Pencarian Model Proses dalam *Workflow Repository* Berbasis Graph Database Menggunakan BPMN-Q

Galang Luhur Pekerti^{1,*}, Muhammad Ainul Yaqin², Suhartono³

Jurusan Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim, Indonesia

¹14650060@student.uin-malang.ac.id; ²yaqinov@ti.uin-malang.ac.id; ³suhartono@ti.uin-malang.ac.id

*corresponding author

INFO ARTIKEL

Sejarah Artikel

Diterima: 24 Maret 2021

Direvisi: 20 April 2021

Diterbitkan: 30 April 2021

Kata Kunci

Workflow Repository
Graph Database
BPMN-Q

ABSTRAK

Workflow repository menyimpan banyak *workflow*. *Workflow* yang disimpan akan dicari sesuai kebutuhan. Sebagian besar pencarian alur kerja berbasis teks. Penelitian ini menggunakan BPMN-Q untuk mencari BPMN pada *workflow repository* berbasis database grafik (NEO4J). BPMN yang disimpan dalam bentuk file XPDL agar dapat diparsing. Selanjutnya XPDL akan diparsing untuk disimpan dalam graph database menggunakan neo4j sebagai DBMS. Kemudian BPMN-Q dibuat menggunakan microsoft visio 2007. Lalu akan diparsing untuk diterjemahkan ke bahasa *cypher*. Selanjutnya *cypher* memproses dalam database NEO4J. Dan menghasilkan BPMN yang sesuai dalam bentuk daftar. BPMN dan BPMN-Q yang diekstrak harus disimpan dalam bentuk xpd. Penelitian ini memberikan skema database yang sesuai untuk penyimpanan *workflow* dalam bentuk grafik. Arsitektur ini juga menyimpan definisi alur kerja, seperti judul, penulis, dan lainnya. Performa penggunaan BPMN-Q pada *workflow repository* berbasis graph database adalah cukup baik dengan nilai *Precision* dari pengujian adalah 82.74%.

PENDAHULUAN

Workflow adalah rangkaian tindakan atau langkah-langkah yang digunakan dalam suatu proses, biasanya dijalankan oleh lebih dari satu pihak, dan menggunakan banyak sumber daya berbeda, yang biasanya dihasilkan menurut rangkaian aturan kerja [1]. Menurut Atmagi & Sarno [2], alur kerja merupakan hal yang menarik dalam perkembangan teknologi SOA. Melalui alur kerja ini, pengguna dapat mengubah alur kerja sistem. Namun, jumlah alur kerja yang digunakan perusahaan cukup besar. Dengan cara ini, setiap modul mungkin memiliki alur kerja yang serupa / berbeda dengan alur kerja lainnya. Oleh karena itu, repositori alur kerja diperlukan untuk menyimpan alur kerja. *Workflow repository* adalah sistem untuk menyimpan dan menjelaskan alur kerja proses bisnis di repositori. Menurut Chengfei Liu et al [3], fungsi repositori adalah untuk mendistribusikan data pada database yang menyimpan alur kerja yang dibuat dan digunakan oleh perusahaan. Manajer repositori menerapkan lapisan layanan kontrol untuk pemodelan, pengambilan dan pengelolaan objek dalam repositori di DBMS.

Pada penelitian Yan et al [4] semangat diperoleh melalui pengembangan sistem basis data informasi BPM. Pertama, basis data informasi menyediakan ruang penyimpanan dan menyederhanakan pencarian proses bisnis. Repositori ini kemudian dapat berfungsi untuk membentuk ulang model proses khusus industri. Penelitian mereka juga berfokus pada prosedur penggunaan kembali, seperti preset dalam pilihan komposisi dan penggunaan ontologi untuk menyesuaikan antar proses.

Dalam penelitian yang dilakukan oleh Astrininditya et al [5] Repositori alur kerja dibangun dengan 2 lapisan. Lapisan pertama menyimpan data alur kerja, dan lapisan kedua menyimpan metadata alur kerja. Metadata yang disusun dicatat dalam bentuk Resource Description Framework (RDF). Kemudian dibandingkan 2 tier dengan beberapa *query* menggunakan SQL dan SPARQL. Kueri SPARQL dalam repositori metadata dapat memberikan hasil yang lebih baik daripada data repositori. Namun, RDF harus mengandung subjek, predikat, dan objek.

