CoSciTech
Computer Science and Information Technology

p-ISSN: 2723-567X
e-ISSN: 2723-5661

Jurnal Computer Science and Information Technology (CoSciTech)

http://ejurnal.umri.ac.id/index.php/coscitech/index

SQLite Sebagai Pengganti Lucene.Net pada Pencarian Produk Toko Online

Mukhaimy Gazali[1], Rudy Anshari[2], Imam Farisi[3]
Email: [1]mukhaimy.gazali@umbjm.ac.id, [2]rudy@umbjm.ac.id, [3]1855201110006@umbjm.ac.id

[123]Universitas Muhammadiyah Banjarmasin

Diterima: 15 Oktober 2020 | Direvisi: - | Disetujui: 30 Oktober 2020

Abstrak

Suatu lembaga usaha dapat mempromosikan produk yang dijual pada toko online miliknya. Dengan adanya promosi pada toko online konsumen dapat mengetahui produk yang dijual pada toko tersebut. Konsumen yang mencari suatu produk bisa saja mencari berdasarkan berbagai atribut barang yang tersebar pada berbagai field pada tabel basis data, bahkan bisa tersebar pada tabel yang berbeda. Semua atribut yang mengarah pada suatu barang dikumpulkan pada satu dokumen yang akan dicari menggunakan sistem Pencarian Teks Lengkap. Dipilih dua alternatif sistem Pencarian Teks Lengkap yang akan digunakan, yaitu Lucene.Net dan Sqlite Full Text Search. Sebelum benar-benar digunakan kedua alternatif sistem pencarian ini diuji terlebih dahulu. Pada ukuran penyimpanan dokumen, Lucene.Net lebih unggul sebesar 6,78 kali. Pada kecepatan penulisan dokumen pencarian Sqlite lebih unggul sebesar antara 1,875 kali sampai dengan 5,197 kali. Pada kecepatan pencarian kunci, Lucene.Net lebih unggul sebesar antara 1,169 kali sampai dengan 1,698 kali. Berdasarkan pertimbangan kecepatan dan perkembangan Lucene.Net Core yang masih dalam tahap beta, Sqlite Full Text Search layak digunakan untuk proses pencarian produk di Toko Online.

Kata kunci: *Pencarian Teks Lengkap, Lucene.Net, Sqlite Full Text Search, Toko Online, Pencarian Produk*

SQLite As a Replacement To Lucene.Net in Online Store Product Searching

Abstract

A business institution can promote their products in its online shop. With the promotion of online stores, consumers can find out which products are sold at the store. Consumers who are looking for a product can search based on various attributes of goods that are scattered in various fields in the database table, and can even be spread across different tables. All attributes that point to an item are collected in one document which will be searched using the Full Text Search system. Two alternatives to the Full Text Search system were selected; Lucene.Net and Sqlite Full Text Search. Before actually being used, these two search system alternatives were tested first. In document storage size, Lucene.Net is superior by 6.78 times. The speed of writing Sqlite search documents is superior by between 1,875 times to 5,197 times. In terms of key search speed, Lucene.Net was superior by between 1,169 and 1,698 times. Based on the consideration of the speed and development of Lucene.Net Core which is still in beta stage, Sqlite Full Text Search is suitable for use in the product search process in the Online Store..

Keywords: *Full Text Search, Lucene.Net, Sqlite Full Text Search, Online Store, Product Search*

1. PENDAHULUAN

Suatu lembaga usaha / toko dapat mempromosikan usaha atau barang yang dijual melalui internet menggunakan sistem *ecommerce* dalam bentuk toko online[1, 2]. Dengan adanya toko online, konsumen dapat mengetahui informasi toko dengan lebih detail, baik dari segi jenis barang yang dijual, jasa yang ditawarkan, alamat usaha, maupun hal lainnya. Kemudahan penyampaian informasi produk melalui toko online diharapkan dapat meningkatkan penjualan[3].

