretycznymi aspektami procesów ETL, a praktycznymi zastosowaniami, ale także dostarczenie konkretnych danych dotyczących wydajności SQL i HiveQL w różnych kontekstach [2, 8, 9, 12, 13].

2. Definicja ETL

Procesy ETL, to procesy związane z przetwarzaniem danych, które polegają na pobraniu danych z różnych źródeł, ich przekształceniu oraz załadowaniu do docelowego systemu przechowywania, co pozwala na ich łatwiejsze zarządzanie, analizowanie oraz udostępnianie. [2]. Schemat obrazujący proces ETL przedstawiono na Rysunku 1.



Rysunek 1: Schemat procesu ETL.

Ekstrakcja polega na pobieraniu danych z różnorodnych źródeł, takich jak bazy danych, pliki tekstowe, CRM, SCM, ERP i inne aplikacje biznesowe. Etap ten musi być skalowalny i elastyczny, aby dostosować się do zmieniających się wymagań. Transformacja przekształca dane, aby były zgodne z wymaganiami systemu docelowego. Proces ten może obejmować czyszczenie danych, konwersję typów danych, normalizację, mapowanie wartości, filtrowanie, agregację i scalanie danych. Ostatni etap, ładowanie, polega na przekazaniu danych do systemu docelowego, gdzie są one indeksowane, sortowane i przygotowane do analizy [7-9].

3. Opis środowiska badawczego

Procesy ETL można realizować za pomocą wielu narzędzi. W badaniu wykorzystane zostały takie platformy jak Hadoop, Apache Hive, Cloudera, SQL Server i SQL Server Management Studio. Oraz język SQL i HiveQL.

3.1. Narzędzia

1. Hadoop to otwartoźródłowy framework, stworzony do przetwarzania i przechowywania dużych zbiorów danych na klastrach komputerów. Składa się z dwóch głównych komponentów: Hadoop Distributed File System (HDFS) do przechowywania danych na wielu maszynach i MapReduce do przetwarzania danych w sposób równoległy i rozproszony [4, 13].
2. Apache Hive, opracowany przez Apache Software Foundation, jest narzędziem do tworzenia, implementacji i zarządzania magazynami danych na platformie Hadoop. Obsługuje przetwarzanie danych w sposób rozproszony na wielu węzłach, a także różne formaty danych [1].
3. Cloudera to dystrybucja Hadoop oferująca różne narzędzia do zarządzania i przetwarzania danych. Wśród nich znajdują się Cloudera Data Platform (CDP) do zarządzania danymi na różnych platformach, Cloudera Data Engineering (CDE) do tworzenia, wdrażania i zarządzania procesami ETL, a także Cloudera Manager do zarządzania i monitorowania klastrów Hadoop [3].
4. SQL Server, zaawansowany system zarządzania relacyjnymi bazami danych firmy Microsoft, używany jest do przechowywania, manipulowania i zarządzania danymi. Sprawdza się zarówno w procesach szybkiego przetwarzania transakcji (OLTP), jak i skomplikowanej analizy danych (OLAP) [2, 5, 8].
5. SQL Server Management Studio (SSMS) to narzędzie do zarządzania infrastrukturą SQL. Wykorzystywane jest do zarządzania danymi, indeksami, użytkownikami i ich rolami, a także do tworzenia zapytań, debugowania, tworzenia i zarządzania schematami oraz instancjami serwerów [2, 5, 8].

3.2. Języki

SQL (Structured Query Language) jest językiem używanym do interakcji z relacyjnymi bazami danych. Jego podstawową strukturą są zapytania, które służą do wykonywania określonych działań na danych. SQL jest podzielony na cztery główne podzbiory: DDL (Data Definition Language), DML (Data Manipulation Language), DQL (Data Query Language) i DCL (Data Control Language), każdy odpowiedzialny za różne operacje na danych, od definicji i modyfikacji struktury bazy danych, poprzez manipulację danymi, po zarządzanie uprawnieniami dostępu. Język SQL posiada również funkcje agregujące oraz różne słowa kluczowe i operatory, które umożliwiają tworzenie skomplikowanych zapytań [5, 6].

