TEKNIK CONNECTION POOLING UNTUK MENINGKATKAN PERFORMA APLIKASI DENGAN BASIS DATA JARAK JAUH

Tony Wijaya
Jurusan Sistem Informasi STMIK Pontianak
Jl. Merdeka No. 372 Pontianak, Kalimantan Barat
E-mail: mail.tonywijaya@gmail.com

Abstrak

Pemrograman dengan basis data jarak jauh biasanya diterapkan pada aplikasi desktop. Hal ini dikarenakan aplikasi desktop berada di sisi client dan basis data di sisi server. Apabila client dan server masih di dalam ruang lingkup Local Area Network (LAN) atau Wide Area Network (WAN), hal ini tidak menjadi masalah. Namun apabila client harus berkomunikasi dengan server yang berada di cloud, diperlukan teknik khusus di dalam pemrogramannya. Jika hanya menerapkan teknik dasar untuk koneksi basis data, maka akan menghasilkan aplikasi yang lambat dan bahkan tidak merespons sama sekali (hang). Tujuan penelitian ini adalah memaparkan teknik connection pooling pada pemrograman demi meningkatkan performa aplikasi dengan basis data yang sangat jauh. Pendekatan dalam perancangan menggunakan metode Agile dengan pendekatan Extreme Programming yang lebih mengedepankan tercapainya fitur yang akan dibangun. Pengujian dilakukan dengan metode black box untuk membandingkan performa aplikasi. Hasil penelitian ini menunjukkan peningkatan performa yang signifikan dan bahkan menghilangkan terjadinya hang secara total, khususnya untuk basis data di cloud.

***Kata kunci** : Connection pooling, basis data jarak jauh, cloud.*

Abstract

Apps that connect to remote database usually are in desktop platform. This is due to desktop apps resides on clients while the database is on a server machine. So the client and the database are not installed on the same machine On Local Area Network (LAN) and Wide Area network (WAN), these apps usually face no problem. But if we are talking about database in the cloud server, then we need special technique in our programming. Basic database connection technique will result in a slow responsive applications, or may be even hang. This research will describe connection pooling technique in programming in order to increase app performance when connecting to a remote database such as cloud. Agile Methodology with Extreme Programming is used in this research. Black box testing is used to ensure features are correctly implemented. The result has proven significant performance improvement and even totally eliminate hangs, especially with the database in the cloud.

***Keywords**: Connection pooling, remote database, cloud.*

1. PENDAHULUAN

Aplikasi desktop biasanya melakukan koneksi ke basis data dalam komputer yang sama atau dalam jaringan lokal (LAN). Namun pada era teknologi Internet sekarang ini, tidak sedikit aplikasi desktop yang melakukan koneksi jarak jauh (remote) hingga ke cloud server. Cloud server dapat berupa shared hosting, dedicated hosting, Virtual Private Server (VPS), atau dedicated server. Dengan koneksi ke basis data jarak jauh, perusahaan atau organisasi dapat

membuka aplikasi dari mana saja, tidak harus datang ke kantor atau gedung tertentu. Hal ini dapat membantu perusahaan atau orgnisasi dalam meningkatkan produktivitas kerja.

Teknologi desktop tidak dirancang khusus untuk koneksi ke sumber data yang jauh. Akan lebih baik apabila aplikasi desktop tersebut melakukan koneksi ke web service untuk menghindari delay selama melakukan koneksi ke basis data. Namun ada beberapa alasan bagi perusahaan, organisasi, atau pengembang aplikasi untuk tidak menggunakan web service, atau dengan kata lain mempertahankan aplikasi yang sudah ada yaitu:

- Aplikasi harus dirancang ulang dari awal apabila harus melakukan koneksi ke web service. Hal ini memerlukan waktu yang cukup lama dan biaya yang cukup besar.
- Perusahaan atau organisasi ingin proses migrasi server dari lokal ke cloud server dilakukan dalam waktu yang sesingkat mungkin.

