OPTIMASI *QUERY* RDF MELALUI TRANSFORMASI KOLUMNAR PARQUET MENGGUNAKAN APACHE SPARK

Astria Febrian Anggraini[1], Nadya RudieSucipto[2], Master Edison Siregar[3]

[1,2]Informatika, Sains dan Teknologi, Universitas Pradita, [3] Universitas Pradita, Indonesia
[1]astria.febrian@student.pradita.ac.id, [2]nadya.rudie@student.pradita.ac.id, [3]edison.siregar@pradita.ac.id

Abstrak

Perkembangan teknologi *Semantic Web* mendorong peningkatan signifikan dalam produksi metadata terstruktur yang direpresentasikan menggunakan *Resource Description Framework* (RDF). Seiring dengan pertumbuhan volume data RDF yang semakin besar, proses *querying* terhadap dataset RDF dalam format teks seperti *N-Triples* menghadapi berbagai kendala performa, terutama akibat mekanisme *full table scan* pada pemrosesan *query*, terutama saat *query* bersifat selektif. Kondisi ini menyebabkan peningkatan beban Input/Output (I/O), latensi eksekusi yang tinggi, serta pemanfaatan sumber daya komputasi yang kurang optimal. Meskipun berbagai pendekatan optimasi RDF telah dikembangkan, sebagian besar penelitian masih terfokus pada level algoritma *query* dan sistem *triple store* khusus, sehingga belum banyak mengeksplorasi pendekatan optimasi berbasis format penyimpanan kolumnar pada platform *distributed computing* modern seperti Apache Spark. Oleh karena itu, penelitian ini mengusulkan *Proof of Concept* (PoC) transformasi RDF ke format Parquet dengan *partitioning* berbasis predikat menggunakan Apache Spark untuk meningkatkan efisiensi *query* RDF. Metode yang digunakan adalah pendekatan eksperimental kuantitatif dengan membandingkan performa *query* pada dataset RDF sebelum dan sesudah penerapan optimasi. Dataset yang digunakan berasal dari DBpedia *Mapping-Based Objects* yang tersedia melalui DBpedia Databus dalam format *N-Triples* dan terdiri dari jutaan *triple* RDF yang merepresentasikan relasi antar entitas pada *knowledge graph* DBpedia. Proses optimasi dilakukan dengan mentransformasikan dataset ke format kolumnar Parquet serta menerapkan *partitioning* berbasis predikat pada platform Apache Spark. Evaluasi dilakukan melalui enam skenario *query* berbasis predikat tunggal, yaitu *team*, *careerStation*, *birthPlace*, *subdivision*, *country*, dan *starring*. Hasil pengujian mengonfirmasi bahwa secara arsitektural, pendekatan yang diusulkan mampu menghindari *full table scan* melalui mekanisme *partition pruning*, menghasilkan rata-rata peningkatan performa waktu eksekusi sebesar 99.87% pada skala data uji. Waktu eksekusi juga turun drastis dari rentang 224–234 detik menjadi sekitar 0.18–0.58 detik. Temuan awal ini membuktikan bahwa kombinasi format kolumnar dan *partitioning* memiliki potensi fundamental yang efektif. Penelitian ini meletakkan dasar eksperimental yang valid, yang ke depannya perlu dievaluasi lebih lanjut pada dataset berskala masif dan skenario *query* multi-join yang lebih kompleks untuk menguji batas skalabilitasnya.

Kata kunci: RDF, *query optimization, distributed database*, apache spark, parquet, *partitioning*

1. Pendahuluan

Perkembangan paradigma *Semantic Web* telah mendorong peningkatan signifikan dalam produksi metadata terstruktur di berbagai domain, seperti pemerintahan terbuka, bioinformatika, hingga integrasi data lintas institusi (Regino, Rossanez, Torres, & dos Reis, 2026). Representasi data dalam kerangka *Resource Description Framework* (RDF) memungkinkan integrasi informasi yang heterogen melalui model graf berbasis *Subject–Predicate–Object* (*triple*). Model ini mendukung interoperabilitas dan fleksibilitas skema sehingga menjadi pondasi utama dalam ekosistem *Linked Data* dan *knowledge graph modern* (Kalogeros, Gergatsoulis, Damigos, & Nomikos, 2023). Teknologi RDF saat ini juga banyak dimanfaatkan dalam pengembangan sistem pencarian semantik, integrasi data lintas organisasi, perkembangan *knowledge graph*, serta berbagai aplikasi berbasis kecerdasan buatan yang membutuhkan representasi hubungan antar entitas secara eksplisit. *Knowledge graph* telah menjadi komponen penting dalam berbagai aplikasi modern seperti sistem rekomendasi, pencarian semantik, serta integrasi data lintas domain karena kemampuannya merepresentasikan hubungan antar entitas secara eksplisit dalam bentuk graf semantik (Hogan et al., 2022).

