MIGRASI APLIKASI MULTITENANCY PADA LAYANAN KOMPUTASI AWAN

Erick Kurniawan
Restyandito

Abstrak

Membangun infrastruktur teknologi informasi merupakan pekerjaan yang besar dan membutuhkan banyak sumber daya baik berupa biaya maupun sumber daya manusia. Biaya yang dibutuhkan meliputi biaya pengadaan perangkat keras, lisensi perangkat lunak, biaya pemantauan, pengaturan, dan perawatan infrastruktur. Solusi layanan komputasi awan hadir untuk menjawab masalah tersebut dengan menawarkan berbagai layanan yang dapat digunakan sesuai kebutuhan.

Proses migrasi dari aplikasi on-premis kedalam layanan komputasi awan menjadi satu masalah yang sering dihadapi oleh pengembang aplikasi. Banyaknya pilihan desain arsitektur multitenancy yang dapat diimplementasikan membuat pengembang menjadi bingung untuk memilih desain arsitektur dan layanan komputasi awan yang tepat.

Pada penelitian ini akan dievaluasi berbagai penerapan arsitektur multitenancy dan layanan komputasi awan dengan mempertimbangkan faktor kemudahan dalam hal migrasi aplikasi. Diharapkan hasil dari penelitian ini dapat digunakan oleh pengembang aplikasi ataupun perusahaan/instansi yang akan membangun aplikasi multitenancy berbasis layanan komputasi awan.

Kata Kunci : *Komputasi Awan, Arsitektur Multitenancy.*

1. Pendahuluan

Untuk membangun infrastruktur teknologi informasi dibutuhkan banyak biaya dari sisi biaya pengadaan perangkat keras, lisensi perangkat lunak, biaya pemantauan, pengaturan, dan perawatan infrastruktur teknologi informasi. Komputasi Awan merupakan solusi dari masalah tersebut. Teknologi komputasi awan yang ada sekarang menawarkan layanan infrastruktur teknologi informasi untuk perusahaan, karena itu perusahaan tidak perlu lagi mmbangun infrastruktur sendiri, cukup dengan menyewa layanan yang disediakan oleh perusahaan penyedia layanan seperti Amazon, Microsoft, atau Google.

Kelebihan penggunaan layanan Komputasi Awan dibandingkan dengan*on-premise* atau infrastuktur yang dibangun sendiriadalah dalam hal biaya, skalabilitas, perawatan, dan kehandalan layanan. Dengan menggunakan layanan Komputasi Awan, sumber daya yang dapat digunakan menjadi tidak terbatas. Layanan Komputasi Awan menyediakan fasilitas untuk menambah atau mengurangi sumber daya yang digunakan sesuai dengan kebutuhan tanpa harus menambahkan atau mengganti infrastruktur fisik yang digunakan.

Namun abstraksi yang disediakan oleh komputasi awan kadang masih tidak cukup untuk memenuhi kebutuhan perusahaan untuk menghemat biaya penggunaan infrastruktur teknologi informasi. Arsitektur *multitenancy* yang memungkinkan aplikasi tunggal untuk mengemulasikan lebih dari satu instan aplikasi telah direkomendasikan sebagai solusi untuk masalah tersebut. Dengan membagi sebuah aplikasi untuk melayani banyak tenant, arsitektur *multitenancy* mencoba untuk mengganti banyak instan aplikasi yang kecil dengan satu instan aplikasi yang besar yang diharapkan akan bedampak pada turunnya biaya infrastruktur teknologi informasi (Azeez et al., 2010).

Aplikasi *multitenancy* berbasis komputasi awan memiliki banyak keuntungan karena menyediakan elastisitas dan skalabilitas sumber daya komputasi dan membebaskan pengguna dari banyak pekerjaan seperti konfigurasi, pengaturan, dan perawatan sumber daya teknologi informasi (Zhang, Chen, Chen, & Zang, 2011).