HiveQL, język zapytań wysokiego poziomu, został stworzony do współpracy z Apache Hive. Jest strukturalnie i składniowo podobny do SQL, chociaż nie obsługuje wszystkich funkcji SQL, takich jak operacje modyfikacji danych czy transakcje. Natomiast posiada unikalne funkcje, które go wyróżniają. Klauzule takie jak PARTITION BY i CLUSTER BY, które umożliwiają partycjonowanie i klastrowanie danych, są szczególnie ważne dla efektywnego przetwarzania dużych zbiorów danych, zgodnie z modelowaniem danych w ekosystemie Hadoop. HiveQL, oparty na MapReduce, umożliwia efektywne przetwarzanie dużych zbiorów danych w rozproszonych środowiskach [1, 10, 11].

3.3. Platforma systemowa i sprzętowa

Badania przeprowadzono w dwóch hermetycznych środowiskach testowych z użyciem maszyn wirtualnych VMware. Pierwsze środowisko skupiało się na języku SQL i wykorzystywało Microsoft SQL Server 2019 Express oraz SQL Server Management Studio do zarządzania komponentami SQL Server. Środowisko testowe zostało postawione na systemie operacyjnym Linux Red Hat 6 64-bit uruchomionym na maszynie wirtualnej ze 150 GB dysku twardego NVMe M2, 3-rdzeniowym procesorem o taktowaniu 3.8 GHz (5.1 GHz w boost) i 16 GB pamięci RAM o taktowaniu 3600 MHz. Natomiast drugie środowisko testowe utworzono dla HiveQL, języka zapytań wzorowanym na SQL, zaprojektowanym dla środowiska Hadoop. Wykorzystano tutaj dystrybucję Hadoop od Cloudera w wersji 2.5.0, która jest zoptymalizowana i łatwa w obsłudze. Środowisko to było uruchomione na identycznym sprzęcie co środowisko SQL. Obie konfiguracje umożliwiały efektywne przeprowadzenie badań i porównanie wydajności obu środowisk w procesach ETL.

4. Struktura bazy danych

W badaniu utworzona została baza danych typu Data Warehouse, jaką jest AdventureWorksDW. Baza została zaprojektowana zgodnie z modelem Star Schema. Baza składa się z tabel wymiarów (Dim) i faktów (Fact), które są powiązane relacjami. Tabele wymiarów zawierają atrybuty służące do analiz, np. informacje o klientach. Tabele faktów natomiast, zawierają metryki poddawane analizie, np. dane o sprzedaży. W środowisku SQL Server tabele wymiarów i faktów są połączone za pomocą kluczy, tworząc strukturę gwiazdy. W środowisku Hadoop, nie ma bezpośrednich połączeń pomiędzy tabelami, gdyż Hive nie obsługuje takich mechanizmów relacyjnych jak klucze główne i klucze obce [8].

Baza danych AdventureWorksDW, powstała na podstawie relacyjnej bazy danych AdventureWorks2019, przedstawiającej działalność fikcyjnej firmy. Zawiera ona szczegółowe dane dotyczące procesów biznesowych firmy, reprezentowane przez szereg powiązanych tabel.

5. Implementacja

5.1. Różnice w implementacji zapytań SQL i HiveQL

Z faktu, iż HiveQL bazuje na podstawach języka SQL, ich składnia w zapytaniach jest niemal identyczna. Jednakże przy implementacji zapytań, jakie wykorzystano w badaniu, występuj pewne różnice.

W SQL, aby uniknąć niejednoznaczności i błędów wynikających z użycia słów kluczowych jako nazw kolumn, umieszcza się nazwy kolumn w nawiasach kwadratowych []. Natomiast w ekosystemie Hadoop, używając HiveQL, nazwy kolumn w zapytaniach nie muszą być umieszczane w nawiasach kwadratowych [1, 2, 5, 8]. Przykład tej metody użycia przedstawiono na Listingu 1.