Seiring dengan pertumbuhan volume data yang kini mencapai skala *terabyte* bahkan *petabyte*, tantangan serius muncul pada aspek penyimpanan dan efisiensi pemrosesan data RDF (Lim et al., 2022). Dalam praktiknya, dataset RDF sering didistribusikan dalam format teks mentah seperti *N-Triples* karena format ini sederhana, mudah diproses, serta menjadi standar umum dalam distribusi dataset *Linked Data*. Namun pendekatan penyimpanan berbasis baris (*row-based storage*) seperti *N-Triples* menimbulkan

inefisiensi komputasi ketika diterapkan pada lingkungan pemrosesan data berskala besar.

Permasalahan utama muncul ketika sistem harus mengeksekusi *query* selektif terhadap atribut tertentu, khususnya predikat. Pada format *N-Triples*, ketiadaan mekanisme indeks kolumnar memaksa mesin analitik melakukan pemindaian menyeluruh (*full table scan*) terhadap seluruh data untuk menemukan *triple* yang relevan. Kondisi ini menjadi semakin kompleks ketika jumlah *triple* meningkat secara signifikan, karena pemrosesan *query* RDF umumnya melibatkan banyak operasi *self-join* untuk memproses pola relasi yang kompleks (Ben Mahria, Chaker, & Zahi, 2021). Akibatnya, proses *querying* dapat menimbulkan beban Input/Output (I/O) yang tinggi, latensi eksekusi yang panjang, serta pemanfaatan sumber daya komputasi yang tidak optimal, terutama pada sistem analitik dan pendukung keputusan yang membutuhkan respons cepat (Lim et al., 2022).

Berbagai penelitian telah dilakukan untuk meningkatkan efisiensi pemrosesan data RDF. Ben Mahria dkk. melakukan studi empiris terhadap beberapa strategi penyimpanan RDF, yaitu *statement table*, *property table*, dan *vertical partitioning*. Hasil penelitian tersebut menunjukkan bahwa pendekatan *vertical partitioning* mampu memberikan performa *query* yang lebih baik dibandingkan dengan metode lainnya pada dataset RDF berskala besar yang mengandung jutaan *triple* (Ben Mahria et al., 2021).

Sagi dkk. mengidentifikasi ruang desain baru untuk representasi data RDF yang mempertimbangkan pola akses *query*, sehingga memungkinkan pemilihan strategi penyimpanan yang lebih sesuai dengan beban kerja nyata (Sagi, Lissandrini, Pedersen, & Hose, 2022). Penelitian lain mengusulkan kerangka pemrosesan *query* SPARQL berbasis Hadoop dan MapReduce untuk dataset RDF berskala besar. Hasil evaluasi menunjukkan bahwa teknik *partitioning* yang diusulkan mampu mengurangi jumlah operasi *join* antar node serta menurunkan waktu eksekusi *query* secara signifikan pada *benchmark* RDF seperti LUBM (Kumar & P.S., 2023). Selain itu Peng dkk. juga mengusulkan strategi partisi graf RDF berbasis motif kerja (Minimum Motif-Cut) yang memaksimalkan jumlah *query* SPARQL yang dapat dieksekusi dalam satu partisi dan terbukti mampu mengurangi join antar node dengan cara mengeksekusi lebih banyak *subquery* dalam satu partisi (Peng, Ji, Özsu, & Zou, 2024).

Penelitian terbaru juga mengkaji strategi partisi data RDF dalam lingkungan terdistribusi. Hassan dan Bansal memperkenalkan skema partisi relasional baru yang bernama *Property Table Partitioning* (PTP) yang membagi tabel properti menjadi beberapa tabel berdasarkan kelompok properti yang berbeda (Mahmudul Hasan & Bansal, 2023). Pendekatan ini bertujuan untuk meminimalkan ukuran data yang diproses serta mengurangi jumlah operasi join, sehingga mampu meningkatkan performa pemrosesan *query* pada sistem terdistribusi berbasis Apache Spark.