Konsumen toko online yang ingin mencari suatu produk bisa saja mencari berdasarkan nama barang, kategori barang, harga maupun hal lainnya. Kata kunci yang dicari bisa jadi tersimpan di berbagai *field* pada tabel basis data bahkan bisa terdapat pada

tabel lain yang berhubungan dengan tabel produk seperti tabel kategori produk [1, 2]. Dengan tersebar-nya data yang mungkin sesuai dengan keperluan konsumen, maka *field* tabel yang mengandung informasi yang akan mengarahkan pengguna kepada produk yang dicari perlu dicatat dalam dokumen tersendiri. Sistem pencarian akan dilakukan menggunakan sistem Pencarian Teks Penuh (*Full Text Search*).

Sistem Pencarian Teks Penuh, merupakan sistem untuk memeriksa semua kata dalam setiap dokumen yang disimpan dan dicocokkan dengan kata / rangkaian kata yang dicari[4]. Pada sistem Pencarian Teks Penuh ini seringkali menghasilkan nilai peringkat yang menunjukkan relevansi / kesesuaian data yang diberikan [5]. Dengan sistem seperti ini konsumen toko online dapat diarahkan ke produk yang dicari berdasarkan kesesuaian kata kunci yang diberikan konsumen.

Sistem Full-text Search ini tidak harus dibuat sendiri. Terdapat berbagai pilihan sistem full text search yang telah tersedia, baik bersifat gratis (*freeware*) maupun yang berbayar [6].

Sistem Pencarian Teks Penuh ini kami kategori-kan menjadi dua jenis berdasarkan cara instalasi di komputer server, yaitu: *basis server* dan *basis file*. Basis server yang dimaksud adalah agar sistem dapat berjalan di komputer server, harus dilakukan instalasi sistem ke dalam komputer server, karena terdapat layanan yang harus dijalankan pada komputer server. Contoh sistem Pencarian Teks Penuh dengan basis server diantaranya: Elasticsearch [7 - 10], sistem Pencarian Teks Penuh yang tersedia di sistem basis data umum; Microsoft SQL Server [11, 12], MySQL [13, 14], Postgresql [13, 15]. Sistem Pencarian Teks Penuh basis file yang dimaksud adalah sistem Pencarian Teks Penuh yang tidak memerlukan instalasi sistem ke dalam komputer sistem, karena tidak memerlukan layanan yang harus dijalankan terus-menerus, cukup dengan menyertakan file kerja sistem agar sistem dapat berjalan. Contoh sistem Pencarian Teks Penuh berbasis file yang bisa digunakan adalah Lucene [16, 17] dan Sqlite Full Text Search [18].

Toko online yang akan dibuat menggunakan ASP.NET Core, sebagai kerangka kerja (*framework*) pengembangan web menggunakan bahasa C# dan .Net Core [19]. Sebagai sistem Pencarian Teks Lengkap untuk pencarian produk akan menggunakan sistem berbasis file yang menjadi alternatif pilihan kami adalah: Lucene.Net sebagai pengembangan dari Lucene ke bahasa pemrograman C# [20, 21]; dan Sqlite Full Text Search. Sebelum benar-benar diterapkan ke dalam sistem toko online maka, sistem Pencarian Teks Lengkap Lucene.Net dan Sqlite Full Text Search ini diuji kemampuannya terlebih dahulu. Sehingga terlihat bagaimana kelayakan Sqlite Full Text Search dibandingkan dengan Lucene.net.

Mengingat Lucene.net pada .Net Core masih dalam tahap beta [22], maka sistem pencarian menggunakan Lucene.net akan diterapkan pada bahasa pemrograman C# menggunakan .Net Framework lama / tidak menggunakan .Net Core. Teks bahan pengujian diambil beberapa teks yang terdapat pada buku gratis yang ada pada Project Gutenberg [23].

2. METODE PENELITIAN

Penelitian ini dibuat menggunakan Visual Studio 2019 edisi komunitas dengan bahasa pemrograman yang digunakan adalah bahasa C#.

2.1. Project pada Visual Studio

Proses pengujian menggunakan 3 project pada Visual Studio, sebagaimana yang disebutkan pada Tabel 1.