Dalam penelitian yang dilakukan oleh Ardan et al [6] implementasi graph database dengan metode *Weighted Directed Acyclic Graph* (WDAG) digunakan dalam penentuan rute transportasi umum. Data jalur trayek angkot di Kota Malang dan Kota Batu disimpan dalam Graph Database. Titik pemberhentian yang dilalui disimpan dalam bentuk node. Sedangkan jalur angkot melaju disimpan dalam bentuk relasi atau edge. Dalam penelitian yang dilakukan oleh Abriani & Yaqin [7] metode *semantic similarity* digunakan untuk mengukur kemiripan makna antar kalimat. Pada tahap pertama, dilakukan penghitungan *word similarity* menggunakan pendekatan wu palmer, lin, dan path menggunakan WS4J dengan melibatkan tiga kriteria tingkat kemiripan yaitu mirip, sedang, dan tidak mirip. Setelah didapat nilai *word similarity*, kemudian dilakukan perhitungan *sentence similarity* dengan dua jenis bobot nilai yaitu (0,75, 0,25) dan (0,25, 0,75). Pada umumnya *workflow repository* menggunakan RDBMS (Relational Database Management System). RDBMS menyimpan data yang sangat terstruktur dalam tabel dengan kolom yang telah ditentukan sebelumnya dari jenis tertentu dan banyak baris dari jenis informasi yang sama. Relasi antar tabel pada RDBMS dihubungkan dengan primary key dan foreign key. *Query JOIN* digunakan untuk menggabungkan dua records dari 2 tabel atau lebih. Namun *sql query* tidak dapat mengambil nama relasi. Selain itu RDBMS memiliki schema yang kaku. Hal ini akan menghambat proses pencocokan struktural. Karena *workflow* itu sendiri memiliki schema yang flexible. Sehingga dalam penelitian ini digunakan graph database.

Graph Database Management System (selanjutnya disebut basis data grafik) yaitu sistem manajemen basis data online dengan metode create, read, update, dan delete (CRUD) untuk menampilkan pola data grafik. Database graph biasanya dibuat untuk diperuntukkan dengan sistem transaksional (OLTP). Cara ini, biasanya dioptimalkan untuk kemampuan transaksi dan memiliki desain untuk transaksi dan integritas operasional [8]. Graph database menyimpan data dalam bentuk node. Kemudian diberikan edge untuk relasi antar data. Sehingga schema yang dibuat lebih fleksibel daripada RDBMS. Selain itu graph database mempunyai bahasa *query* sendiri yang disebut *cypher*. Neo4j merupakan aplikasi database yang menggunakan graph database. Neo4j ini bekerja dengan cara menyimpan data ke node yang terhubung dengan relationship [9].

Pencarian pada *workflow* umumnya berdasarkan teks. Sedangkan *workflow* sendiri berupa grafik diagram. Maka pencarian berdasarkan teks tidak sesuai untuk pencarian *workflow*. Agar sesuai dengan bentuk dari *workflow* maka dalam penelitian ini menggunakan Business Process Models *Query language* yang disebut BPMN-Q. BPMN-Q adalah kueri visual berdasarkan notasi BPMN [10]. BPMN-Q dapat mengekspresikan *query* struktural dan menentukan apakah model proses yang diberikan (grafik) secara struktural mirip dengan grafik permintaan. *Query* pada BPMN-Q berupa notasi-notasi yang di ambil dari notasi pada BPMN itu sendiri. Kemudian diterjemahkan dalam bahasa *cypher* untuk diproses pada graph database. *Cypher* adalah bahasa kueri database grafik yang ekspresif (tapi tetap ringkas). Meskipun saat ini khusus untuk Neo4j, ini sangat dekat dengan kebiasaan kita dalam merepresentasikan grafik sebagai grafik, jadi sangat cocok untuk mendeskripsikan grafik secara terprogram [8]. *Cypher* akan mudah digunakan untuk pencarian *workflow*. Karena *cypher* merepresentasikan grafik. Selain itu *cypher* mampu memanggil nama relasi bahkan atribut dari relasi itu sendiri.

METODE

Gambar 1. Alur Penelitian

Pada Gambar 1 dijelaskan tentang alur penelitian. Dimulai dari BPMN yang disimpan dalam bentuk file XPDL agar dapat diparsing. Selanjutnya XPDL akan diparsing untuk disimpan dalam graph database. Pada penelitian ini menggunakan neo4j. Kemudian BPMN-Q dibuat menggunakan microsoft visio 2007. Lalu akan diparsing untuk diterjemahkan ke bahasa *cypher*. Selanjutnya *cypher* memproses dalam database NEO4J. dan dihasilkan BPMN yang sesuai dalam bentuk list.

Parsing XPDL

Pada tahap ini adalah proses pengambilan data dari file *.xpdL. Data yang akan diambil adalah Deskripsi, *Activity*, *Data Object*, *Transition*, *Data Transition*, *Data Store*, *Message Flow*, *Association*. Berikut penjelasannya :

- Deskripsi

Data pada deskripsi yang akan diambil yaitu author dan version. Kode xml deskripsi dapat dilihat pada Gambar 2.

```

<?xml version="1.0" encoding="utf-8"?>
<Package xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
Id="8e0c3517-6ac1-4325-aabb-6dfe68aa9fae"
Name="Diagram 1" xmlns="http://www.wfmc.org/2009/XPDL2.2">
  <PackageHeader>
    <XPDLVersion>2.2</XPDLVersion>
    <Vendor>Bizagi Process Modeler.</Vendor>
    <Created>2016-11-14T17:51:36.1406269+07:00</Created>
    <ModificationDate>2016-11-16T04:49:05.9378539+07:00</ModificationDate>
    <Description>Diagram 1</Description>
    <Documentation />
  </PackageHeader>
  <RedefinableHeader>
    <Author>MOY.CastleClash</Author>
    <Version>1.0</Version>
    <Countrykey>C0</Countrykey>
  </RedefinableHeader>
  <ExternalPackages />