Listing 1: Użycie nawiasów kwadratowych w SQL

```
SELECT DISTINCT
    st.TerritoryID AS SalesTerritoryAlternateKey,
    st.[Name] AS SalesTerritoryRegion,
    st.CountryRegionCode AS SalesTerritoryCountry,
    st.[Group] AS SalesTerritoryGroup
FROM AdventureWorks2019.Sales.SalesTerritory AS st
```

W SQL podczas ładowania danych do tabeli, wykorzystuje się klauzulę INSERT INTO, natomiast HiveQl użyta musi być ta klauzula, poszerzona o słowo TABLE: INSERT INTO TABLE. Sposób w jaki dane są dodawane do tabeli może się różnić pomiędzy tymi dwoma językami [1, 2, 5, 8].

W SQL, do generowania unikalnych identyfikatorów dla wierszy (często dla kolumn będących kluczem głównym) używa się właściwości IDENTITY. SQL Server automatycznie zwiększa wartość w tej kolumnie, za każdym razem, gdy dodawany jest nowy wiersz. HiveQL nie ma natomiast takiej funkcji. W zastępstwie, wykorzysta została funkcja DENSE_RANK(), która przypisuje rangę dla każdego wiersza w partycji danych, sortując je na podstawie określonych kolumn, a przy pomocy klauzuli DISTINCT, otrzymane zapytania są unikalne. Zarówno SQL, jak i HiveQL korzystają z klauzuli DISTINCT do eliminowania powtarzających się wierszy [1, 2, 5, 8]. Listing 2 przedstawia przykład użycie DENSE_RANK() w HiveQL do wygenerowania unikalnej wartości.

Listing 2: Generowanie unikalnej wartości w HiveQL

```
INSERT INTO TABLE AdventureWorksDW.DimProduct
SELECT DISTINCT
    DENSE_RANK() OVER(ORDER BY p.ProductID) AS PoductKy,
```

Aktualna data i czas w języku SQL, są zwykle pobierane za pomocą funkcji GETDATE(). Działanie tego polecenia jest proste i zwraca aktualną datę i czas serwera SQL. Natomiast w wykorzystywanej w badaniu wersji HiveQL, dla uzyskania aktualnej daty i czasu, potrzebne jest wykorzystanie kombinacji dwóch funkcji: UNIX_TIMESTAMP() i FROM_UNIXTIME(). UNIX_TIMESTAMP() zwraca aktualny czas Unix (liczbę sekund od 1970-01-01 00:00:00 UTC), a FROM_UNIXTIME() konwertuje ten czas Unix na czytelny format daty [1, 2, 5, 8]. Przykład pobrania aktualnej daty w języku HiveQL przedstawiono na Listingu 3.

Listing 3: Pobranie aktualnej daty w HiveQL

```
WHERE edh.enddate IS NULL OR
edh.enddate >= date(from_unixtime(unix_timestamp()));
```

SQL zawiera wbudowane funkcje takie jak DATEPART(), DATENAME(), i DATEADD(), które są wygodne do manipulowania danymi daty i czasu. W HiveQL, takie funkcje nie są bezpośrednio dostępne. Dlatego, aby uzyskać takie same wyniki, wykorzystane zostały kombinacje funkcji, takich jak FROM_UNIXTIME(), UNIX_TIMESTAMP(), TO_DATE(), i CAST() [1, 2, 5, 8]. Przykład użycia tych funkcji w HiveQL, przedstawiono na Listingu 4. Ta część zapytania wyciąga z wartości w kolumnie „orderdate" kolejno numer dnia, numer miesiąca, numer dnia w roku oraz numer tygodnia w roku.