Studi lain oleh Palagin dkk. meneliti pengaruh partisi ontologi OWL/RDF/XML yang dikombinasikan dengan eksekusi paralel pada framework Apache Jena. Hasil penelitian menunjukkan bahwa kombinasi teknik partisi dan paralelisasi mampu mengurangi durasi eksekusi *query* kompleks hingga sekitar 45%. Namun penelitian tersebut juga menemukan bahwa fragmentasi data yang terlalu banyak justru dapat menurunkan efisiensi pemrosesan karena meningkatnya overhead koordinasi antar partisi (Palagin, Petrenko, Kaverinskiy, & Malakhov, 2025).

Selain itu, Elzein dkk. mengembangkan model pemrosesan *query* RDF bernama JQPro yang memanfaatkan teknik *hash-merge join* dalam sistem terdistribusi untuk meningkatkan efisiensi pemrosesan *query* pada dataset berskala besar. Hasil eksperimen menunjukkan bahwa pendekatan tersebut mampu meningkatkan performa eksekusi *query* hingga 87,77% lebih cepat dibandingkan beberapa sistem RDF yang ada seperti RDF-3X dan RDFox (Elzein et al., 2023).

Dalam konteks komputasi terdistribusi modern, pemanfaatan platform analitik seperti Apache Spark juga mulai banyak diteliti. Troullinou dkk. menunjukkan bahwa penerapan teknik *partition* pada sistem pemrosesan RDF berbasis Spark dapat mengurangi jumlah data yang perlu diakses selama proses *query* sehingga meningkatkan efisiensi pemrosesan data berskala besar (Troullinou, Agathangelos, Kondylakis, Stefanidis, & Plexousakis, 2024). Selain itu pendekatan lain dilakukan oleh Yamasaki dkk., yang mengembangkan metode partisi data RDF untuk pemrosesan *query* efisien menggunakan Spark SQL dengan memanfaatkan statistik data dan informasi beban kerja, sehingga memungkinkan pemuatan selektif partisi yang relevan saja (Yamasaki & Amagasa, 2023).

Meskipun berbagai pendekatan optimasi telah diusulkan, sebagian besar penelitian masih berfokus pada pengembangan sistem *triple store* khusus atau optimasi algoritma pemrosesan *query*. Pendekatan tersebut sering kali kurang terintegrasi secara optimal dengan ekosistem pemrosesan data terdistribusi yang umum digunakan dalam lingkungan *Big Data* modern. Di sisi lain, perkembangan teknologi penyimpanan kolumnar seperti Parquet menawarkan potensi optimasi yang signifikan melalui mekanisme pembacaan selektif kolom, kompresi data yang efisien, serta dukungan terhadap teknik *partition pruning* pada mesin analitik seperti Apache Spark.

Berdasarkan permasalahan tersebut, penelitian ini menyajikan sebuah *Proof of Concept* (PoC) sebagai studi awal pendekatan optimasi *querying* RDF melalui transformasi format penyimpanan dari *N-Triples* ke format kolumnar Parquet serta penerapan *partitioning* berbasis predikat pada

platform komputasi terdistribusi berbasis Apache Spark. Pendekatan ini bertujuan untuk mengurangi beban pembacaan data, menurunkan latensi eksekusi *query*, serta meningkatkan efisiensi pemanfaatan sumber daya komputasi. Fokus penelitian ini terletak pada pembuktian empiris terhadap peningkatan performa melalui serangkaian skenario pengujian terkontrol pada dataset RDF. Dengan demikian, penelitian ini diharapkan dapat meletakkan landasan empiris awal terkait kelayakan teknis integrasi teknologi *Semantic Web* dengan arsitektur *Big Data* yang lebih efisien dan skalabel di masa mendatang.

2. Metode

Penelitian ini menggunakan pendekatan eksperimental kuantitatif dengan tujuan untuk mengevaluasi peningkatan performa *query* pada dataset *Resource Description Framework* (RDF) setelah dilakukan transformasi arsitektur penyimpanan dan optimasi strategi partisi data. Pendekatan eksperimental dipilih karena penelitian ini bertujuan untuk membandingkan secara terukur kinerja sistem sebelum dan sesudah penerapan perlakuan, yaitu perubahan format penyimpanan dan strategi partisi predikat.

2.1 Objek dan Karakeristik Data

Objek penelitian berupa dataset percobaan yang digunakan sebagai representasi struktur RDF untuk menguji mekanisme optimasi *query*. Dataset yang digunakan berasal dari proyek DBpedia, yaitu *knowledge graph* terbuka yang mengekstraksi informasi terstruktur dari Wikipedia dalam format RDF. Dataset ini merupakan bagian dari DBpedia *Mapping-Based Objects* yang tersedia melalui DBpedia Databus dalam format *N-Triples* (DBpedia Association, 2022). Dataset ini berisi relasi antar entitas yang membentuk struktur *knowledge graph* berskala besar dan umum digunakan dalam penelitian *Semantic Web* serta pemrosesan RDF terdistribusi.