Proses migrasi basis data dari lokal ke cloud server relatif mudah dilakukan. Cukup dengan melakukan backup pada basis data lokal, upload data backup ke cloud server, dan akhirnya melakukan proses restorasi basis data ke cloud server dengan menggunakan file backup tersebut. Yang menjadi masalah adalah ketika aplikasi desktop dijalankan, akan terasa waktu delay yang signifikan. Bahkan tidak jarang terjadi gagal koneksi karena faktor jarak antara aplikasi dengan server basis data. Performa aplikasi akan turun secara signifikan. Pemakai aplikasi tidak dapat menggunakan aplikasi sebagaimana mestinya sehingga perusahaan atau organisasi justru mengalami penurunan produktivitas.

Untuk melakukan satu kali koneksi dari client ke server, proses yang dilakukan cukup panjang dan memakan sumber daya [1] yaitu :

1. Client mengirimkan permintaan ke server dengan pesan seperti “client-hello”
2. Server merespon client dengan mengirimkan pesan “server-hello”
3. Apabila menggunakan protokol SSL, maka client memilih string berukuran 48 byte dengan nama R dan mengenkripsinya dengan kunci publik RSA dan menghasilkan chiper text C. Maka R akan menjadi pre-master-secret.
4. Server menerima R dan menggunakan kunci pribadi RSA untuk mendekripsi C. Apabila berhasil, maka sudah tercipta sesi antara client dan server.

Langkah ke-4 pada poin di atas cukup memakan waktu karena ada proses dekripsi. Hal ini ditambah lagi dengan jarak yang jauh antara client dan server sehingga menyebabkan delay yang signifikan pada mekanisme aplikasi desktop.

Untuk mencegah delay yang cukup lama pada koneksi basis data jarak jauh, perlu dilakukan teknik khusus pada pemrogramannya. Salah satunya adalah dengan teknik connection pooling. Teknik ini dapat mempertahankan koneksi antara aplikasi dan server basis data yang jauh (remote) sehingga tidak terjadi delay yang signifikan pada saat menggunakan aplikasi. Penerapan connection pooling tidak menyebabkan perancangan ulang pada aplikasi, melainkan hanya perubahan kecil pada coding program yang sudah ada.

Penelitian ini mengacu pada paten mengenai metode dan sisten untuk connection pooling pada basis data yang transparan dan antrean query SQL [2]. Pada paten ini dijelaskan bahwa perlu ada akselerator basis data yang dipasang secara transparan antara client dan server. Di mana akselerator ini mempertahankan koneksi ke basis data supaya dapat digunakan kembali apabila diperlukan. Apabila tidak ada perubahan jenis koneksi ke basis data, maka akselerator tidak akan membuat koneksi baru ke basis data, melainkan menggunakan kembali koneksi yang sudah ada sehingga mempercepat performa dari keseluruhan query SQL. Dan apabila permintaan query SQL yang masuk sudah melebihi kapasitas maksimum, maka akselerator akan mengantrekan query tersebut sehingga tidak memberatkan kerja basis data pada server. Pemasangan akselerator tidak memerlukan perubahan apapun pada sistem client maupun server yang sudah ada.

2. METODE PENELITIAN

Penelitian ini dilakukan dengan pendekatan kualitatif [3]. Objek penelitian adalah STMIK Pontianak. Pengumpulan data dilakukan dengan melakukan observasi terhadap:

- Kecepatan akses aplikasi sebelum dilakukan perpindahan server (masih server lokal yaitu dalam LAN).
- Kecepatan akses aplikasi setelah dilakukan perpindahan server (cloud server VPS), sebelum penerapan teknik connection pooling.
- Kecepatan akses aplikasi setelah dilakukan perpindahan server dan setelah penerapan teknik connection pooling.

Instrumen penelitian yang digunakan dalam penelitian ini sebuah aplikasi Windows Event Viewer yang dapat merekam:

- Waktu mulai koneksi ke basis data dan
- Waktu mendapat respon koneksi (hasil query SQL) dari basis data.
- Mengambil interval antara waktu mulai dan waktu respon sehingga didapatkan kecepan koneksi basis data.