Listing 4: Manipulacja datą w HiveQl

```
CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(orderdate, 'yyyy-MM-dd'),
                   'd') AS INT) as daynumberofmonth,
CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(orderdate, 'yyyy-MM-dd'),
                   'D') AS INT) as daynumberofyear,
CAST(FROM_UNIXTIME(UNIX_TIMESTAMP(orderdate, 'yyyy-MM-dd'),
                   'w') AS INT) as weeknumberofyear
```

5.2. Metoda wykonania badań

W każdym z badanych scenariuszy, zapytanie zostało wykonane dziesięć razy w celu uzyskania wiarygodnej próbki czasu wykonania. W celu zminimalizowania wpływu innych czynników na wyniki, przed każdym wykonaniem zapytania, odpowiednie tabele zostały oczyszczone za pomocą polecenia TRUNCATE TABLE. Dodatkowo w środowisku SQL Server, pamięć podręczna i bufor danych zostały oczyszczone za pomocą poleceń DBCC FREEPROCCACHE i DBCC DROPCLEANBUFFERS, aby upewnić się, że żadne wcześniejsze zapytania nie wpłyną na wyniki. W przypadku HiveQL, wyczyszczone zostały tabele, ale z powodu braku funkcji do czyszczenia pamięci podręcznej i bufora danych, tego kroku nie wykonano [5].

5.3. Scenariusze badawcze

W scenariuszu pierwszy, wyekstrahowano dane do dwóch tabel wymiaru, pochodzące z 5 tabel z bazy danych AdventureWorks2019. W tabelach docelowych otrzymano kolejno 37 i 504 rekordy.

W scenariuszu drugim do ekstrakcji użyto 15 tabel z AdventureWorksDW i 1 tabeli, która została utworzono na początku tego scenariusza i osadzona w AdventureWorksDW. Dane zostały załadowane do 3 tabel wymiaru w AdventureWorksDW, które otrzymały kolejno: 683, 10 i 20058 rekordów.

W scenariuszu trzecim dane są pobierane z 19 tabel z AdventureWorks2019. Trzy docelowe tabele wymiaru umieszczone w AdventureWorksDW otrzymują kolejno: 290, 1530 i 1127 rekordów.

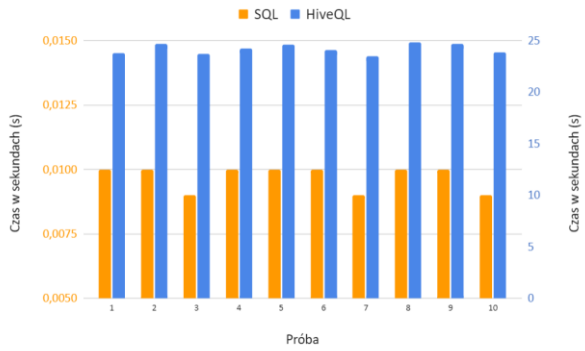
W scenariuszu czwartym uzupełniono tabelę faktu w AdventureWorksDW. Dane wyekstrahowano z 5 tabel z bazy danych AdventureWorks2019 i 5 tabel z AdventureWorksDW. Do tabeli faktu załadowano 7789440 rekordów.

W scenariuszu piątym również została utworzona tabela faktu. Dane do niej zostały pobrane z 7 tabel z bazy danych AdventureWorks2019 i 5 tabel z AdventureWorksDW. Do tabeli docelowej w wyniku zapytania załadowano 7730944 rekordów.

6. Wyniki

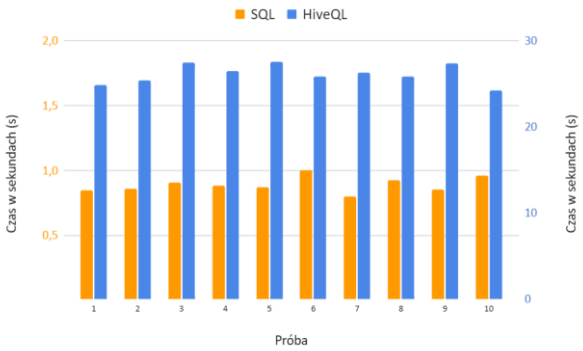
6.1. Analiza i porównanie wyników badań

W pierwszym scenariuszu, zapytanie SQL wykonuje się niezwykle szybko, potrzebując jedynie 0,01 sekundy, podczas gdy HiveQL wymaga znacznie więcej czasu - aż 24,2 sekundy. Jest to ogromna różnica, z HiveQL wykonującym to samo zapytanie aż 2420 razy wolniej niż SQL. Zestawienie wyników z tego scenariusza pokazuje Rysunek 2.