Tabel 1.Project pada Visual Studio 2019

No	Nama	Jenis	Kegunaan
1	PengolahanBuku	Console App (.Net Framework 4.7.2)	Mengubah file teks dari buku yang diambil dari project Guttenberg menjadi potongan file teks
2	LuceneRun	Console App (.Net Framework 4.7.2)	Menguji waktu baca dan pencarian teks menggunakan Lucene.Net
3	SqliteRun	Console App (.Net Core 3.1)	Menguji waktu baca dan pencarian teks menggunakan Sqlite

Digunakan 20 buku dalam bentuk file teks. File teks yang ada ini yang dipotong menjadi file teks berdasarkan kumpulan paragraf. Apabila kumpulan paragraf tersebut telah melebihi 2000 karakter maka file akan diakhiri dan paragraf selanjutnya akan menjadi bagian file berikutnya. Potongan file ini menjadi wakil dari keseluruhan deskripsi data barang yang akan dicari pada Toko online. File yang digunakan diperoleh dari link yang disebutkan pada Tabel 2.

Tabel 2. Link File untuk Pengujian

No	Link File
1	https://www.gutenberg.org/files/98/98-0.txt
2	https://www.gutenberg.org/files/120/120-0.txt
3	https://www.gutenberg.org/files/158/158-0.txt
4	https://www.gutenberg.org/files/1400/1400-0.txt
5	https://www.gutenberg.org/files/4107/4107-0.txt
6	https://www.gutenberg.org/files/42405/42405-0.txt

7	https://www.gutenberg.org/files/58887/58887-0.txt
8	https://www.gutenberg.org/files/60226/60226-0.txt
9	http://www.gutenberg.org/cache/epub/16768/pg16768.txt
10	http://www.gutenberg.org/cache/epub/25420/pg25420.txt
11	http://www.gutenberg.org/cache/epub/27547/pg27547.txt
12	http://www.gutenberg.org/cache/epub/27801/pg27801.txt
13	http://www.gutenberg.org/cache/epub/30047/pg30047.txt
14	http://www.gutenberg.org/cache/epub/30233/pg30233.txt
15	http://www.gutenberg.org/cache/epub/33098/pg33098.txt
16	http://www.gutenberg.org/cache/epub/38187/pg38187.txt
17	http://www.gutenberg.org/cache/epub/44484/pg44484.txt
18	http://www.gutenberg.org/cache/epub/60751/pg60751.txt
19	https://www.gutenberg.org/files/974/974-0.txt
20	https://www.gutenberg.org/files/2097/2097-0.txt

Setelah pemotongan file yang ada, diperoleh sebanyak 4.768 file dengan ukuran sekitar 11,34 Megabyte. Dengan jumlah tersebut dianggap terdapat 4.768 jenis barang sebagai bahan pencarian di Toko Online.

2.2. Model Data yang Diolah

File teks yang telah dipotong pada project PengolahanBuku, akan disimpan dalam dokumen Lucene.Net maupun tabel basis data pada Sqlite. Format model data yang disimpan diperlihatkan pada Class ItemPencarian yang diperlihatkan pada kode berikut.

Class ItemPencarian

```
public class ItemPencarian
{
    public string Id { get; set; }
    public string Isi { get; set; }
}
```

Seluruh file teks yang ada akan disimpan dalam bentuk "List" sesuai *Class* di atas. Data Id pada class ItemPencarian adalah nama file beserta nomor urut potongan file, sedangkan data Isi pada class ItemPencarian adalah isi file teks.

Pada project yang menggunakan Lucene.Net yaitu LuceneRun, model data disimpan menggunakan fungsi AddFromItemPencarianList sebagai berikut.

Fungsi AddFromItemPencarianList

```
public static void AddFromItemPencarianList(string lucenePath, List<ItemPencarian> list1)
{
  Directory directory = FSDirectory.Open(lucenePath);
  Analyzer analyzer = new StandardAnalyzer(Lucene.Net.Util.Version.LUCENE_30);
  IndexWriter writer = new IndexWriter(directory, analyzer, false,
                       IndexWriter.MaxFieldLength.UNLIMITED);
  foreach (ItemPencarian item in list1)
  {
    writer.AddDocument(item.ToLuceneDocument());
  }
  writer.Optimize();
  writer.Dispose();
}

 public Lucene.Net.Documents.Document ToLuceneDocument()
 {
   Lucene.Net.Documents.Document doc = new Lucene.Net.Documents.Document();
   doc.Add(new Field("Id", Id, Field.Store.YES, Field.Index.ANALYZED_NO_NORMS));
   doc.Add(new Field("AnalyzedData", Isi, Field.Store.NO, Field.Index.ANALYZED));
   return doc;
 }
```

Proses pencarian kata kunci yang pada project LuceneRun, diperlihatkan pada fungsi Search sebagai berikut.