```

Gambar 2. Deskripsi

- Activity

Data pada *activity* yang akan diambil yaitu Id dan Name. Kode xml activity dapat dilihat pada Gambar 3.

```
<Activities>
  <Activity Id="f4599dc2-e4d2-4fe9-90ff-83830dd9261e" Name="">
    <Description />
    <Event>
      <StartEvent Trigger="None" />
    </Event>
    <Documentation />
    <NodeGraphicsInfos>
      <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="30" Width="30"
        BorderColor="-10311914" FillColor="-1638505">
        <Coordinates XCoordinate="119" YCoordinate="65" />
      </NodeGraphicsInfo>
    </NodeGraphicsInfos>
    <ExtendedAttributes>
      <ExtendedAttribute Name="RuntimeProperties" value="{}" />
    </ExtendedAttributes>
  </Activity>
```

Gambar 3. Activity

- Data Object

Data pada data *object* yang akan diambil yaitu Id dan Name. Kode xml data object dapat dilihat pada Gambar 4.

```
<DataObjects>
  <DataObject Id="db5db842-34c0-41ff-a8ce-78986d7dce44" Name="Data Kesantrian">
    <Object>
      <Documentation />
    </Object>
    <NodeGraphicsInfos>
      <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="50" Width="40"
        BorderColor="-10066330" FillColor="-986896">
        <Coordinates XCoordinate="153" YCoordinate="844" />
      </NodeGraphicsInfo>
    </NodeGraphicsInfos>
    <DataField />
    <ExtendedAttributes />
  </DataObject>
```

Gambar 4. Data Object

- Flow

Data pada *flow* yang akan diambil yaitu Id, From dan To. Kode xml flow dapat dilihat pada Gambar 5.

```
<Transitions>
  <Transition Id="25e102a1-b126-4c37-af56-bdeb6cfff246" From="f4599dc2-e4d2-4fe9-90ff-83830dd9261e"
    To="9183ad4c-a125-4e1c-a665-c79d5202a694">
    <Condition />
    <Description />
    <ConnectorGraphicsInfos>
      <ConnectorGraphicsInfo ToolId="BizAgi_Process_Modeler" BorderColor="-16777216">
        <Coordinates XCoordinate="149" YCoordinate="80" />
        <Coordinates XCoordinate="195" YCoordinate="80" />
      </ConnectorGraphicsInfo>
    </ConnectorGraphicsInfos>
    <ExtendedAttributes />
  </Transition>
```

Gambar 5. Flow

- Flow data

Data pada *flow* data yang akan diambil yaitu Id, From dan To. Kode xml flow data dapat dilihat pada Gambar 6.

```

<DataAssociations>
  <DataAssociation Id="e21d1f0b-c043-44cb-9360-9c0de7fa03a2" From="2f0bd9fa-4ab2-42dc-9764-832ed5111fe0"
    To="9f80d5df-aa68-4aa7-a968-9b9986aacb41">
    <Description />
    <ConnectorGraphicsInfos>
      <ConnectorGraphicsInfo ToolId="BizAgi_Process_Modeler" BorderColor="-16777216" />
    </ConnectorGraphicsInfos>
    <ExtendedAttributes />
  </DataAssociation>

```

Gambar 6. Flow Data

- Data Store

Data pada *data store* yang akan diambil yaitu Id, Source dan Target. Kode xml data store dapat dilihat pada Gambar 7.

```

<MessageFlows>
  <MessageFlow Id="93192374-8283-414d-a40f-033d621fbb13"
    Source="60c3f02f-58b3-47fc-832e-195cebb0c33" Target="aabdc0e9-d078-4f02-b0d9-2d04c840d79d">
    <ConnectorGraphicsInfos>
      <ConnectorGraphicsInfo ToolId="BizAgi_Process_Modeler" BorderColor="-16777216">
        <Coordinates XCoordinate="650" YCoordinate="264" />
        <Coordinates XCoordinate="769" YCoordinate="264" />
        <Coordinates XCoordinate="769" YCoordinate="1064" />
      </ConnectorGraphicsInfo>
    </ConnectorGraphicsInfos>
    <ExtendedAttributes />
  </MessageFlow>

```

Gambar 7. Data Store

- Message Flow

Data pada *message flow* yang akan diambil yaitu Id dan Name. Kode xml message flow dapat dilihat pada Gambar 8.

```

<DataStores>
  <DataStore Id="62b3e5cd-d8d5-400c-b6b6-6999caabdac6" Name="" IsUnlimited="true">
    <Object>
      <Documentation />
    </Object>
  </DataStore>