Desain arsitektur *multitenancy* sangat cocok dengan lingkungan pengembangan aplikasi berbasis Komputasi Awan. Desain arsitektur *multitenancy* maka dapat dimungkinkan untuk membagi sumber daya yang ada untuk melayani banyak pengguna. Pada lingkungan pengembangan aplikasi *multitenancy* berbasis Komputasi Awan, lebih dari satu vendor dapat menggunakan infrastruktur yang sama untuk mengakses aplikasi.

Bezemer, Zaidman, Platzbeecker, Hurkmans, & Hart, (2010) berpendapat bahwa biarpun arsitektur *multitenancy* relatif baru namun arsitektur ini memiliki banyak keuntungan karena arsitektur ini menawarkan pembagian kegunaan dari sumber daya perangkat keras, meningkatkan kemudahan perawatan aplikasi, dan menurunkan keseluruhan biaya penggunaan aplikasi. Arsitektur ini cocok digunakan untuk penyedia layanan aplikasi untuk *small and medium enterprise* (SME).

Ada beberapa alternatif desain *multitenancy* yang dapat diterapkan pada layanan komputasi awan baik menggunakan layanan IaaS (Infrastructure as a Services) maupun PaaS (Platform as a Service).Untuk menentukan desain *multitenancy* yang akan dipilih maka ada beberapa faktor yang harus dipertimbangkan seperti biaya, kemudahan migrasi dari sistem yang sudah ada, dan performa pengaksesan data.

Berdasarkan uraian diatas maka evaluasi terhadap penerapan desain *multitenancy* pada layanan komputasi awan perlu dilakukan. Diharapkan hasil dari evaluasi yang dilakukan dapat digunakan sebagai acuan oleh pengembang aplikasi yang akan menerapkan desain *multitenancy* pada layanan komputasi awan.

2. Landasan Teori

2.1. Komputasi Awan

Teknolgi Komputasi Awan memungkinkan penggunaan layanan dan sumber daya yang dapat digunakan berdasarkan kebutuhan. Komputasi Awan menggunakan teknologi yang sudah ada seperti virtualisasi, *web services*, enkripsi, *utility computing*, dan Internet.

Menurut Qian, Luo, Du, & Guo, (2009) Komputasi awan adalah jenis layanan komputasi dimana layanan teknologi informasi disediakan oleh sejumlah besar unit komputer berbiaya rendah yang dihubungkan melalui jaringan. Ada 5 karakteristik dari komputasi awan yaitu: (1) sumber daya komputasi skala besar (2) skalabilitas yang tinggi dan elastisitas (3) pembagian sumber daya (secara virtual dan fisik) (4) penjadwalan sumber daya secara dinamis (5) tujuan yang umum.

Peng et al., (2010) mengkategorikan layanan komputasi awan menjadi tiga bagian yaitu IaaS (*Infrastructur as a Services*), PaaS (*Platform as a Services*), dan SaaS (*Software as a Services*). Dimana layanan SaaS memiliki arti layanan yang diberikan kepada pengguna adalah aplikasi yang berjalan pada infrastruktur komputasi awan yang disediakan oleh penyedia layanan. Layanan ini dapat diakses oleh antar muka thin client seperti browser. PaaS menunjuk kepada platform untuk memasang aplikasi yang dibuat oleh pengembang menggunakan berbagai bahasa seperti Java, Phyton, .Net, dll yang disediakan oleh layanan penyedia komputasi awan. IaaS mengacu pada layanan yang disediakan berupa infrastruktur yang (*processing power*, penyimpanan data, jaringan, dan sumber daya dasar yang lain) dimana pengembang dapat memasang *software* yang dibutuhkan untuk pengembangan aplikasi termasuk sistem operasi.