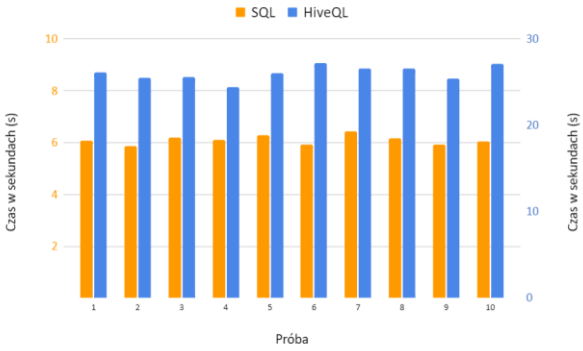
Rysunek 2: Zestawienie wyników z scenariusza pierwszego.

W drugim scenariuszu, SQL nadal przewyższa HiveQL, choć różnica nie jest już tak drastyczna. SQL kończy zadanie w ciągu 0,89 sekundy, podczas gdy HiveQL potrzebuje 26,13 sekundy do wykonania tego samego zadania. Mimo, iż różnica jest mniejsza, HiveQL jest wciąż około 29 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 3.



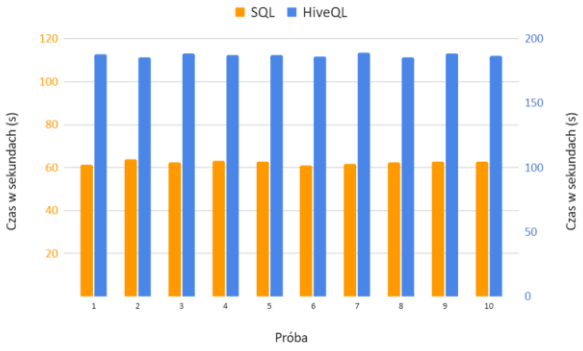
Rysunek 3: Zestawienie wyników scenariusza drugiego.

W trzecim scenariuszu, SQL nadal zdecydowanie przewyższa HiveQL. Zapytanie SQL kończy się w ciągu 6,1 sekundy, a zapytanie HiveQL wymaga 26,04 sekundy. W tym przypadku, HiveQL jest około 4 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 4.



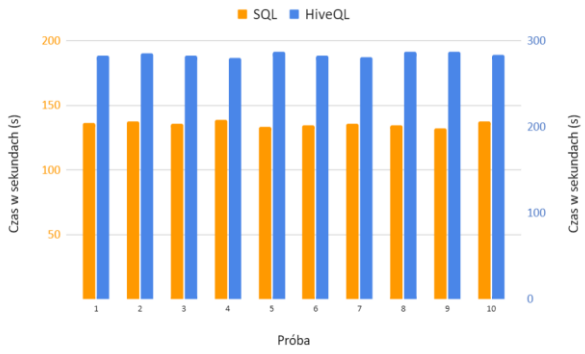
Rysunek 4: Zestawienie wyników z scenariusza trzeciego.

Czwarty scenariusz pokazuje jeszcze bardziej zauważalne różnice. SQL kończy zadanie w ciągu 62,3 sekundy, a HiveQL potrzebuje aż 187,33 sekundy na wykonanie tego samego zadania. To oznacza, że HiveQL jest około 3 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 5.



Rysunek 5: Zestawienie wyników z scenariusza czwartego.

W scenariuszu piątym, SQL wykonuje zadanie w ciągu 135,66 sekundy, podczas gdy HiveQL wymaga 284,19 sekundy. HiveQL jest w tym przypadku około 2 razy wolniejszy. Zestawienie wyników z tego scenariusza pokazuje Rysunek 6.