Fungsi Search (pada LuceneRun)

```
public static List<LuceneSearchResult> Search(string dataPath, string kataDicari,
                                              int hitLimit = 1000)
{
  List<LuceneSearchResult> hsl = new List<LuceneSearchResult>();
  Directory lucDirectory = FSDirectory.Open(new System.IO.DirectoryInfo(dataPath));
  Analyzer lucAnalyzer = new StandardAnalyzer(Lucene.Net.Util.Version.LUCENE_30);
```

```
    QueryParser lucParser = new QueryParser(Lucene.Net.Util.Version.LUCENE_30,
                                            "AnalyzedData", lucAnalyzer);
    Searcher lucSearcher = new IndexSearcher(
                           Lucene.Net.Index.IndexReader.Open(lucDirectory, true));
    Query query = lucParser.Parse(kataDicari);
    TopScoreDocCollector collector = TopScoreDocCollector.Create(hitLimit, true);
    lucSearcher.Search(query, collector);
    ScoreDoc[] hits = collector.TopDocs().ScoreDocs;

    for (int i = 0; i < hits.Length; i++)
    {
      int docId = hits[i].Doc;
      float score = hits[i].Score;
      Document doc = lucSearcher.Doc(docId);
      LuceneSearchResult r1 = LuceneSearchResult.Doc2LuceneSearchResult(doc, i + 1, score);
      hsl.Add(r1);
    }
    return hsl;
  }
```

Sedangkan pada project yang menggunakan Sqlite yaitu SqliteRun, penyimpanan isi file teks dibantu dengan pengolahan basis data menggunakan Entity Framework Core sebagai pemetaan relasi objek (Object Relational Mapping) [24]. Proses penyimpanan ItemPencarian diperlihatkan pada fungsi OAdd sebagai berikut.

Fungsi OAdd

```
  public string OAdd(ItemPencarian item)
  {
    try
    {
      dbContext.ItemPencarianSet.Add(item);
      dbContext.SaveChanges();
      return "1";
    }
    catch (Exception ex)
    {
      return "SqFT ERROR: " + ex.Message;
    }
  }
```

Sebelum data ItemPencarian dapat disimpan ke basis data Sqlite, terlebih dahulu dibuat fungsi untuk membuat tabel ItemPencarian yang diperlihatkan pada fungsi CreateTable sebagai berikut.

Fungsi CreateTable

```
  private void CreateTables()
  {
    var sb = new System.Text.StringBuilder();
    string virtCommand = $"CREATE VIRTUAL TABLE {PencarianTableName} USING fts5 (";
    sb.AppendLine(@"PRAGMA foreign_keys = off;");
    sb.AppendLine(@"BEGIN TRANSACTION;");
    sb.AppendLine(@"");
    sb.AppendLine(virtCommand);
    sb.AppendLine(@"    Id, ");
    sb.AppendLine(@"    Isi");
    sb.AppendLine(@");");
    sb.AppendLine(@"");
    sb.AppendLine(@"COMMIT TRANSACTION;");
    sb.AppendLine(@"PRAGMA foreign_keys = on;");

    using (var command = dbContext.Database.GetDbConnection().CreateCommand())
    {
      command.CommandText = sb.ToString();
      dbContext.Database.OpenConnection();
      var reader = command.ExecuteNonQuery();
    }
  }
```

Proses pencarian kata kunci yang pada project SqliteRun, diperlihatkan pada fungsi OSearch sebagai berikut.

Fungsi OSearch

```
  public List<ItemPencarian> OSearch(string key)
  {
    string selectCommand =
```

```
    string.Format($"SELECT * FROM {PencarianTableName} WHERE Isi MATCH '{key}' ORDER BY rank");
    var hsl = dbContext.ItemPencarianSet.FromSqlRaw(selectCommand).ToList();
    return hsl;
}
```

2.3. Proses Pengujian

Hal yang menjadi bahan pengujian meliputi: 1. Ukuran dokumen untuk proses pencarian; 2. Tingkat kesamaan hasil pencarian baik dalam segi jumlah maupun urutan; 3. Kecepatan penulisan dokumen pencarian; 4. Kecepatan pencarian berdasarkan kunci yang diberikan.