Layanan yang disediakan oleh komputasi awan dapat merepresentasikan peningkatan efisiensi dan efektivitas pada kegiatan pelaku bisnis enterprise, improvisasi pada efektivitas

biaya yang berhubungan dengan konsumsi sumber daya dan layanan. Komputasi Awan sendiri memiliki keunggulan dibandingkan dengan layanan komputasi tradisional yang sudah ada, misalnya dapat mengurangi investasi awal, performa yang lebih baik sesuai dengan kebutuhan, ketersediaan layanan yang lebih baik, skalabilitas yang tak terbatas, lebih dapat meminimalkan kesalahan, dan masih banyak lagi. Perusahaan *enterprise* seperti Amazon. Google dan Microsoft menyediakan bermacam layanan berbasis komputasi awan seperti Gmail dan Office 365, sedangkan banyak juga perusahaan atau instansi pemerintah yang lebih memilih untuk membangun komputasi awan privat pada pusat data mereka atau mengintegrasikan infrastruktur mereka dengan komputasi awan publik yang sering disebut sebagai *hybrid cloud* (Leandro, 2012).

2.2. Arsitektur *Multitenancy*

Fiaidhi, Bojanova, Zhang, & Zhang, (2012) menjelaskan bahwa *multitenacy* adalah desain arsitektur yang memungkinkan pembagian infrastruktur untuk digunakan secara bersama oleh banyak vendor/tenant, sehingga efektif dan efisien dari segi biaya, perawatan, dan skalabilitasnya. Karena arsitektur ini menggunakan sebuah instan yang di gunakan oleh banyak tenant maka isu keamanan data sangat penting, sehingga antar tenant tidak dapat melihat data dari tenant yang lain. Ada tiga macam model multitenacy yang dapat digunakan pada layanan Komputasi Awan, yang pertama adalah menggunakan basis data, yang kedua adalah menggunakan teknologi virtualisasi, dan yang ketiga adalah pemisahan secara fisik. Untuk layanan Software as a Services, desain multitenancy hampir pasti menggunakan cara yang pertama yaitu menggunakan basis data, dan menambahkan pembatasan akses data pada application layer-nya.

Untuk menggunakan basis data pada aplikasi multitenancy, ada tiga pendekatan yang dapat digunakan untuk pengaturan multitenant data pada layanan Komputasi Awan yaitu:

1. Menyimpan data dari tenant pada basis data yang terpisah.
2. Menyimpan data dari banyak tenant pada basis data yang sama, dimana setiap tenant akan memiliki tabel yang berbeda dalam basis data tersebut.
3. Menggunakan basis data yang sama dan tabel yang sama untuk menyimpan data tenant.

Menurut Krebs, Momm, & Kounev (2012), ketika mengembangkan aplikasi berbasis *multitenancy* ada beberapa tantangan dan masalah yang harus dihadapi. Tantangan pertama adalah menentukanafinitas, afinitas mendefinisikan bagaimana hubungan antar pengguna atau tenant terhubung pada sebuah titik pemrosesan server. Jadi permintaan dari pengguna atau tenant dapat dipisahkan menjadi beberapa node untuk membagi beban pemrosesan, ada tiga model afinitas yang dapat digunakan yaitu *Non-affine, Server-affine, dan cluster-affine.* Tantangan yang kedua adalah menentukan *persistence design* yaitu ada dua jenis model yang bisa digunakan yaitu setiap tenant disimpan pada basis data yang berbeda, atau penggunaan bersama tabel dimana data tenant disimpan pada sebuh tabel yang sama. Tantangan yang ketiga adalah soal kustomisasi pada aplikasi multitenant sehingga diharapkan perubahan pada satu tenant tidak berimbas pada tenant lain.

(Azeez et al., 2010) membuat aplikasi multitenancyberbasis komputasi awan yang dibangun diatas layanan Platform as a Services (PaaS) yang juga mendukung arsitektur SOA (Service Oriented Architecture) menggunakan framework WSO2 Carbon Platform. Aplikasi ini mampu untuk memilih fitur secara dinamis berdasarkan kebutuhan tenant ketika pertama kali dijalankan, namun karena aplikasi ini menggunakan Service Oriented Architecture (SOA) maka harus dipikirkan isu keamanan akses terhadap web services pada aplikasi yang dibuat.