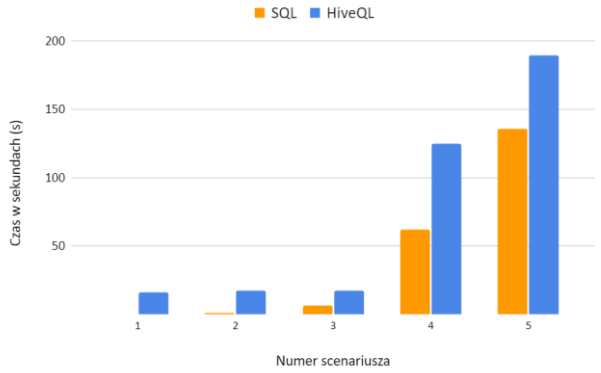
Rysunek 6: Zestawienie wyników z scenariusza piątego.

Dla każdego scenariusza wyliczono średnią z czasów otrzymanych z każdej próby poszczególnego scenariusza. Wyniki przedstawiono w Tabeli 1.

Tabela 1: Średni czas wykonania scenariuszy w 10 próbach

Scenariusz	Czas SQL (s)	Czas HiveQL (s)
1	0,01	24,2
2	0,89	26,13
3	6,1	26,04
4	62,3	187,33
5	135,66	284,19

Dane z Tabeli zostały zestawione w postaci wykresu, na Rysunku 7.



Rysunek 7: Porównanie średniego czasu wykonania scenariuszy dla SQL i HiveQL.

Podsumowując, wyniki testów wydajności jasno pokazują, że SQL jest znacznie szybszy w wykonaniu zapytań dla wszystkich pięciu scenariuszy. Różnice w wydajności między SQL, a HiveQL są szczególnie wyraźne dla prostszych zapytań, a złożoność zapytania wydaje się zwiększać wydajność HiveQL w porównaniu do SQL, choć nadal pozostaje on znacznie wolniejszy. Te wyniki podkreślają znaczenie dobrego zrozumienia narzędzi, które wybieramy do przetwarzania i analizy naszych danych, a także tego, jak złożoność zapytania może wpływać na wydajność tych narzędzi.

7. Dyskusja

W zarządzaniu danymi kluczową rolę pełnią procesy ETL, które umożliwiają przetwarzanie i analizę dużych zbiorów danych. Dwa popularne w tym zakresie języki to SQL i HiveQL, oba posiadające swoje unikalne zalety i wady.

SQL, będący stabilnym standardem branżowym, jest wszechstronny dzięki kompatybilności z wieloma systemami zarządzania bazami danych. Wykazuje efektywność w obsłudze różnych zapytań, szczególnie na mniejszych zestawach danych. Dodatkowo, doskonale integruje się z innymi technologiami i narzędziami analitycznymi. Wśród jego wad należy wymienić koszty związane z niektórymi systemami zarządzania bazami danych oraz potencjalne problemy z przetwarzaniem i analizą bardzo dużych zbiorów danych.

Z kolei HiveQL, część ekosystemu Hadoop, wyróżnia się skalowalnością umożliwiającą obsługę Big Data. Jest projektem open source, co minimalizuje bezpośrednie koszty, chociaż niektóre dystrybucje Hadoop mogą wiązać się z kosztami licencyjnymi. HiveQL jest podobny do SQL, co ułatwia naukę tego języka, ale może być trudniejszy w zarządzaniu i utrzymaniu, szczególnie dla osób bez doświadczenia. Przetwarzanie danych jest wolniejsze niż w SQL dla prostych zapytań i na mniejszych zestawach danych.

W kontekście operacji na pojedynczych wierszach, SQL umożliwia precyzyjne manipulowanie danymi dzięki funkcjom takim jak UPDATE, DELETE, INSERT, skomplikowanym funkcjom skalarnym oraz pętlom i instrukcjom warunkowym. HiveQL, zorientowany na przetwarzanie dużych zestawów danych w sposób równoległy, nie oferuje tych samych możliwości. W tym języku zmiany w danych są zazwyczaj dokonywane przez przetwarzanie całego zestawu danych, a wyniki są zapisywane do nowego zestawu danych.