Kesimpulan dibuat berdasarkan interpretasi data oleh peneliti. Bentuk penelitian adalah penelitian eksperimen dengan objek yaitu sistem berjalan STMIK Pontianak. Peneliti melakukan pengukuran kecepan koneksi basis data sebelum dan sesudah penerapan teknik connection pooling sesuai yang dijelaskan pada poin di atas dan mengambil kesimpulan.

Metode perancangan perangkat lunak yang digunakan adalah metode Agile [4]. Prinsip dasar Agile tertuang dalam *Agile Manifesto for Software Development* [5]. Selain itu, metode ini juga dapat merespon dengan cepat terhadap perubahan daripada hanya mengikuti rancangan UML yang dibuat. Hal ini relevan karena penelitian ini dilakukan dengan bentuk eksperimen.

Pendekatan perancangan perangkat lunak yang digunakan adalah Extreme Programming. Extreme Programming merupakan disiplin dari pengembangan perangkat lunak yang berdasar pada nilai-nilai kesederhanaan, komunikasi, umpan balik, dan keberanian [6]. XP melibatkan semua anggota tim dalam setiap pekerjaannya: *pair programming* (coding bersama) dan semua pekerjaan lainnya secara bersama-sama. Jadi setiap anggota tim terlibat dalam semua aktifitas dan tidak ada pembagian tugas. Semua memiliki tanggung jawab yang sama, teknik menulis program yang sama, dan lain sebagainya. Hal ini sangat cocok diterapkan dalam penelitian ini karena coding dalam penelitian ini dilakukan oleh 1 orang saja. Karena dalam XP, setiap anggota tim juga melakukan semua pekerjaan tanpa ada pembagian tugas.

Sistem Informasi Akademik STMIK Pontianak dibangun menggunakan bahasa pemrograman C# dengan menggunakan IDE Microsoft Visual Studio 2015 [7] yang menghasilkan aplikasi desktop yang dapat dieksekusi dalam lingkungan .NET Framework [8] versi 4.5.2 atau yang lebih baru. Server basis data STMIK Pontianak ada di cloud yang berupa Virtual Private Server (VPS) dan menggunakan sistem operasi Ubuntu 18.04 LTS (Long Time Support) [9]. Basis data yang digunakan adalah MariaDB 10.1 yang merupakan basis data yang bersifat open source [10]. MariaDB digunakan karena sangat mendukung dengan sistem operasi yang digunakan yaitu Ubuntu sehingga menghasilkan performa yang sangat tinggi.

3. HASIL DAN PEMBAHASAN

Koneksi program tanpa teknik connection pooling dapat dilihat pada listing program di bawah ini:

```
OdbcConnection connection = new OdbcConnection(Program.ConnectionString);

string sql = "SELECT mmk_id AS Kode, mmk_nama AS Nama, mmk_semester AS Smt, mmk_sks AS Sks " +
    "FROM master_mata_kuliah " +
    "WHERE Aktif = 1 AND mmk_jurusan_prodi = '" + jpComboBox.SelectedValue + "' " +
        "AND mmk_id LIKE '" + DAL.NormalizeStringValue(kodeMkTextBox.Text, true) + "' " +
        "AND mmk_nama LIKE '" + DAL.NormalizeStringValue(namaMkTextBox.Text, true) + "' " +
    "ORDER BY mmk_nama, mmk_semester";

OdbcDataAdapter adapter = new OdbcDataAdapter(sql, connection);
DataTable table = new DataTable();

try
{
    adapter.Fill(table);
}
catch (Exception e)
{
    Program.MessageBoxes.ShowError(e);
}

BindingSource bindingSource = new BindingSource();
bindingSource.DataSource = table;

bindingNavigator1.BindingSource = bindingSource;

dataGridView1.DataSource = bindingSource;
dataGridView1.AutoResizeColumns();
```

Gambar 1. Contoh Koneksi ke Basis Data Tanpa Connection Pooling.