```

Gambar 8. Message Flow

- Association

Data pada *association* yang akan diambil yaitu Id, Source dan Target. Kode xml association dapat dilihat pada Gambar 9.

```

<Associations>
  <Association Id="580e314c-c808-4da4-b5fb-79153050b394"
    Source="433eb629-eae3-4f7c-83e9-2353e1db4d" Target="db5db042-34c0-43ff-a8ce-7898bd7dce44" Name="">
    <ConnectorGraphicsInfos>
      <ConnectorGraphicsInfo ToolId="BizAgi_Process_Modeler" BorderColor="-16777216">
        <Coordinates XCoordinate="171" YCoordinate="943" />
        <Coordinates XCoordinate="171" YCoordinate="926" />
        <Coordinates XCoordinate="173" YCoordinate="926" />
        <Coordinates XCoordinate="173" YCoordinate="894" />
      </ConnectorGraphicsInfo>
    </ConnectorGraphicsInfos>
    <ExtendedAttributes />
  </Association>

```

Gambar 9. Association

Graph Database

Selanjutnya data yang sudah didapat akan disimpan ke dalam NEO4J. Skema yang digunakan pada graph database tentunya berbeda. Grap database menggunakan node untuk merepresentasikan sebuah data. Dan edge sebagai penghubungnya. Hal ini sesuai dengan bentuk visual dari BPMN. Skema yang digunakan pada penelitian ini diawali dengan node pertama yang selalu berlabel deskripsi. Ini bertujuan untuk menyimpan deskripsi dari sebuah BPMN. Kemudian dilanjutkan dengan edge yang diberi label: DIBUAT. Dan memiliki property: tgl_pembuatan, tgl_modifikasi. Jadi deskripsi dengan *workflow* akan tetap

terhubung, supaya saat pencarian dapat dipanggil juga. Selanjutnya node dan edge akan mengikuti notasi-notasi pada BPMN. Skema dapat dilihat pada Tabel 1.

Tabel 1. Skema BPMN dalam Neo4J

Node/Edge	Label	Properti	Notasi BPMN
<i>Node</i>	:Deskripsi	- Penulis - Versi	-
<i>Edge</i>	:DIBUAT	- tgl_pembuatan - tgl_modifikasi	-
<i>Node</i>	<i>Event</i>	- tipe	Notasi <i>Event</i>
<i>Node</i>	<i>Activity</i>	- nama - jenis	Notasi <i>Activity</i>
<i>Node</i>	<i>Gateway</i>	- jenis	Notasi <i>gateway</i>
<i>Edge</i>	Sesuai dengan jenis relasinya. Misal: - sequence flow = SEQUENCE	-	Notasi <i>flow</i> (relasi)
<i>Node</i>	Data	- nama - tipe	Notasi data

BPMN-Q & Cypher

Proses selanjutnya membuat BPMN-Q menggunakan microsoft office visio 2007. BPMN-Q yang telah dibuat disimpan dalam format vdx (.vdx). kemudian diparsing untuk diambil datanya. Berikut data yang akan diambil :

- Shape
Data pada *shape* yang akan diambil yaitu Id dan Master. Kode xml shape dapat dilihat pada Gambar 10.

```
<Shapes>
  <Shape ID='1' NameU='Start' Type='Shape' Master='0'>
    ...
  </Shape>
</Shapes>
```

Gambar 10. Shape

- Connect
Data pada *connect* yang akan diambil yaitu FronSheet dan ToSheet. Kode xml connect dapat dilihat pada Gambar 11.

```
<Connects>
  <Connect FromSheet='2' FromCell='BeginX' FromPart='9' ToSheet='1' ToCell='Connections.X2' ToPart='101' />
  <Connect FromSheet='2' FromCell='EndX' FromPart='12' ToSheet='4' ToCell='Connections.X4' ToPart='103' />
  <Connect FromSheet='8' FromCell='BeginX' FromPart='9' ToSheet='4' ToCell='Connections.X1' ToPart='100' />
  <Connect FromSheet='8' FromCell='EndX' FromPart='12' ToSheet='10' ToCell='Connections.X2' ToPart='101' />
  <Connect FromSheet='9' FromCell='BeginX' FromPart='9' ToSheet='4' ToCell='Connections.X2' ToPart='101' />
  <Connect FromSheet='9' FromCell='EndX' FromPart='12' ToSheet='11' ToCell='Connections.X2' ToPart='101' />
</Connects>
```

Gambar 11. Connect

- Master

Data pada master yang akan diambil yaitu Name. Kode xml master dapat dilihat pada Gambar 12.

```
<Masters>
  <Master ID='2' NameU='Path' Name='Path' Prompt='' IconSize='1' AlignName='2' MatchByName='0'
  IconUpdate='1' UniqueID='{315D6DA4-0003-0000-8E40-00608CF305B2}' BaseID='{C32DD7B7-C894-4085-
  8988-C28AB0F69116}' PatternFlags='0' Hidden='0'>
    <PageSheet LineStyle='0' FillStyle='0' TextStyle='0'>...
  </PageSheet>
  <Shapes>
    <Shape ID='5' NameU='Path.35' Type='Shape' LineStyle='6' FillStyle='6'
  TextStyle='6'>...
  </Shape>
  </Shapes>
  <Icon>...
  </Icon>
</Master>
</Masters>
```

Gambar 12. Master

Selanjutnya diterjemahkan ke *cypher*. Notasi BPMN-Q diterjemahkan seperti pada Tabel 2.