Gao et al., (2011) juga mengembangkan sebuah aplikasi *Software as a Service* (SaaS) multitenancy yang dapat digunakan pada aplikasi dengan skala besar. Aplikasi ini menggunakan database tradisional seperti Oracle, SQL Server, atau MySQL yang terpasang pada layanan *Infrastruktur as a Services* (IaaS). Aplikasi ini menggunakan berbagai macam teknik seperti caching dan load balancing pada basis data untuk meningkatkan performa aplikasi.

3. Perancangan Sistem

3.1. Perancangan Basis Data

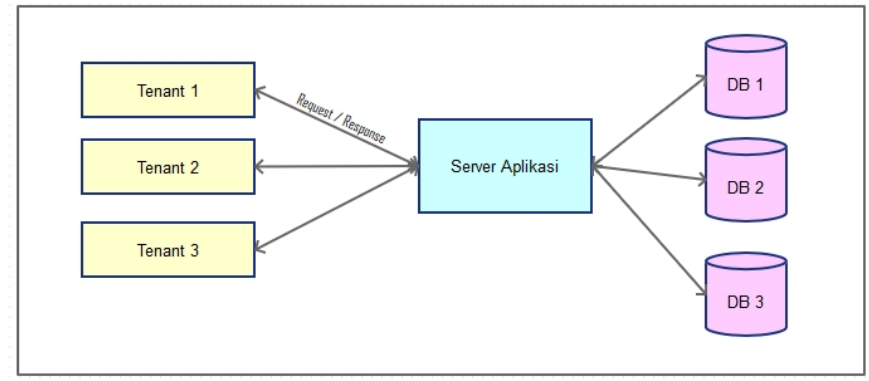
Pada penelitian ini akan digunakan beberapa tabel yang diambil dari basis data yang digunakan pada studi kasus Sistem Informasi Klinik Gigi. Tabel yang akan digunakan sebagai contoh adalah tabel yang digunakan untuk menyimpan data pasien dan pengguna sistem. Penjelasan detail tentang tabel-tabel dan bisnis proses yang digunakan pada Sistem Informasi Klinik Gigi tidak akan dibahas secara detail pada penelitian ini. Hanya tabel Pasien dan tabel yang digunakan Berikut adalah keterangan dari tabel-tabel yang akan digunakan pada percobaan.

3.1.1. Tipe Basis Data Terpisah (*Separate Databases*)

Tipe yang pertama yang akan digunakan adalah model arsitektur multitenancy yang menggunakan basis data yang berbeda. Masing-masing basis data akan diakses oleh satu instan aplikasi yang sama. Untuk menggunakan arsitektur ini makan instan aplikasi yang dibuat harus dapat menyimpan koneksi basis data dari masing-masing tenant dan dapat mengarahkan ke tenant yang tepat ketika pengguna dari tenant tertentu masuk kedalam system.

Berikut adalah bagan arsitektur aplikasi multitenant yang pertama. Tenant meminta data ke server aplikasi menggunakan url tertentu, kemudian server aplikasi akan menentukan tenant tersebut berdasarkan url yang dimasukan. Server aplikasi akan menentukan tenant mana yang mengakses dan membaca basis data terisolasi yang sudah tersedia untuk masing-masing tenant.

Isolasi basis data adalah salah satu faktor yang sering dipertimbangkan pengembang untuk menggunakan arsitektur jenis ini. Ada beberapa kebijakan dari instansi/perusahaan yang mewajibkan penyimpanan data secara terisolasi (tidak tercampur dengan data dari instansi/perusahaan lain).



Gambar 1. Arsitektur Multitenancy Basis Data Terpisah

Berikut adalah beberapa tabel yang akan digunakan sebagai contoh, yang pertama adalah data tentang pasien memiliki informasi yang dibutuhkan tentang pasien seperti nama, tanggal lahir dan, alamat. Data pasien tersimpan pada entitas pasien. Tabel 3.1 akan menjelaskan atribut yang dimiliki entitas pasien.