Kata kunci yang dicari diperoleh dari melihat sebagian isi file, yang kemudian disusun dalam *Array of String* sebagai berikut.

Kata Kunci Pencarian

```
string[] keys = new string[] { "london", "ship", "london AND ship", "london OR ship", "emma",
              "Treasure Island", "Flint", "Dyak", "Dyaks", "Borneo", "banjar",
              "Kalamantan", "Kalimantan", "\"Sherlock Holmes\"", "Sherlock OR Holmes",
              "Canadian", "Canada",  "\"Air Force\"", "Indonesia", "Malaysia",
              "Opium", "french" };
```

Terdapat 22 string sebagai kunci pencarian. Proses pengujian akan diulang sebanyak 30 kali sebagai kelayakan pengolahan data.

2.4. Perangkat Keras untuk Pengujian Kecepatan Pengolahan Data

Proses pengujian kecepatan pengolahan data melibatkan komputer dengan spesifikasi yang diperlihatkan pada Tabel 3.

Tabel 3. Komputer untuk Proses Pengujian

No	Processor	Penyimpanan	Memori	Skor Processor	Skor Keseluruhan
1	AMD FX 6100	SSD 480 GB	DDR3 10 GB	468	1190
2	AMD FX-8800P	SSD M2 240 GB	DDR3 4 GB	384	988
3	AMD A4 3350B	SSD 120 GB	DDR4 8 GB	321	633
4	Intel Core i3-2367M	SSD 120 GB	DDR3 8 GB	196	454

Skor processor dan skor keseluruhan adalah skor yang diperoleh perangkat lunak Novabench edisi personal sebagai perangkat lunak untuk tolak ukur (*benchmark*) kemampuan kinerja komputer [25]. Seluruh komputer yang digunakan, menggunakan Sistem Operasi Windows 10 Profesional.

3. HASIL DAN PEMBAHASAN

Proses pengujian dilakukan berdasarkan hal-hal yang dijelaskan pada bagian 2.3.

3.1. Ukuran Dokumen Proses Pencarian

Dari semua komputer yang digunakan, ukuran dokumen untuk proses pencarian diperlihatkan pada Tabel 4.

Tabel 4. Ukuran Dokumen untuk Pengolahan Pencarian Teks Penuh

No	Sistem	Ukuran Dokumen (Dalam Megabyte)
1	Lucene.Net	3,91
2	Sqlite Full Text Search	26,54

Dari tabel 4, terlihat bahwa Lucene.Net lebih hemat dalam menyimpan data dibandingkan dengan Sqlite. Ukuran dokumen pencarian pada Sqlite lebih besar 6,78 kali daripada ukuran dokumen Lucene.Net. Selain itu ukuran pada Sqlite meninggkat 2,34 kali dari ukuran file sumber.

Hal ini dimungkinkan karena pada Lucene.Net semua kata yang ada pada file sumber disimpan dalam bentuk kata ber-indeks, namun pada Sqlite, semua isi text pada file sumber disimpan sebagaimana adanya beserta keperluan tambahannya. Peningkatan ukuran file dokumen untuk proses pencarian pada Sqlite bisa saja tidak linear namun bersifat eksponensial. Peningkatan ini tidak bisa dilihat karena tidak ada penelitian bagaimana perilaku penyimpanan dengan ukuran file sumber yang berbeda-beda.

3.2. Tingkat Kesamaan Hasil Pencarian

Dari semua komputer yang digunakan, hasil proses pencarian terhadap kata kunci yang diberikan diperlihatkan pada Tabel 5.