W artykule Data Processing in Hive vs. SQL Server: A comparative analysis in the query performance, autorzy przeprowadzają badanie przy użyciu Hive oraz SQL Server. Badanie to wykazuje, że SQL Server jest wydajniejszy przy wykonywaniu operacji w czasie rzeczywistym. Natomiast Hive jest efektywniejszy pod względem czasu wykonania zapytania, wraz ze wzrostem rozmiaru danych do przetworzenia. Jak piszą autorzy, Hive jest odpowiedni przy przetwarzaniu dużej ilości rozległych zbiorów danych, które mogą obejmować nawet setki tysięcy maszyn [14].

8. Wnioski

Podczas testów wydajności, SQL okazał się szybszy we wszystkich pięciu scenariuszach testowych, szczególnie dla mniej skomplikowanych zapytań. Natomiast wydajność języka HiveQL, zaczęła wzrastać wraz ze złożonością zapytań oraz przy przetwarzaniu większej ilości danych. Przy wyższej złożoności zapytań, czas wykonania zapytania w HiveQL zaczyna się zbliżać do czasu

wykonania zapytania w SQL, co sugeruje, że HiveQL może być bardziej odpowiedni w kontekście przetwarzania bardzo dużych zbiorów, sięgających terabajtów, a nawet petabajtów danych Wybór między SQL a HiveQL uzależniony powinien być od specyfiki zadania i charakterystyki danych.

Literatura

[1] E. Capriolo, D. Wampler, J. Rutherglen, Programming Hive: Data Warehouse and Query Language for Hadoop, O'Reilly Media, 1st edition, 2012.

[2] J. Caserta, R. Kimball, The Data Warehouse ETL Toolkit., Wiley, 2004.

[3] Cloudera Data Platform, https://www.cloudera.com/products/cloudera-data-platform.html, [25.05.2023].

[4] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM 51(1) (2008) 107-113, https://doi.org/10.1145/1327452.1327492.

[5] B. Karwin, SQL Antipatterns: Avoiding the Pitfalls of Database Programming, Pragmatic Programmers LLC, The 1st edition 2017.

[6] P. Mellor, SQL and Relational Theory: How to Write Accurate SQL Code, O'Reilly Media Inc., 2011.

[7] B. Oliveira, O. Belo, J. Caldeira, A Systematic Literature Review on Big Data Extraction, Transformation and Loading (ETL), Proceedings of the 2021 Computing Conference Volume 2 held virtually (2021) 308-324, https://doi.org/10.1007/978-3-030-80126-7_24.

[8] A. Pelikant, Hurtownie danych. Od przetwarzania analitycznego do raportowania, Wydanie II, Helion, 2021.

[9] A. Simitsis, P. Vassiliadis, T. Sellis, Optimizing ETL processes in data warehouses, 21st International Conference on Data Engineering (ICDE'05), Tokyo, Japan (2005) 564-575, https://doi.org/10.1109/ICDE.2005.103.

[10] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, Hive - a Petabyte Scale Data Warehouse using Hadoop, 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), Long Beach, CA USA (2010) 996-1005, https://doi.org/10.1109/ICDE.2010.5447738.

[11] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, Hive: a warehousing solution over a map-reduce framework, Proceedings of the VLDB Endowment 2(2) (2009) 1626–1629, https://doi.org/10.14778/1687553.1687609.

[12] T. White, Hadoop: The definitive guide, O'Reilly Media Inc., 2012.

[13] P. C. Zikopoulos, C. Eaton, Understanding big data: Analytics for enterprise class Hadoop and streaming data, McGraw-Hill Osborne Media, 2011.

[14] N. Ahmed, S. Ahamed, J. I. Rahim, Data Processing in Hive vs. SQL Server: A comparative analysis in the query performance, 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences, Bangkok, Thailand (2017) 1-5, https://doi.org/10.1109/icetss.2017.8324202.