Dataset tersebut berukuran sekitar 3,5 GB dan terdiri dari 22,791,171 *triple* RDF yang merepresentasikan berbagai definisi ontologi dalam DBpedia. Setiap *triple* terdiri dari tiga komponen utama, yaitu *subject*, *predicate*, dan *object*, yang membentuk hubungan semantik antar entitas. Struktur ini memungkinkan analisis terhadap distribusi atribut *predicate* dalam dataset. Berdasarkan analisis distribusi data, beberapa predikat dengan frekuensi kemunculan tertinggi antara lain *team*, *careerStation*, *birthPlace*, *subdivision*, *country*, dan *starring*.

Dataset *Mapping-Based Objects* dipilih karena memiliki volume data yang jauh lebih besar dibandingkan dataset ontology DBpedia standar, sehingga lebih representatif untuk mengevaluasi performa pemrosesan RDF pada platform *distributed computing* seperti Apache Spark. Selain itu, dataset ini memiliki distribusi predikat yang beragam dengan jumlah triple mencapai jutaan baris, sehingga memungkinkan pengujian mekanisme optimasi *query* pada skenario data berskala besar. Dataset digunakan sebagai representasi struktur RDF dalam format teks berbasis baris (*row-based storage*) untuk menguji transformasi ke format penyimpanan kolumnar Parquet serta penerapan strategi *partitioning* berbasis predikat. Pendekatan ini relevan untuk mengevaluasi permasalahan *full table scan* yang umum terjadi pada proses *querying* RDF di lingkungan Big Data (Ben Mahria et al., 2021; Ryen, Soylu, & Roman, 2022).

Unit analisis dalam penelitian ini adalah *triple* RDF yang terdiri atas *subject*, *predicate*, dan *object*. Fokus optimasi diarahkan pada atribut *predicate*, karena dalam banyak skenario *query* RDF proses pencarian biasanya bersifat selektif terhadap jenis relasi tertentu. Oleh karena itu, atribut *predicate* digunakan sebagai dasar dalam penerapan strategi optimasi penyimpanan dan pengolahan data pada penelitian ini.

2.2 Tahapan Penelitian

Gambar 1. Diagram Alur Tahapan Penelitian

Tahapan penelitian yang digunakan dalam penelitian ini ditunjukkan pada Gambar 1. Alur penelitian yang meliputi proses sebagai berikut:

1. Analisis Masalah dan Perumusan Hipotesis
   Mengidentifikasi tantangan utama terkait performa *query* RDF berskala besar dan merumuskan hipotesis bahwa transformasi penyimpanan dan optimasi partisi dapat meningkatkan efisiensi eksekusi *query*.
2. Analisis Deskriptif Dataset
   Menghitung jumlah total triples dan jumlah *unique subjects* per predikat, serta rasio keduanya untuk memahami pola relasi (*one-to-one*, *one-to-many*, atau lainnya). Analisis ini digunakan untuk memperoleh gambaran empiris struktur internal dataset dan mengantisipasi potensi data skew sebelum strategi partisi diterapkan.
3. Pemilihan Predikat untuk Pengujian
   Berdasarkan frekuensi kemunculan tertinggi, enam predikat dipilih sebagai fokus pengujian: *team*, *careerStation*, *birthPlace*, *subdivision*, *country*, dan *starring*. Predikat ini dianggap representatif untuk mengevaluasi performa *query* pada dataset RDF.
4. Perancangan Eksperimen
   Menyusun skenario pengujian yang memungkinkan evaluasi performa *query* sebelum dan sesudah optimasi. Tahapan ini

mencakup desain metrik pengukuran, penentuan lingkungan komputasi, dan strategi validasi hasil untuk memastikan konsistensi dan akurasi pengukuran.

5. Validasi dan Pengendalian Variabel
   Mengontrol variabel bebas (format penyimpanan dan strategi partisi) dan variabel terikat (waktu eksekusi *query*) sehingga perubahan performa dapat diatribusikan secara langsung pada perlakuan optimasi. Validasi dilakukan melalui replikasi skenario pengujian dan penggunaan konfigurasi sistem yang konstan.
   Waktu eksekusi dicatat sebagai Woptimasi. Efisiensi peningkatan performa dihitung menggunakan Persamaan (1).