Tabel 5. Jumlah Dokumen yang ditemukan pada pencarian kunci

No	Kunci Pencarian	Lucene.Net	Sqlite	Beda?
1	london	317	317	-
2	ship	694	694	-
3	london AND ship	88	88	-
4	london OR ship	923	923	-
5	emma	325	325	-
6	Treasure Island	501	36	YA
7	Flint	44	44	-
8	Dyak	94	94	-
9	Dyaks	133	133	-
10	Borneo	193	193	-
11	banjar	0	0	-
12	Kalamantan	3	3	-
13	Kalimantan	0	0	-
14	"Sherlock Holmes"	30	30	-
15	Sherlock OR Holmes	73	73	-
16	Canadian	56	56	-
17	Canada	61	61	-
18	"Air Force"	22	22	-
19	Indonesia	1	1	-
20	Malaysia	2	2	-
21	Opium	248	248	-
22	french	230	230	-

Dari pengujian dengan 22 kunci pencarian, diperoleh hanya 1 kunci pencarian dengan jumlah yang berbeda, yaitu pada kunci pencarian: “Treasure Island”. Hal ini disebabkan pada Lucene.Net kemungkinan sudah menggunakan operasi OR pada saat pencarian. Jadi kunci “Treasure Island” pada Lucene.Net dianggap pencarian menjadi “Treasure OR Island”. Sehingga semua dokumen yang memiliki kata “treasure” maupun “island” akan dimasukkan dalam hasil pencarian. Berbeda dengan Sqlite yang tidak mengubah kunci pencarian “Treasure Island” menjadi “Treasure OR Island”.

Melihat dari hasil pencarian pada kata kunci ke-1 hingga ke-4, terlihat penggunaan operator OR dan AND bekerja pada proses pencarian. Di mana: jumlah temuan pada kunci “london” + jumlah temuan pada kunci “ship” - jumlah temuan pada kunci “london AND ship” = jumlah temuan pada kunci “london OR ship”.

3.3. Kecepatan Penulisan Dokumen Pencarian

Dari 4.768 file teks sebagai sumber data, dengan ukuran sekitar 11,34 Megabyte, pada setiap komputer yang digunakan diperoleh waktu penulisan dokumen pencarian sebagaimana diperlihatkan pada Tabel 6.

Tabel 6. Lama Proses Penulisan Dokumen Pencarian (dalam satuan detik)

No	Processor	Lucene.Net	Sqlite	Tingkat Keunggulan Sqlite
1	AMD FX 6100	408,9793615	218,1204315	1,875
2	AMD FX-8800P	536,2444655	119,4014051	4,491
3	AMD A4 3350B	760,1031589	146,2399134	5,197
4	Intel Core i3-2367M	664,6884777	226,3887462	2,936

Nilai tingkat keunggulan Sqlite pada Tabel 6, diperoleh dari lama proses penulisan dokumen pencarian pada Lucene.Net dibagi dengan lama proses penulisan dokumen pencarian pada Sqlite.

Berdasarkan nilai pada Tabel 6, Sqlite bisa lebih unggul dalam proses penulisan dokumen pencarian sebesar 1,875 kali sampai dengan 5,197 kali daripada Lucene.Net. Hal ini bisa dikarenakan proses penulisan dokumen pada Sqlite lebih sederhana daripada penulisan dokumen pencarian pada Lucene.Net.

3.4. Kecepatan Pencarian Berdasarkan Kunci yang Diberikan

Proses pencarian ini berdasarkan seluruh kunci yang diberikan sebagaimana yang berikan pada bagian 2.3. Jadi proses pencarian ini tidak dihitung berdasarkan tiap kata kunci namun langsung dihitung untuk 22 kunci. Kecepatan proses pencarian diperlihatkan pada Tabel 7.

Tabel 7. Lama Proses Pencarian Kunci (dalam satuan milidetik)

No	Processor	Lucene.Net	Sqlite	Tingkat Keunggulan Lucene.net
1	AMD FX 6100	96,45703333	144,2560033	1,496
2	AMD FX-8800P	107,71511	125,9317433	1,169
3	AMD A4 3350B	142,1880267	241,36476	1,698
4	Intel Core i3-2367M	144,2663667	215,0855533	1,491

Pada Tabel 7 terlihat Lucene.Net lebih cepat mengolah pencarian daripada Sqlite. Karena Lucene.Net lebih cepat melakukan proses pencarian maka pada kolom terakhir dihitung tingkat keunggulan Lucene.Net dibandingkan dengan Sqlite. Berbeda dengan Tabel 6 yang dihitung berdasarkan keunggulan Sqlite. Pada Tabel 7 terlihat Lucene.Net bisa lebih cepat mencari kunci yang masukkan sebesar 1,169 kali sampai dengan 1,698 kali. Namun harus diingat bahwa pengolahan ini dihitung dalam satuan milidetik, bisa jadi pengguna tidak akan merasakan perbedaan waktu tunggu pengolahan.