Gambar 1 menunjukkan cara standar yang digunakan pada bahasa pemrograman C# untuk melakukan koneksi ke basis data, yang dalam hal ini adalah mengambil data mata kuliah. Teknik pada gambar 2 tidak akan mengalami kendala sama sekali apabila digunakan dalam lingkungan Local Area Network (LAN). Namun apabila teknik ini digunakan untuk melakukan koneksi ke basis data jarak jauh (remote), akan mengalami kendala koneksi. Masalah yang sering timbul adalah munculnya error seperti pada gambar di bawah ini:

```
Message:
ERROR [HY000] [ma-3.1.1]Can't connect to MySQL server on '45.76.158.215' (10065)
ERROR [HY000] [ma-3.1.1]Can't connect to MySQL server on '45.76.158.215' (10065)
```

Gambar 2. Error yang Keluar Ketika Gagal Koneksi ke Basis Data.

Proses pembuatan koneksi ke basis data terdiri atas beberapa tahapan yang memakan waktu. Channel fisik seperti socket atau named pipe harus diciptakan, inisialisasi handshake antara client dan server harus dibuat, connection string dari program aplikasi harus diterjemahkan, koneksi harus diotentikasi oleh server basis data, query SQL harus diverifikasi [11].

Error pada gambar 2 tidak muncul karena kesalahan coding program. Error tersebut juga tidak muncul begitu saja ketika digunakan. Beberapa kondisi yang dapat memicu error tersebut antara lain:

- Jumlah data yang dihasilkan dalam query SQL cukup besar, misalnya 500 record. Beberapa modul dan laporan tertentu pada aplikasi akademik STMIK Pontianak melakukan query SQL dengan ukuran yang cukup besar seperti. Salah satu contohnya adalah modul “Update IPK” yang melakukan 4 (empat) kali looping terhadap sekitar 1,200 (seribu dua ratus) record. Hal ini berarti terjadi 4 x 1200 = 4800 kali koneksi yang diciptakan pada saat modul ini dijalankan. Apabila program aplikasi melakukan koneksi sebanyak itu, maka koneksi Internet tidak akan dapat mengimbangi kecepatan looping program aplikasi dengan kecepatan prosedur penciptaan *handshake* antara client dan server basis data sehingga muncul error seperti pada gambar 2.
- Beberapa pemakai menggunakan program aplikasi pada saat yang bersamaan. Hal ini dikarenakan beberapa pemakai tersebut menggunakan jaringan Internet yang sama sehingga

waktu tunda (delay) yang tercipta untuk prosedur penciptaan *handshake* antara client dan server basis data menjadi semakin lambat. Karena apabila menggunakan jaringan Internet yang sama, maka server akan mendeteksi beberapa pemakai tersebut berasal dari satu IP publik yang sama pula.

- Kecepatan koneksi Internet sedang menurun. Hal ini akan semakin menambah waktu tunda (delay) penciptaan *handshake* antara client dan server basis data.

Untuk menghindari kendala koneksi seperti di atas, maka perlu dilakukan pemutakhiran coding program pada gambar 1 dengan teknik connection pooling. Dengan menggunakan connection pooling, maka proses *handshake* antara client dan server basis data hanya tercipta 1 (satu) kali saja untuk setiap PC. Apabila koneksi sudah tercipta, maka sistem operasi akan menyimpannya di dalam memori selama beberapa waktu supaya dapat digunakan kembali. Apabila program aplikasi melakukan koneksi kembali, maka sistem operasi akan memeriksa apakah tujuan server-nya sama atau berbeda. Apabila tujuan server-nya sama, maka sistem operasi akan menggunakan kembali koneksi yang sebelumnya, sehingga tidak ada waktu tunda (delay) untuk penciptaan koneksi, melainkan program aplikasi langsung dapat mengirimkan query SQL ke server basis data.