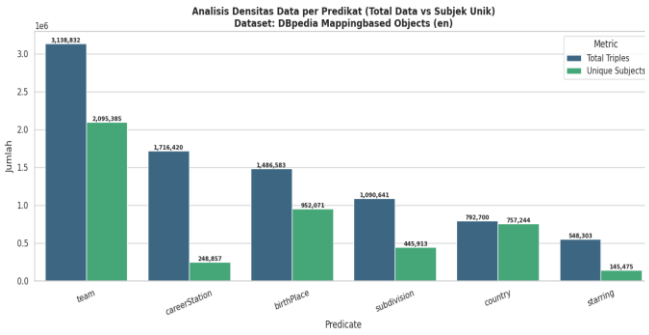
$$E = \frac{Wmentah - Woptimasi}{Wmentah} \times 100\% \quad (1)$$

   Keterangan:
   E = Efisiensi
   Wmentah = waktu eksekusi sebelum optimasi
   Woptimasi = Woptimasi adalah waktu eksekusi setelah optimasi

6. Interpretasi dan Dokumentasi Hasil
   Menyusun temuan penelitian dalam bentuk narasi, tabel, dan visualisasi untuk mendukung analisis performa *query*, sekaligus menarik kesimpulan terkait efisiensi strategi optimasi yang diterapkan.

2.3 Representasi Visual Dataset

Sebagai bagian dari pemahaman awal, distribusi total *triples* dan jumlah *unique subjects* per predikat divisualisasikan dalam bentuk diagram batang pada Gambar 2. Berdasarkan visualisasi data pada Gambar 2, terlihat bahwa distribusi predikat menunjukkan adanya data *skew*, di mana beberapa predikat memiliki jumlah triple jauh lebih besar dibandingkan predikat lainnya. Kondisi ini menjadi relevan dalam evaluasi strategi *partitioning* karena dapat memengaruhi distribusi beban pemrosesan pada lingkungan komputasi terdistribusi.



Gambar 2. Grafik Analisis Data per Predikat

3. Hasil dan Pembahasan

3.1 Transformasi Kolumar dan *Partitioning*

Pada penelitian ini dilakukan transformasi penyimpanan dataset RDF dari format teks *N-Triples* menjadi format kolumnar Parquet. Transformasi ini bertujuan untuk meningkatkan efisiensi pemrosesan *query* pada lingkungan komputasi terdistribusi berbasis Apache Spark. Format Parquet dirancang sebagai format penyimpanan kolumnar yang memungkinkan sistem membaca hanya kolom yang relevan dengan kebutuhan *query*. Dengan pendekatan ini, proses pemrosesan data tidak lagi memerlukan pembacaan seluruh atribut pada setiap baris data sebagaimana terjadi pada format berbasis baris seperti *N-Triples*.

Selain itu, format Parquet juga menyediakan mekanisme kompresi data yang lebih efisien sehingga ukuran penyimpanan dapat berkurang secara signifikan. Reduksi ukuran data ini secara langsung berkontribusi terhadap penurunan beban Input/Output (I/O) pada proses pembacaan data. Dalam lingkungan pemrosesan data terdistribusi seperti Apache Spark, pengurangan beban I/O menjadi faktor penting dalam meningkatkan performa eksekusi *query*.

Optimasi tambahan dilakukan melalui penerapan teknik *partitioning* berbasis atribut *predicate*. Dataset disimpan dalam struktur direktori hierarkis yang memisahkan data berdasarkan nilai *predicate*. Dengan struktur ini, ketika *query* dijalankan dengan filter terhadap predikat tertentu, Spark dapat menerapkan mekanisme *partition pruning* untuk melewati partisi data yang tidak relevan. Pendekatan ini memungkinkan sistem hanya membaca sebagian kecil data yang benar-benar dibutuhkan oleh *query* sehingga mengurangi beban komputasi secara signifikan.

Pendekatan ini sejalan dengan konsep optimasi penyimpanan RDF yang telah dibahas dalam penelitian sebelumnya, di mana strategi partisi data terbukti mampu meningkatkan efisiensi pemrosesan *query* pada dataset RDF berskala besar (Ben Mahria et al., 2021; Troullinou et al., 2024).