Tabel 2. Terjemahan notasi BPMN-Q ke *cypher*

Notasi BPMN-Q	<i>Cypher</i>
(a:activity{nama:"variabel"}) = variabel (a:activity) = @variabel
(e:event)
(e:event{tipe:"join"})
(e:event{tipe:"split"})
(e:event{tipe:"xor"})
(e:event{tipe:"or"})
(e:event{tipe:"and"})


```

MATCH      j=(e:event)-[r*2..]->(g:gateway)-->(a:activity)      where
e.nama="start"      or      g.nama="xor"      or      a.nama="piket"      or
a.nama="sarapan" return j

```

Gambar 16. Cypher

Hasil dari *cypher* Gambar 16 pada Gambar 17.

Gambar 17. Hasil query

Data *workflow* dan BPMN-Q

Dalam penelitian ini digunakan beberapa *workflow* penerimaan santri baru dari berbagai pondok pesantren dan beberapa prosedur lain yang sudah disimpan dalam format xpd. Penelitian ini menggunakan 35 *workflow*. Pada tiap *workflow* memiliki struktur yang berbeda. Untuk memudahkan pembacaan data *workflow*, struktur disusun sebagai berikut. Struktur disusun berdasarkan notasi *workflow*. Untuk notasi activity akan ditulis tanpa nama activity. Dan hanya dicantumkan jumlah activity yang dilewati sebelum menuju notasi berikutnya. Data struktur *workflow* dapat dilihat pada Tabel 3.

Tabel 3. Data *Workflow*

NO	KODE	Struktur
1.	WF01	
2.	WF02	

NO	KODE	Struktur
3.	WF03
4.	WF04
5.	WF05
6.	WF06
7.	WF07
8.	WF08
9.	WF09
10.	WF10
11.	WF11

NO	KODE	Struktur
12.	WF12	<pre> graph TD Start(()) --> N5[5] N5 --> D1{x} D1 --> N2_1[2] D1 --> D2{x} N2_1 --> D2 D2 --> N4[4] D2 --> D3{x} D3 --> N2_2[2] D3 --> D4{x} N2_2 --> D4 D4 --> N5_2[5] D4 --> N1[1] N5_2 --> N7[7] N7 --> D5{x} D5 --> N2_3[2] D5 --> D6{x} N2_3 --> D6 D6 --> N2_4[2] D6 --> N6[6] N2_4 --> D6 N1 --> End(()) N6 --> End </pre>
13.	WF13	<pre> graph TD Start(()) --> N7_1[7] N7_1 --> D1{x} D1 --> N4[4] D1 --> N3_1[3] N4 --> D2{x} D2 --> N1_1[1] D2 --> N3_2[3] N3_1 --> D3{x} D3 --> N2_1[2] D3 --> N2_2[2] N2_1 --> D4{x} D4 --> N2_3[2] N2_2 --> D4 N2_3 --> N1_1 N3_2 --> D5{x} D5 --> N1_2[1] D5 --> N3_3[3] N1_2 --> D6{x} D6 --> N2_4[2] D6 --> N7_2[7] N3_3 --> D7{x} D7 --> N2_5[2] D7 --> N6[6] N2_4 --> D8{x} D8 --> N2_6[2] D8 --> N6 N2_5 --> D8 N2_6 --> D8 N7_2 --> End(()) N6 --> End </pre>
14.	WF14	<pre> graph LR Start(()) --> N14[14] N14 --> End(()) </pre>

NO	KODE	Struktur
15.	WF15	<pre> graph LR Start(()) --> 6[6] 6 --> D1{x} D1 --> 2t[2] D1 --> 2b[2] 2t --> D2{x} D2 --> 4[4] 4 --> End1(()) 2b --> D3{x} D3 --> 11[11] 11 --> D4{x} D4 --> 2c[2] 2c --> D5{x} D5 --> 2d[2] 2d --> D6{x} D6 --> 4[4] 4 --> End2(()) </pre>
16.	WF16	<pre> graph LR Start(()) --> 5[5] 5 --> D1{x} D1 --> 2t[2] D1 --> 2b[2] 2t --> D2{x} D2 --> 7[7] 7 --> End1(()) 2b --> D3{x} D3 --> 10[10] 10 --> D4{x} D4 --> 2c[2] 2c --> D5{x} D5 --> 2d[2] 2d --> D6{x} D6 --> 7[7] 7 --> End2(()) </pre>
17.	WF17	<pre> graph LR Start(()) --> 19[19] 19 --> D1{x} D1 --> 1t[1] D1 --> 6[6] 1t --> D2{x} D2 --> 1c[1] 1c --> D3{x} D3 --> 1d[1] 1d --> D4{x} D4 --> 3[3] 3 --> End1(()) 6 --> D5{x} D5 --> 1b[1] 1b --> D6{x} D6 --> 1e[1] 1e --> D7{x} D7 --> 3[3] 3 --> End2(()) </pre>
18.	WF18	<pre> graph LR Start(()) --> 13[13] 13 --> D1{x} D1 --> 1[1] D1 --> 3[3] 1 --> D2{x} D2 --> 7[7] 7 --> End1(()) 3 --> D3{x} D3 --> 2[2] 2 --> D4{x} D4 --> 5[5] 5 --> D5{x} D5 --> 7[7] 7 --> End2(()) </pre>
19.	WF19	<pre> graph LR Start(()) --> 5[5] 5 --> D1{x} D1 --> 1t[1] D1 --> 1b[1] 1t --> D2{x} D2 --> 17[17] 17 --> End1(()) 1b --> D3{x} D3 --> 2[2] 2 --> D4{x} D4 --> 4[4] 4 --> D5{x} D5 --> 3[3] 3 --> D6{x} D6 --> 17[17] 17 --> End2(()) </pre>