Tabel 1. Atribut Pada Entitas Pasien

Atribut	Konten	Tipe(Ukuran)
IdPasien	Nomor urut pasien	int
Nama	Nama pasien	varchar(50)
Gender	Gender pasien	varchar(10)
TanggalLahir	Tanggal lahir pasien	date
Alamat	Alamat pasien	varchar(50)
Kota	Kota pasien	varchar(50)
NoTelpon	Nomor telpon pasien	varchar(20)
NoHp1	Nomor hp pasien 1	varchar(20)
NoHp2	Nomor hp pasien 2	varchar(20)
Email	Email pasien	varchar(50)
Registrasi	Registrasi pasien	datetime
Ubah	Pengguna terakhir yang melakukan perubahan data	varchar(50)
Timestamp	Waktu terakhir saat data diubah	datetime

Tabel Pengguna digunakan untuk menyimpan informasi pengguna system. Tabel ini berisi informasi username dan password yang dapat digunakan oleh pengguna kedalam sistem. Pengguna pada aplikasi ini dapat memiliki jabatan yang berbeda seperti administrator, perawat, atau dokter. Masing-masing pengguna akan mempunyai hak akses yang berbeda terhadap layanan yang ada pada aplikasi.

Tabel 2. Atribut Pada Entitas Pengguna

tribut	Konten	Tipe(Ukuran)
Username	Nama pengguna staf	varchar(50)
Password	Kata sandi staf	varchar(50)
Nama	Nama staf	varchar(50)
Gender	Gender staf	varchar(50)
Alamat	Alamat staf	varchar(50)
Kota	Kota staf	varchar(50)
NoTelpon	Nomor telpon staf	varchar(20)
NoHp	Nomor hp staf	varchar(20)
Email	Email staf	varchar(50)
Jabatan	Jabatan staf	varchar(7)

Pada desain arsitektur multiteenancy basis data terpisah, dibutuhkan satu basis data berisi tabel yang digunakan untuk menyimpan konfigurasi dari masing-masing tenant. Tabel ini dapat dibuat pada basis data khusus yang digunakan untuk menyimpan semua konfigurasi yang dibutuhkan untuk mengarahkan aplikasi ke basis data tertentu yang sesuai dengan tenant tertentu.

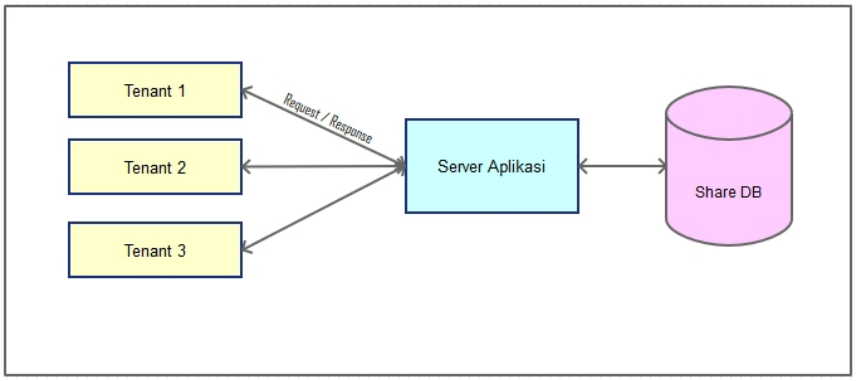
Tabel 3. Atribut Pada Entitas Pengguna

Atribut	Konten	Tipe(Ukuran)
IdTenant	Nomor tenant	int
NamaTenant	Nama tenant/instansi	varchar(50)
NamaBasisData	Nama basis data untuk setiap tenant	varchar(50)

3.1.2. Tipe Basis Data Berbagi (*Share Database Share Schema*)

Tipe arsitektur yang kedua adalah tipe basis data berbagi, yaitu menggunakan satu basis data dan tabel-tabel yang sama untuk menyimpan semua data dari tenant. Jadi pada tipe arsitektur ini semua data tenant disimpan pada satu basis data yang sama. Pendekatan ini berbeda dengan tipe yang sebelumnya dimana semua data tenant disimpan pada basis data yang berbeda sehingga data terisolasi satu dengan yang lain.