3.5. Kelayakan Sqlite Dibandingkan dengan Lucene.Net

Berdasarkan hasil pengujian di atas, sistem Sqlite Full Text Search dapat dianggap layak sebagai pengganti Lucene.Net untuk proses pencarian produk pada toko online. Sqlite dianggap layak karena proses penulisan dokumen pencarian lebih cepat; dan proses pencarian pada dokumen tidak benar-benar membuat waktu tunggu yang besar, yaitu masih di bawah 1 detik. Waktu pencarian di bawah 1 detik ini untuk 22 kunci yang diberikan.

Mengenai penyimpanan dokumen pada Sqlite yang lebih besar, maka harus dipertimbangkan ukuran media penyimpanan yang diberikan pada saat hosting toko online.

Perkembangan Lucene.Net pada sistem .Net Core yang masih dalam tahap Beta, juga menjadi pertimbangan untuk menganggap Sqlite Full Text Search sebagai pengganti Lucene.Net.

4. KESIMPULAN

Dari hasil pengujian terlihat bahwa hasil pencarian menggunakan Lucene.Net maupun Sqlite Full Text Search, sama-sama menghasilkan jumlah temuan yang sama berdasarkan kata kunci yang diberikan. Dengan syarat kesamaan maksud. Apabila perlu menggunakan operator “OR” sama-sama harus menggunakan opertor “OR”, karena pada Sqlite operator “OR” tidak digunakan jika tidak disebutkan.

Sqlite kalah unggul daripada Lucene.Net pada ukuran penyimpanan dokumen dan proses pencarian kunci. Lucene.Net kalah unggul daripada Sqlite pada proses penyimpanan dokumen pencarian. Pada ukuran penyimpanan dokumen, Lucene.Net lebih unggul sebesar 6,78 kali. Pada kecepatan penulisan dokumen pencarian Sqlite lebih unggul sebesar antara 1,875 kali sampai dengan 5,197 kali. Pada kecepatan pencarian kunci, Lucene.Net lebih unggul sebesar antara 1,169 kali sampai dengan 1,698 kali.

Berdasarkan pertimbangan kecepatan dan perkembangan Lucene.Net pada sistem .Net Core yang masih dalam tahap beta, Sqlite Full Text Search layak digunakan untuk menggantikan Lucene.Net pada proses pencarian produk di Toko Online.

DAFTAR PUSTAKA

[1] Susilo, Muhammad, 2018. Rancang Bangun Website Toko Online Menggunakan Metode Waterfall. *InfoTekJar (Jurnal Nasional Informatika Dan Teknologi Jaringan)* 2 (6 Maret 2018): 98–105. https://doi.org/10.30743/infotekjar.v2i2.171.

[2] Handayani, Sutri. 2018. Perancangan Sistem Informasi Penjualan Berbasis E-Commerce Studi Kasus Toko Kun Jakarta.ILKOM Jurnal Ilmiah 10,(31 Agustus 2018): 182. https://doi.org/10.33096/ilkom.v10i2.310.182-189.

[3] Novarin, Fuad, and Sintya Sukarta. 2016. Sistem Informasi Penjualan Berbasis Website Pada Beholder Cloth..*Majalah Ilmiah UNIKOM 14* (12 May 2016). https://doi.org/10.34010/miu.v14i1.168.

[4] Aruleba, Kehinde, R. Aremu, Peter Oriogun, K. Agbele, and A. Agho. 2015. Evaluation Of Full Text Search Retrieval System. *Nigeria Computer Society* (ISSN: 2141-9663, Vol 26, 154-159) 26 (22 Juli 2015): 154–59.