NO	KODE	Struktur
20.	WF20	
21.	WF21	
22.	WF22	
23.	WF23	
24.	WF24	
25.	WF25	
26.	WF26	
27.	WF27	

NO	KODE	Struktur
28.	WF28
29.	WF29
30.	WF30
31.	WF31
32.	WF32

Kemudian dari *workflow* yang ada akan dicari menggunakan BPMN-Q. BPMN-Q yang digunakan terbagi menjadi 3 variasi, yaitu:

- 1) BPMN-Q dengan struktur linear. BPMN-Q dapat dilihat pada Tabel 4.

Tabel 4. BPMN-Q linear

NO	KODE	BPMN-Q
1.	LR01	
2.	LR02	
3.	LR03	

2) BPMN-Q dengan struktur cabang. BPMN-Q dapat dilihat pada Tabel 5.

Tabel 5. BPMN-Q cabang

NO	KODE	BPMN-Q
1.	BR01
2.	BR02
3.	BR03

3) BPMN-Q dengan struktur siklus. BPMN-Q dapat dilihat pada Tabel 6.

Tabel 6. BPMN-Q siklus

NO	KODE	BPMN-Q
1.	CC01

HASIL DAN PEMBAHASAN

Perhitungan akurasi dilakukan untuk mengetahui seberapa besar akurasi yang dihasilkan dari langkah-langkah implementasi database grafik saat mencari alur kerja menggunakan *query* visual pada repositori alur kerja. Dalam penelitian ini akurasi dihitung menggunakan *Precision* pada Persamaan 1, kemudian *Recall* pada Persamaan 2 dan *Accuracy* pada Persamaan 3. Efektivitas sistem temu kembali dapat diukur dengan menghitung nilai *Recall*, *Precision*, dan *Fallout* [11]. Namun, di antara metode tersebut, akurasi perhitungan adalah metode yang paling umum digunakan [11]. Akurasi menunjukkan seberapa akurat sistem mengembalikan informasi yang relevan kepada pengguna. *Recall* adalah nilai yang mewakili tingkat pengembalian dari hasil yang dikembalikan oleh sistem.

$$Precision = \frac{TP}{TP + FP} \dots\dots\dots (1)$$

$$Recall = \frac{TP}{TP + FN} \dots\dots\dots (2)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \dots\dots\dots (3)$$

TP (*True Prositif*) adalah Jumlah *workflow* yang sesuai dengan *query* BPMN-Q dan muncul pada hasil pencarian *workflow*. FP (*False Positif*) adalah semua *workflow* yang sesuai dengan *query* BPMN-Q. TN (*True Negative*) semua *workflow* yang tidak sesuai dengan *query* BPMN-Q. FN (*False Negative*) adalah *workflow* yang tidak sesuai dengan *query* BPMN-Q dan muncul pada hasil pencarian *workflow*. Keterangan juga ditampilkan pada Tabel 7.

Tabel 7. Pengertian TP, FP, TN dan FN

No	Variabel	Keterangan
1	TP (<i>True Prositif</i>)	Jumlah <i>workflow</i> yang sesuai dengan <i>query</i> BPMN-Q dan muncul pada hasil pencarian <i>workflow</i>
2	FP (<i>False Positif</i>)	semua <i>workflow</i> yang sesuai dengan <i>query</i> BPMN-Q
3	TN (<i>True Negative</i>)	semua <i>workflow</i> yang tidak sesuai dengan <i>query</i> BPMN-Q
4	FN (<i>False Negative</i>)	<i>workflow</i> yang tidak sesuai dengan <i>query</i> BPMN-Q dan muncul pada hasil pencarian <i>workflow</i>

Seperti yang ditunjukkan oleh data dalam tabel langkah pengujian, jumlah total alur kerja adalah 35. Untuk mengetahui tingkat akurasi yang digunakan, dilakukan beberapa simulasi dengan menggunakan *query* BPMN-Q yang berbeda. Simulasi dilakukan sebanyak 7 kali. Kemudian hitung *Precision*, *Recall*, dan *Accuracy* dari setiap simulasi secara terpisah, lalu cari rata-ratanya. Hasil pencarian ditunjukkan pada Tabel 8.