Berikut adalah gambar arsitektur multitenant dengan pendekatan basis data berbagi. Tenant mengakses server aplikasi, kemudian server aplikasi akan secara otomatis dapat menentukan tenant mana yang mengakses dan mengambil data yang ada pada tabel-tabel dalam basis data yang sudah disediakan. Karena semua data disimpan pada tabel-tabel yang sama, maka dibutuhkan sebuah identitas khusus pada setiap baris data untuk membedakan satu tenant dengan tenant yang lain.



Gambar 2. Arsitektur Multitenancy Basis Data Berbagi

Berikut adalah perancangan tabel untuk menyimpan data pasien pada jenis arsitektur basis data berbagi. Dapat dilihat pada tabel dibawah bahwa data yang disimpan memiliki identitas unik berupa IdTenant yang dapat membedakan satu tenant dengan tenant yang lain.

Berbeda dengan tipe arsitektur basis data terpisah yang sudah dibahas sebelumnya, pada tipe arsitektur basis data berbagi ini hanya menggunakan dua tabel contoh yaitu Pasien dan Pengguna. Perbedaan desain tabelnya terletak pada satu atribut yaitu IdTenant yang ditambahkan pada setiap tabel. Atribut ini akan digunakan sebagai identitas yang membedakan data yang dimiliki oleh satu tenant dengan tenant yang lain.

Pada tabel dibawah ini dapat dilihat bahwa tabel pasien memiliki atribut IdTenant yang digunakan untuk membedakan pasien dari tenant tertentu. Pada jenis arsitektur basis data berbagi, semua data pasien akan disimpan dalam satu tabel seperti pada desain tabel berikut.

Tabel 4. Penjelasan Atribut pada Entitas Pasien

Atribut	Konten	Tipe(Ukuran)
IdTenant	Identitas tenant yang disimpan di setiap baris data.	int
IdPasien	Nomor urut pasien	int
Nama	Nama pasien	varchar(50)
Gender	Gender pasien	varchar(10)
TanggalLahir	Tanggal lahir pasien	date
Alamat	Alamat pasien	varchar(50)
Kota	Kota pasien	varchar(50)
NoTelpon	Nomor telpon pasien	varchar(20)
NoHp1	Nomor hp pasien 1	varchar(20)
NoHp2	Nomor hp pasien 2	varchar(20)
Email	Email pasien	varchar(50)
Registrasi	Registrasi pasien	datetime
Ubah	Pengguna terakhir yang melakukan perubahan data	varchar(50)
Timestamp	Waktu terakhir saat data diubah	datetime

Tabel kedua yang dibuat adalah tabel Pengguna. Sama dengan tabel sebelumnya, tabel ini juga memiliki satu atribut yaitu IdTenant yang akan digunakan sebagai identitas pengguna. Dengan menggunakan atribut ini dapat diketahui pengguna yang masuk kedalam sistem berasal dari tenant yang mana.

Tabel 5. Penjelasan Tentang Tabel Pengguna

Atribut	Konten	Tipe(Ukuran)
IdTenant	Identitas tenant yang disimpan pada setiap baris data	int
Username	Nama pengguna staf	varchar(50)
Password	Kata sandi staf	varchar(50)
Nama	Nama staf	varchar(50)
Gender	Gender staf	varchar(50)
Alamat	Alamat staf	varchar(50)
Kota	Kota staf	varchar(50)
NoTelpon	Nomor telpon staf	varchar(20)
NoHp	Nomor hp staf	varchar(20)
Email	Email staf	varchar(50)
Jabatan	Jabatan staf	varchar(7)