[5] Gene Milener, et al. Limit Search Results with RANK - SQL Server. https://docs.microsoft.com/en-us/sql/relational-databases/search/limit-search-results-with-rank, diakses 9 Oktober 2020

[6] Wikipedia. Full Text Search. https://en.wikipedia.org/w/index.php?title=Full-text_search&oldid=982280955, diakses 9 Oktober 2020

[7] Vohra, Deepak. 2015. Using Elasticsearch. 175–96, https://doi.org/10.1007/978-1-4842-1434-3_7

[8] Fedorova, V., E. Efremov, and I. Kolyagina. 2019. Search And Index Data Using Elasticsearch. *Issues of Radio Electronics* (20 Maret 2019), 74–77. https://doi.org/10.21778/2218-5453-2019-3-74-77.

[9] Vavliakis, Konstantinos, George Katsikopoulos, and Andreas Symeonidis. 2019. E-Commerce Personalization with Elasticsearch. *Procedia Computer Science 151* (1 January 2019): 1128–33. https://doi.org/10.1016/j.procs.2019.04.160.

[10] Elastic. What Is Elasticsearch? https://www.elastic.co/what-is/elasticsearch., diakses 9 Oktober 2020

[11] Seo, Chiyoung, S W Lee, and Hyoung-Joo Kim. 2013. An Efficient Inverted Index Technique for XML Documents Using RDBMS. Information and Software Technology 45 (1 January 2003): 11–22. https://doi.org/10.1016/S0950-5849(02)00157-X.

[12] Gene Milener, et al. 2017. Implementing Full-Text Search, https://docs.microsoft.com/en-us/sql/relational-databases/server-management-objects-smo/tasks/implementing-full-text-search?view=sql-server-ver15 , diakses 9 Oktober 2020

[13] Chaitanya, B., D. Reddy, B. Chandra, A. Krishna, and Remya Menon. 2019. Full-Text Search Using Database Index, https://doi.org/10.1109/ICCUBEA47591.2019.9128683.

[14] "MySQL :: MySQL 8.0 Reference Manual :: 12.10 Full-Text Search Functions." https://dev.mysql.com/doc/refman/8.0/en/fulltext-search.html. diakses 9 Oktober 2020

[15] "PostgreSQL: Documentation: 13: 12.1. Introduction." https://www.postgresql.org/docs/13/textsearch-intro.html. diakses 9 Oktober 2020

[16] Gao, Rujia. 2012. Application of Full Text Search Engine Based on Lucene. *Advances in Internet of Things* 02 (1 January 2012): 0–0. https://doi.org/10.4236/ait.2012.24013.

[17] ZHANG, Na-na, Yi-song WANG, and Kun ZHU. 2017. Design and Implementation of Full Text Search Engine Based on Lucene. *DEStech Transactions on Computer Science and Engineering*, (31 July 2017). https://doi.org/10.12783/dtcse/cst2017/12483.

[18] "SQLite FTS5 Extension." https://sqlite.org/fts5.html. diakses 9 Oktober 2020

[19] Troelsen, Andrew, and Phil Japikse. 2020. Introducing ASP.NET Core, 1091–1128. https://doi.org/10.1007/978-1-4842-5756-2_29.

[20] "Welcome to the Lucene.Net Website! | Apache Lucene.NET 4.8.0.". http://lucenenet.apache.org/, diakses 9 Oktober 2020

[21] Purwanto, Devi. 2015. Sinonim Dan Word Sense Disambiguation Untuk Melengkapi Detektor Plagiat Dokumen Tugas Akhir. Jurnal Sistem Informasi 11 (27 April 2015). https://doi.org/10.21609/jsi.v11i1.412.

[22] "Download Lucene.Net 4.8.0-Beta00012 | Apache Lucene.NET 4.8.0.". https://lucenenet.apache.org/download/version-4.8.0-beta00012.html. diakses 9 Oktober 2020

[23] Project Gutenberg. "Project Gutenberg." 2020. https://www.gutenberg.org/ , diakses 9 Oktober 2020

[24] Nikolov, Dimitar, Nikolay Pavlov, and Asen Rahnev. 2019. Using Firebird Embedded With Net Core 2 And Entity Framework Core For Cloud-Based Systems.

[25] "Novabench - Free Computer Benchmark Software." https://novabench.com/ , diakses 9 Oktober 2020