Tabel 8. Hasil pencarian dengan BPMN-Q

KODE QUERY BPMN-Q	JUMLAH WORKFLOW YANG SESUAI	KODE WORKFLOW YANG SESUAI	HASIL	TP	FP	TN	FN
LR01	35	WF01-WF35	WF03,WF05,WF06,WF08,WF09,WF10,WF15,WF16,WF17,WF18,WF20,WF21,WF22,WF23,WF24,WF25,WF26,WF27,WF28,WF29,WF30,WF31,WF32,WF33,WF34,WF35	35	0	0	0
LR02	2	WF12, WF22	WF12,WF22,WF34	2	1	32	0
LR03	24	WF01,WF02,WF04,WF05,WF09,WF11,WF12,WF13,WF15,WF16,WF17,WF18,WF19,WF20,WF21,WF22,WF23,WF	WF01,WF02,WF04,WF05,WF09,WF11,WF12,WF13,WF15,WF16,WF17,WF18,WF19,WF20,WF21,WF22,WF23,WF	24	0	11	0

KODE QUERY BPMN- Q	JUMLAH WORKFLOW YANG SESUAI	KODE WORKFLOW YANG SESUAI	HASIL	TP	FP	TN	FN
		24,WF25,WF31,WF32,WF33,WF34,WF35	24,WF25,WF31,WF32,WF33,WF34,WF35				
BR01	24	WF01,WF02,WF04,WF05,WF09,WF11,WF12,WF13,WF15,WF16,WF17,WF18,WF19,WF20,WF21,WF22,WF23,WF24,WF25,WF31,WF32,WF33,WF34,WF35	WF01,WF02,WF04,WF05,WF09,WF11,WF12,WF13,WF15,WF16,WF17,WF18,WF19,WF20,WF21,WF22,WF23,WF24,WF25,WF31,WF32,WF33,WF34,WF35	24	0	11	0
BR02	11	WF01,WF09,WF11,WF12,WF13,WF15,WF16,WF17,WF18,WF19,WF20	WF01,WF11,WF13,WF17,WF18,WF19,WF20	7	0	24	4
BR03	4	WF20,WF31,WF32,WF33	WF20,WF31,WF32,WF33	4	0	31	0
CC01	3	WF05,WF23,WF25	WF01,WF02,WF04,WF05,WF09,WF11,WF12,WF13,WF15,WF16,WF17,WF18,WF19,WF20,WF21,WF22,WF23,WF24,WF25,WF31,WF32,WF33,WF34,WF35	3	21	11	0

Dari hasil simulasi yang telah dilakukan. Didapatkan nilai TP, TN, FP, FN. Kemudian dilakukan perhitungan nilai *Precision*, *Recall*, *Accuracy*. Untuk nilai *Precision*, *Recall* dan *Accuracy* ditampilkan pada Tabel 9.

Tabel 9. Nilai *Precision*, *Recall* dan *Accuracy*

NO	KODE QUERY	PRECISION	RECALL	ACCURACY
1.	LR01	1	1	1
2.	LR02	0.666666667	1	0.971428571
3.	LR03	1	1	1
4.	BR01	1	1	1
5.	BR02	1	0.636363636	0.885714286
6.	BR03	1	1	1
7.	CC01	0.125	1	0.4

Pada Tabel 9 menunjukkan nilai *Precision*, *Recall*, *Accuracy* setiap query BPMN-Q yang disimulasikan pada *workflow repository*. Persentase *Precision*, *Recall*, dan *Accuracy* didapat dari hasil perhitungan masing-masing evaluasi dikalikan dengan 100. Dari Tabel 9 : Evaluasi *Precision*, *Recall* dan *Accuracy* dengan percobaan 7 query dapat diketahui tingkat *Precision* terendah terdapat pada query dengan kode “CC01” dengan nilai 0.125, kemudian *Precision* tertinggi terdapat pada query dengan kode “LR01, LR03, BR01, BR02, BR03”, dengan masing-masing nilai 1. Untuk *Accuracy* terendah pada query dengan kode “BR02” dengan nilai 0.885714286. Sedangkan untuk *Accuracy* tertinggi pada query dengan kode “LR01,

LR03, BR01, BR03” dengan nilai 1. Pada evaluasi *Recall* tertinggi terdapat pada *query* dengan kode “LR01, LR02, LR03, BR01, BR03, CC01” dengan nilai masing-masing 1. Sedangkan *Recall* terendah terdapat pada *query* dengan kode “BR02” dengan nilai 0.636363636. Selanjutnya nilai rata-rata dari semua simulasi yang sudah dilakukan ditampilkan pada Tabel 10.