3.2 Validasi dan Pengendalian Variabel

Untuk memastikan bahwa peningkatan performa yang diperoleh berasal dari metode optimasi yang diusulkan, penelitian ini menerapkan pengendalian variabel secara sistematis. Eksperimen dilakukan dengan membandingkan dua kondisi dataset, yaitu dataset RDF dalam format *N-Triples* sebagai kondisi baseline dan dataset yang telah ditransformasikan ke format Parquet serta dipartisi berdasarkan predikat sebagai kondisi optimasi.

Pada kondisi baseline, Apache Spark membaca dataset menggunakan mekanisme *full table scan*. Dalam mekanisme ini, sistem harus memindai seluruh file data secara sekuensial untuk menemukan

triple yang relevan dengan kondisi *query*. Pendekatan ini menyebabkan waktu eksekusi *query* sangat dipengaruhi oleh ukuran total dataset dan distribusi data di dalamnya.

Setelah proses transformasi ke format Parquet dan penerapan *partitioning* berbasis predikat, struktur penyimpanan data berubah menjadi lebih terorganisasi. Ketika *query* dijalankan dengan filter predikat tertentu, Spark secara otomatis mengidentifikasi partisi data yang relevan dan melewati partisi lainnya. Validasi eksperimen dilakukan melalui beberapa langkah, yaitu menjalankan skenario *query* yang identik pada kedua kondisi dataset, menggunakan konfigurasi sistem yang sama selama proses pengujian, serta mencatat waktu eksekusi *query* sebagai metrik evaluasi utama. Pendekatan ini bertujuan untuk memastikan bahwa perbedaan performa yang diamati dapat diatribusikan secara langsung pada metode optimasi yang diusulkan.

3.3 Perbandingan Performa Waktu Eksekusi

Pengukuran performa dilakukan pada enam skenario *query* berbasis predikat dengan frekuensi kemunculan tertinggi pada dataset DBpedia *Mapping-Based Objects*, yaitu *team*, *careerStation*, *birthPlace*, *subdivision*, *country*, dan *starring*. Predikat tersebut dipilih karena merepresentasikan distribusi data RDF berskala besar dengan karakteristik relasi yang beragam.

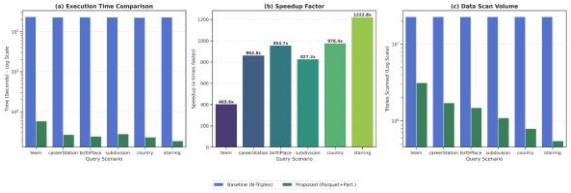
Eksperimen dilakukan dengan membandingkan performa *query* pada dua kondisi, yaitu dataset RDF mentah dalam format *N-Triples* sebagai kondisi baseline dan dataset yang telah ditransformasikan ke format Parquet serta dipartisi berdasarkan atribut predicate sebagai kondisi optimasi.

Tabel 1 menunjukkan perbandingan waktu eksekusi *query* pada kedua kondisi tersebut.

Tabel 1. Perbandingan waktu eksekusi *query* dan efisiensi performa

Skenario *Query*	***Waktu Mentah (detik)***	***Waktu Optimasi (detik)***	***Peningkatan Performa (%)***
Query team	234.9301	0.5821	99.75
Query careerStation	230.4730	0.2668	99.88
Query birthPlace	230.1693	0.2411	99.90
Query subdivision	229.5659	0.2775	99.88
Query country	224.6625	0.2301	99.90
Query starring	228.1654	0.1866	99.92

Berdasarkan hasil tersebut, seluruh skenario *query* mengalami penurunan waktu eksekusi setelah optimasi diterapkan. Pada kondisi baseline, waktu eksekusi *query* berada pada rentang 224 hingga 234 detik. Tingginya latensi ini menunjukkan bahwa Apache Spark harus melakukan *full table scan* terhadap keseluruhan dataset RDF dalam format *N-Triples* untuk setiap skenario *query*. Setelah transformasi ke format Parquet dan penerapan *partitioning* berbasis predikat, waktu eksekusi turun secara signifikan menjadi sekitar 0.18 hingga 0.58 detik. Visualisasi pada Gambar 3(a) mempertegas ketimpangan ini melalui skala logaritmik, yang memperlihatkan bagaimana waktu eksekusi pada kondisi optimasi mampu memangkas hambatan latensi I/O baseline yang masif hingga ke level sub-detik secara konsisten.



Gambar 3. Visualisasi Hasil Pengujian Performa *Query*: (a) Perbandingan Waktu Eksekusi, (b) Faktor Percepatan, dan (c) Volume Pemindaian Data

Penurunan waktu eksekusi ini menunjukkan bahwa pendekatan optimasi yang diusulkan mampu mengurangi volume data yang perlu dipindai selama proses *query*. Hal ini terjadi karena mekanisme *partition pruning* memungkinkan sistem untuk langsung mengakses partisi data yang relevan tanpa harus memproses keseluruhan dataset.