Contoh listing program yang menggunakan teknik connection pooling dapat dilihat pada gambar berikut:

```
using (OdbcConnection connection = new OdbcConnection(Program.ConnectionString))
{
    string sql = "SELECT mmk_id AS Kode, mmk_nama AS Nama, mmk_semester AS Smt, mmk_sks AS Sks " +
        "FROM master_mata_kuliah " +
        "WHERE Aktif = 1 AND mmk_jurusan_prodi = '" + jpComboBox.SelectedValue + "' " +
            "AND mmk_id LIKE '" + DAL.NormalizeStringValue(kodeMkTextBox.Text, true) + "' " +
            "AND mmk_nama LIKE '" + DAL.NormalizeStringValue(namaMkTextBox.Text, true) + "' " +
        "ORDER BY mmk_nama, mmk_semester";

    OdbcDataAdapter adapter = new OdbcDataAdapter(sql, connection);
    DataTable table = new DataTable();

    try
    {
        adapter.Fill(table);
    }
    catch (Exception e)
    {
        Program.MessageBoxes.ShowError(e);
    }

    BindingSource bindingSource = new BindingSource();
    bindingSource.DataSource = table;

    bindingNavigator1.BindingSource = bindingSource;

    dataGridView1.DataSource = bindingSource;
    dataGridView1.AutoResizeColumns();
}
```

Gambar 3. Contoh Koneksi ke Basis Data dengan Connection Pooling.

Pada .NET Framework, teknik connection pooling sudah tertanam (*built-in*) pada komponen ADO.NET. Microsoft sebagai pencipta .NET Framework melihat pada prakteknya, sebagian besar program aplikasi hanya menggunakan satu buah koneksi basis data. Hal ini berarti, dalam sebuah aplikasi terjadi banyak pembukaan dan penutupan koneksi yang sama ke basis data. Dengan connection pooling, maka program aplikasi dapat mengurangi jumlah koneksi yang dibuat ke basis data. Dalam hal ini, *pooler* mengatur kepemilikan koneksi fisik ke basis data. *Pooler* bertugas mempertahankan beberapa koneksi ke basis data supaya tetap hidup berdasarkan *connection string*. Tempat untuk menyimpan beberapa koneksi ini disebut *pool*. Apabila ada permintaan koneksi ke basis data, maka *pooler* akan memeriksa koneksi di dalam *pool*. Apabila koneksi yang diinginkan oleh client ditemukan di dalam *pool*, maka *pooler* akan langsung mengembalikan koneksi tersebut ke client, tanpa meneruskannya ke server. Dan ketika client mengirim perintah untuk menutup koneksi, maka *pooler* tidak secara fisik menutup koneksi tersebut, melainkan menyimpannya kembali ke dalam *pool*. Dan ketika koneksi telah kembali ke dalam *pool*, maka koneksi tersebut siap dipanggil kembali apabila ada client yang memintanya.

Pada gambar 3 dapat dilihat bahwa hanya terjadi perubahan minor dari coding pada gambar

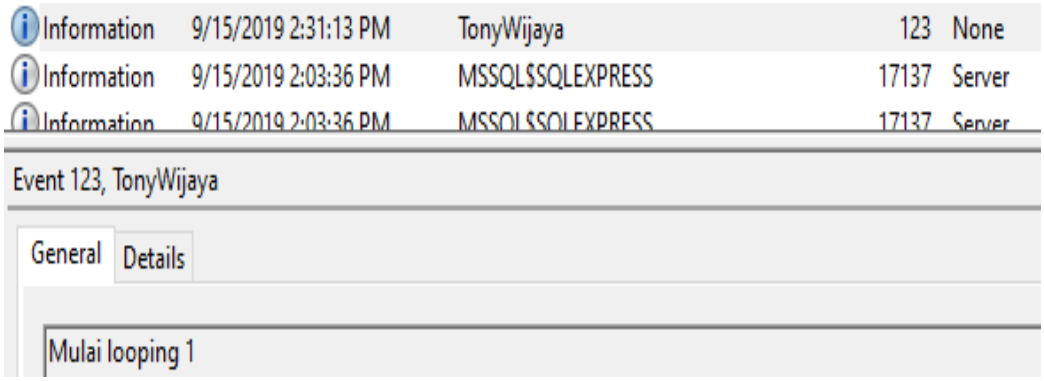
1. Akan tetapi, secara teknis dampak yang terjadi sangat signifikan terhadap performa aplikasi dekstop. Gambar 4 merupakan coding dengan memanfaatkan fitur connection pooling pada ADO.NET yaitu dengan menggunakan kata kunci *using*. Apabila kita menggunakan kata kunci tersebut, maka .NET Framework khususnya komponen ADO.NET akan memanggil *pooler* untuk memeriksa apakah koneksi ada di dalam *pool.* Apabila belum ada, maka *pooler* akan membuat koneksi baru ke server basis data MariaDB di *cloud.* Namun apabila sudah ada, maka *pooler* akan memberikan koneksi yang sudah ada di dalam *pool* sehingga perlu membuat koneksi baru. Dengan demikian tidak akan terjadi lagi error karena koneksi yang gagal seperti pada gambar 2.

```
using (OdbcConnection connection = new OdbcConnection("Dsn=databaseA; Uid=root; Pwd=12345"))
{
    connection.Open();

    // Pool A diciptakan.
}

using (OdbcConnection connection = new OdbcConnection("Dsn=databaseB; Uid=root; Pwd=abcde"))
{
    connection.Open();

    // Pool B diciptakan.
}

using (OdbcConnection connection = new OdbcConnection("Dsn=databaseA; Uid=root; Pwd=12345"))
{
    connection.Open();

    // Menggunakan koneksi dari pool A.
```