Tabel 10. Nilai *Precision*, *Recall* dan *Accuracy* akhir

NO	Variabel	Nilai Rata-rata	Nilai Presentase
1.	<i>Precision</i>	0,827380952	82,74 %
2.	<i>Recall</i>	0,948051948	94,81 %
3.	<i>Accuracy</i>	0,893877551	89,39 %

Nilai persentase *Precision* dari semua *query* yang telah diuji coba adalah 83,74%. *Precision* menunjukkan ketepatan sistem dalam menemukan BPMN yang sesuai dengan *query* BPMN-Q. Nilai persentase *Accuracy* dari semua *query* yang telah diuji coba adalah 94,81%. *Accuracy* menunjukkan kedekatan hasil pencarian pada sistem dengan data aktual. Nilai persentase *Recall* dari semua *query* yang telah diuji coba adalah 89,39%. *Recall* menunjukkan tingkat perolehan hasil yang dikembalikan oleh sebuah sistem. Untuk penentuan performa BPMN-Q menggunakan nilai *Precision*. Karena pada kasus pencarian *workflow* dibutuhkan terjadinya TRUE POSITIF. Maka digunakanlah nilai *Precision* sebagai acuan performa BPMN-Q pada *workflow repository* berbasis graph database. Maka performa penggunaan BPMN-Q pada *workflow repository* berbasis graph database adalah baik. Karna nilai *Precision* dari pengujian adalah 82.74 %.

KESIMPULAN

Dalam *workflow repository* terdapat fitur pencarian. Pencarian ini digunakan untuk menemukan kembali *workflow* yang telah disimpan. Dengan menggunakan BPMN-Q memudahkan pengguna untuk membuat *query* yang sesuai dengan *workflow* yang dicari. Tidak hanya mencari menggunakan *query* melalui nama *workflow* tetapi juga bisa membuat *query* sesuai dengan struktur *workflow*. Penggunaan graph database (NEO4J) pada *workflow repository* memudahkan developer dalam mengembangkan aplikasi. Karna *workflow* mempunyai bentuk graph. Hal ini memudahkan developer dalam penyusunan skema database. Karna graph database sangat fleksibel dalam hal skema dan merepresentasikan graph pada database. Akan tetapi dibutuhkan sumber daya yang lebih tinggi untuk menjalankan graph database dengan jumlah data yang sama dengan database sql. Presentase *Precision*, *Recall*, *Accuracy* yang dihasilkan secara berurutan dari beberapa simulasi adalah 82.74 %, 94.81 %, 89.39 %. Ini menunjukkan bahwa *workflow repository* yang dibuat cukup baik dalam menemukan *workflow* yang sesuai dengan *query*.

REFERENSI

- [1] N. Palmer, “Workflow Management Coalition BT - Encyclopedia of Database Systems,” L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, p. 3550.
- [2] R. W. Atmagi and R. Sarno, “Penggunaan Workflow Repository dan Kombinasi Metode Temu Kembali Berprioritas untuk Meningkatkan Performa Configurable-Process Di ERP,” *J. Ilm. Mikrotek*, vol. 1, no. 2, pp. 39–46, 2014.
- [3] Chengfei Liu, Xuemin Lin, Xiaofang Zhou, and M. Orłowska, “Building a repository for workflow systems,” in *Proceedings Technology of Object-Oriented Languages and Systems (Cat. No. PR00393)*, 2003, pp. 348–357, doi: 10.1109/tools.1999.796505.

- [4] Z. Yan, R. Dijkman, and P. Grefen, "Business process model repositories - Framework and survey," *Inf. Softw. Technol.*, vol. 54, no. 4, pp. 380–395, 2012, doi: 10.1016/j.infsof.2011.11.005.
- [5] R. Astrininditya, M. A. Yaqin, and S. Zaman, "Workflow Repository Menggunakan RDF," in *Konferensi Nasional Sistem Informasi 2018*, 2018, pp. 304–309.
- [6] M. Ardani, M. A. Yaqin, and Suhartono, "Implementasi Graph Database untuk Menentukan Rute Perjalanan Transportasi Umum," in *Seminar Nasional Inovasi dan Aplikasi Teknologi di Industri*, 2019, pp. 36–42.
- [7] G. U. Abriani and M. A. Yaqin, "Analisis Implementasi Metode Semantic Similarity untuk Pengukuran Kemiripan Makna Antar Kalimat," *Ilk. J. Comput. Sci. Appl. Informatics*, vol. 1, no. 2, pp. 47–57, Dec. 2019, doi: 10.28926/ilkomnika.v1i2.15.
- [8] I. Robinson, J. Webber, and E. Eifrem, "Graph Databases," in *O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472*, 2nd ed., M. Beaugureau, Ed. Sebastopol: O'Reilly Media, Inc, 2014, pp. 27–46.
- [9] I. N. P. W. Dharmawan and R. Sarno, "Book recommendation using Neo4j graph database in BibTeX book metadata," in *Proceeding - 2017 3rd International Conference on Science in Information Technology: Theory and Application of IT for Education, Industry and Society in Big Data Era, ICSITech 2017*, Oct. 2017, pp. 47–52, doi: 10.1109/ICSITech.2017.8257084.
- [10] A. Awad, "BPMN-Q: A language to query business processes," in *Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures - Concepts and Applications, EMISA 2007*, 2007, pp. 115–128, [Online]. Available: <https://dl.gi.de/handle/20.500.12116/22179>.
- [11] J. Tague-Sutcliffe and J. Blustein, "A Statistical Analysis of the TREC-3 Data," in *Proceedings of the Third Text Retrieval Conference (TREC-3)*, 1994, pp. 385–389.