3.4 Analisis Peningkatan Performa

Hasil eksperimen menunjukkan bahwa transformasi dataset RDF ke format kolumnar Parquet yang dikombinasikan dengan teknik *partitioning* berbasis predikat mampu meningkatkan performa *query* secara sangat signifikan pada seluruh skenario pengujian. Pada kondisi baseline, setiap *query* memerlukan waktu lebih dari 220 detik karena Spark harus melakukan *full table scan* terhadap keseluruhan dataset RDF dalam format *N-Triples*.

Setelah optimasi diterapkan, waktu eksekusi turun drastis hingga berada pada rentang 0.18–0.58 detik. Penurunan waktu eksekusi ini menunjukkan bahwa Spark berhasil memanfaatkan mekanisme *partition pruning* untuk membaca hanya partisi data yang relevan dengan kondisi *query*.

Peningkatan performa tertinggi terjadi pada skenario *query starring* dengan efisiensi sebesar 99.92%, sedangkan peningkatan terendah terjadi pada *query team* sebesar 99.75%. Meskipun predikat team memiliki jumlah *triple* terbesar, sistem tetap mampu mempertahankan waktu eksekusi di bawah satu detik setelah optimasi diterapkan. Hasil ini menunjukkan bahwa kombinasi format penyimpanan kolumnar dan strategi *partitioning* efektif dalam mengurangi beban I/O serta volume data yang diproses pada lingkungan komputasi terdistribusi.

Secara keseluruhan, rata-rata peningkatan performa waktu eksekusi sebesar 99.87% pada

skenario pengujian yang dilakukan. Analisis lebih lanjut pada Gambar 3(b) menunjukkan bahwa efisiensi ini menghasilkan faktor percepatan (*speedup*) yang sangat masif, berkisar antara 403.6-1222.8 kali lipat pada skenario *query starring*. Lompatan performa yang ekstrem ini berkorelasi langsung dengan reduksi volume pemindaian data yang divisualisasikan pada Gambar 3(c). Hasil ini menunjukkan bahwa kombinasi transformasi penyimpanan kolumnar dan strategi *partitioning* mampu meningkatkan efisiensi pemrosesan *query* secara signifikan. Temuan ini juga konsisten dengan penelitian sebelumnya yang menyatakan bahwa optimasi struktur penyimpanan dan teknik partisi data dapat meningkatkan performa sistem pemrosesan RDF berskala besar (Ben Mahria et al., 2021; Elzein et al., 2023).

4. Kesimpulan

Penelitian ini telah berhasil mendemonstrasikan *Proof of Concept* (PoC) terkait pengaruh fundamental dari transformasi arsitektur penyimpanan terhadap efisiensi eksekusi *query* pada dataset *Resource Description Framework* (RDF). Berdasarkan hasil eksperimen pada dataset skala uji, terbukti bahwa format *N-Triples* memaksa sistem melakukan pemindaian menyeluruh (*full table scan*), yang secara otomatis membebani kinerja sistem pada *query* selektif.

Transformasi ke format penyimpanan kolumnar Parquet yang dipadukan dengan teknik *partitioning* berbasis atribut predikat terbukti secara teknis mampu mengaktifkan mekanisme *partition pruning* pada Apache Spark. Mekanisme ini berhasil mereduksi waktu eksekusi *query* secara signifikan dengan rata-rata peningkatan performa sebesar 99.87% dibandingkan pendekatan konvensional. Selain menurunkan latensi *query* dari rentang 224–234 detik menjadi sekitar 0.18–0.58 detik, pendekatan yang diusulkan juga mampu meningkatkan efisiensi pembacaan data dengan hanya memproses partisi yang relevan terhadap kondisi *query*. Hasil ini menunjukkan bahwa kombinasi format kolumnar dan strategi *partitioning* memiliki potensi yang sangat efektif untuk mendukung pemrosesan RDF pada ekosistem Big Data modern.

Meskipun hasil penelitian menunjukkan peningkatan performa yang sangat signifikan, penelitian ini masih terbatas pada *query* berbasis filter predikat tunggal dan belum mengevaluasi pola *query* SPARQL kompleks yang melibatkan operasi multi-join maupun *graph traversal*. Oleh karena itu, penelitian selanjutnya perlu mengeksplorasi evaluasi performa pada skenario *query* RDF yang lebih kompleks serta pada lingkungan komputasi multi-node untuk menguji batas skalabilitas arsitektur yang diusulkan.