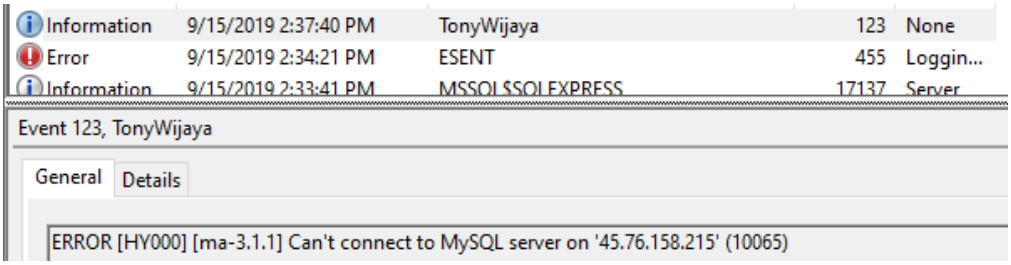
Gambar 4. Pembuatan Pool A dan B serta Penggunaan Kembali Pool A.

Connection pooling dapat meningkatkan performa dan skalabilitas aplikasi secara signifikan, terutama apabila basis data berada pada jarak yang sangat jauh. Contoh pembuatan *pool* koneksi dapat dilihat pada gambar berikut:

Dari gambar 4 dapat dilihat bahwa koneksi pertama (database A) dibuat oleh *pooler* karena belum ada koneksi di dalam *pool*. Sedangkan koneksi kedua (database B) juga dibuat oleh *pooler* karena databasenya berbeda dengan database pada *pool* A. Tapi pada koneksi ketiga, *pooler* tidak lagi membuat koneksi baru, melainkan menggunakan kembali koneksi dari *pool* A karena database-nya sama. *Pooler* pada ADO.NET menyimpan koneksi di dalam *pool* selama 4 hingga 8 menit. Apabila tidak ada koneksi yang masuk selama waktu itu, maka *pooler* akan menutup koneksi ke basis data secara fisik. *Pooler* juga akan otomatis menutup koneksi apabila mendeteksi bahwa server sudah dalam keadaan penuh atau sangat sibuk.



Gambar 5. Tanpa Connection Pooling: Memulai Looping 1 pada Pukul 14:31:13 WIB.



Gambar 6. Tanpa Connection Pooling: Error pada Pukul 14:37:40 WIB, Masih pada Looping ke-1.

Pengujian untuk performa penerapan connection pooling menggunakan aplikasi Windows Event Viewer yang sudah tertanam (*built-in*) pada sistem operasi Windows. Modul yang diuji adalah modul "Update IPK" yang melakukan 4 (empat) kali looping terhadap 1200 record mahasiswa di STMIK Pontianak. Hal ini berarti ada 4800 kali looping secara total. Program akan mencatan ke Event Viewer ketika memulai dan mengakhir setiap looping. Pada pengujian ini, dilakukan 2 (dua) kali yaitu yang pertama dilakukan sebelum implementasi connection pooling, dan yang kedua yaitu setelah implementasi connection pooling. Hasilnya dapat dilihat pada gambar 5 dan 6 di mana aplikasi gagal melakukan looping ke-1 dan menghasilkan error koneksi ke basis data.

Untuk pengujian dengan connection pooling, digunakan tabel supaya lebih ringkat. Pengujian ini membuktikan lancarnya aplikasi dalam mengeksekusi 4800 query SQL walaupun jarak basis data sangat jauh. Hal ini disebabkan koneksi hanya tercipta 1 (satu) kali saja untuk 4800 query SQL ke basis data.

Tabel 1. Hasil Pengujian dengan Teknik Connection Pooling

Looping ke-	Mulai	Selesai
1	14:42: 48 WIB	14:44:11 WIB
2	14:45:11 WIB	14:47:03 WIB
3	14:47:53 WIB	14:49:14 WIB
4	14:50:03 WIB	14:52:06 WIB

4. KESIMPULAN

Berdasarkan hasil dan pembahasan, maka dapat diambil kesimpulan bahwa implementasi teknik connection pooling pada aplikasi desktop yang memiliki basis data jarak jauh berhasil meningkatkan performa aplikasi secara signifikan. Hal ini berarti aplikasi dapat berfungsi sebagaimana mestinya seperti pada saat basis data masih berada pada jaringan Local Area Network (LAN). Selain itu, hasil implementasi juga menghilangkan error pada aplikasi akibat kegagalan koneksi yang sangat intensif, misalnya pada looping yang melibatkan transaksi pada basis data..

5. SARAN

Hasil penelitian ini menyarankan para developer aplikasi desktop untuk menggunakan teknik connection pooling walaupun pada jaringan Local Area Network (LAN) sehingga aplikasi tersebut siap di-upgrade menjadi basis data jarak jauh seperti yang dialami oleh STMIK Pontianak.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Dr. Sandy Kosasi, S.E., M.M., M.Kom. yang telah mengizinkan objek penelitian di kampus STMIK Pontianak serta membantu memberikan informasi mengenai referensi yang layak untuk penelitian ini.

DAFTAR PUSTAKA

[1] H. Sacham dan D. Boneh, "Improving SSL Handshake Performance via Batching," *Cryptographers' Track at The RSA Conference,* pp. 28-43, April 2001.

[2] V. Singh, U. V. Sawanat, P. GOEL dan N. G. Deshaveni, "Method and system for transparent database connection pooling and query queuing". United States Paten US8484242B1, 9 July 2013.

[3] W. Purhantara, Metode Penelitian Kualitatif untuk Bisnis, Yogyakarta: Graha Ilmu, 2010.

[4] A. A. Albarqi, "The Proposed L-Scrumban Methodology to Improve the Efficiency of Agile Software Development," *I.J. Information Engineering and Electronic Business,* vol. 3, p. 13, 2018.

[5] M. Beedle, A. v. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, K. Schwaber, J. Sutherland dan D. Thomas, "Signatories: The Agile Manifesto," 2001. [Online]. Available: http://agilemanifesto.org/. [Diakses 8 September 2018].

[6] L. Lindstrom dan R. Jeffries, "Extreme Programming and Agile Software Development Methodologies," *Information Systems Management,* pp. 41-52, 2004.

[7] Microsoft Developer Network (MSDN), "Visual Studio IDE," Microsoft, 23 August 2018. [Online]. Available: https://msdn.microsoft.com/en-us/library/dn762121.aspx. [Diakses 8 September 2018].

[8] Microsoft Docs, "Overview of the .NET Framework," Microsoft, 30 March 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview. [Diakses 8 September 2018].

[9] "The leading operating system for PCs, IoT devices, servers and the cloud," Ubuntu, [Online]. Available: https://ubuntu.com. [Diakses 15 09 2019].

[10] "Supporting continuity and open collaboration," MariaDB, 2019. [Online]. Available: https://mariadb.org. [Diakses 15 09 2019].

[11] Microsoft Docs, "SQL Server Connection Pooling," Microsoft, 30 3 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql-server-connection-pooling. [Diakses 15 9 2019].