Daftar Pustaka:

Ben Mahria, B., Chaker, I., & Zahi, A. (2021). An empirical study on the evaluation of the RDF storage systems. *Journal of Big Data*, *8*(1). https://doi.org/https://doi.org/10.1186/s40537-021-00486-y

DBpedia Association. (2022). DBpedia Mapping-based Objects Dataset. Retrieved May 12, 2026, from https://databus.dbpedia.org/dbpedia/mappings/mappingbased-objects

Elzein, N. M., Majid, M. A., Hashem, I. A. T., Ibrahim, A. O., Abulfaraj, A. W., & Binzagr, F. (2023). JQPro:Join Query Processing in a Distributed System for Big RDF Data Using the Hash-Merge Join Technique. *Mathematics*, *11*(5). https://doi.org/https://doi.org/10.3390/math11051275

Hogan, A., Blomqvist, E., Cochez, M., D'Amato, C., Melo, G. De, Gutierrez, C., … Zimmermann, A. (2022). Knowledge graphs. *ACM Computing Surveys*, *54*(4). https://doi.org/https://doi.org/10.48550/arXiv.2003.02320

Kalogeros, E., Gergatsoulis, M., Damigos, M., & Nomikos, C. (2023). Efficient query evaluation techniques over large amount of distributed linked data. *Information Systems*, *115*, 1–71. https://doi.org/https://doi.org/10.1016/j.is.2023.102194

Kumar, V. N., & P.S., A. K. (2023). An efficient and scalable SPARQL query processing framework for big data using MapReduce and hybrid optimum load balancing. *Data & Knowledge Engineering*, *148*(C), 102239. Retrieved from https://doi.org/10.1016/j.datak.2023.102239

Lim, J., Kim, Lee, H., Choi, D., Bok, K., & Yoo, J. (2022). An Efficient Distributed SPARQL Query Processing Scheme Considering Communication Costs in Spark Environments. *Applied Sciences (Switzerland)*, *12*(1). https://doi.org/https://doi.org/10.3390/app12010122

Mahmudul Hasan, & Bansal, S. (2023). S3QLRDF: distributed SPARQL query processing using Apache Spark—a comparative performance study. *Distributed and Parallel Database*, *41*, 191–231. https://doi.org/https://doi.org/10.1007/s10619-023-07422-4

Palagin, O., Petrenko, M., Kaverinskiy, V., & Malakhov, K. (2025). A Method for Enhancing the Efficiency of RDF/Xml-Structure Processing in the Apache Jena Semantic Web Framework. *Cybernetics and Systems Analysis*, *61*, 469–486. https://doi.org/https://doi.org/10.1007/s10559-025-00784-w

Peng, P., Ji, S., Özsu, M. T., & Zou, L. (2024). Minimum motif-cut: a workload-aware RDF graph partitioning strategy. *The VLDB Journal*, *33*, 1517–1542. https://doi.org/10.1007/s00778-024-00860-1

Regino, A. G., Rossanez, A., Torres, R. da S., & dos Reis, J. C. (2026). A Systematic Literature Review on RDF Triple Generation From Natural Language Texts. *Semantic Web: – Interoperability, Usability, Applicability*, *17*(1). https://doi.org/https://doi.org/10.1177/22104968251398355

Ryen, V., Soylu, A., & Roman, D. (2022). Building Semantic Knowledge Graphs from (Semi-)Structured Data: A Review. *Future Internet*, *14*(5), 1–24. https://doi.org/https://doi.org/10.3390/fi14050129

Sagi, T., Lissandrini, M., Pedersen, T. B., & Hose, K. (2022). A design space for RDF data representations. *VLDB Journal*, *31*(2), 347–373. https://doi.org/10.1007/s00778-021-00725-x

Troullinou, G., Agathangelos, G., Kondylakis, H., Stefanidis, K., & Plexousakis, D. (2024). DIAERESIS: RDF data partitioning and query processing on SPARK. *Semantic Web*, *15*(5), 1763–1789. https://doi.org/https://doi.org/10.3233/SW-243554

Yamasaki, K., & Amagasa, T. (2023). RDF Data Partitioning for Efficient SPARQL Query Processing with Spark SQL. In *Information Integration and Web Intelligence* (pp. 92–106). Springe, Cham. Retrieved from https://link.springer.com/chapter/10.1007/978-3-031-48316-5_12

Halaman ini sengaja dikosongkan