3.2. Menggunakan Layanan Komputasi Awan

Saat ini ada dua penyedia layanan komputasi awan yang terbesar di dunia yaitu AWS (Amazon Web Services) yang merupakan produk dari Amazon, dan Microsoft Azure yang merupakan produk dari Microsoft. Masing-masing layanan memiliki keunggulan dari segi fitur dan harga yahg ditawarkan. Kedua layanan teersebut juga bersifat platform agnostic (tidak hanya mendukung satu platform saja). Walaupun demikian pada penelitian ini diputuskan untuk menggunakan layanan komputasi awan dari Microsoft yaitu Microsoft Azure dikarenakan aplikasi yang dibuat juga menggunakan framework pengembangan web dari Microsoft yaitu ASP.NET dan aplikasi basis data SQL Server. Karena berasal dari satu vendor yang sama maka tentu saja Microsoft Azure menyediakan berbagai kemudahan untuk bekerja dengan aplikasi yang menggunakan platform Microsoft.

Pada penelitian ini akan digunakan dua tipe layanan komputasi awan yang disediakan oleh Microsoft Azure yaitu:

1. Tipe layanan IaaS (*Infrastructure as a Services*) menggunakan Azure VM (*Virtual Machine*).
2. Tipe layanan PaaS (*Platform as a Services*) menggunakan Azure App Services dan SQL Azure.

4. Hasil dan Pembahasan

4.1. Analisa Proses Migrasi dari Aplikasi On-Premise

Pada tahap ini akan dibandingkan langkah-langkah yang harus dilakukan untuk melakukan proses migrasi dari sistem yang sudah ada di server *on-premise* kedalam layanan komputasi awan dengan menggunakan empat macam desain arsitektur *multitenancy* yang sudah dibahas sebelumnya.

Proses migrasi dari aplikasi yang sudah ada merupakan salah satu persoalan yang cukup kompleks. Banyak hal yang harus dipikirkan ketika akan memilih melakukan migrasi ke dalam layanan komputasi awan. Beberapa faktor yang biasanya menjadi pertimbangan adalah: faktor keamanan, biaya, skalabilitas sistem, dan kemudahan migrasi.

Pada aplikasi on-premise peneliti sudah memiliki aplikasi *backend* yang dibuat dengan menggunakan teknologi ASP.NET Web API dan aplikasi client berbasis web yang dibuat menggunakan aplikasi ASP.NET MVC. Pada sisi basis data peneliti menggunakan SQL Server 2014. Untuk desain arsitektur *multitenancy* yang digunakan pada aplikasi on-premise adalah desain arsitektur basis data terpisah. Jadi data setiap klinik akan disimpan pada basis data yang terpisah, jika ada tiga klinik, maka akan ada 3 basis data dan satu instan aplikasi.

4.1.1. Migrasi ke Layanan Azure VM dengan Arsitektur Basis Data Terpisah

Pada percobaan pertama peneliti mencoba untuk melakukan migrasi aplikasi on-premise kedalam layanan komputasi awan yang bertipe IaaS (Infrastructure as a Services) yaitu Azure VM. Arsitektur *multitenancy* yang digunakan adalah basis data terpisah. Arsitektur ini sama dengan arsitektur yang digunakan pada server on-premise. Pada percobaan ini peneliti membuat sebuat instan Azure VM yang mempunyai sistem operasi Windows Server 2012 dan SQL Server Standard. Adapun urutan langkah yang harus dilakukan adalah sebagai berikut

Adapun langkah-langkah untuk memasang aplikasi dengan desain arsitektur ini adalah sebagai berikut:

1. Masuk ke portal layanan Azure untuk membuat layanan Azure VM.
2. Pilih paket Windows Server dan SQL Server.
3. Konfigurasi basis data SQL Server yang ada pada layanan Azure VM.
4. Melakukan migrasi data untuk setiap basis data pada server on-premise ke layanan cloud.
5. Instalasi web server menggunakan Microsoft Web PI
6. Konfigurasi web server IIS agar dapat digunakan sebagai web hosting.
7. Memasang aplikasi backend Web API pada IIS.
8. Memasang aplikasi client ASP.NET MVC pada IIS.
9. Mengkonfigurasi koneksi basis data pada Web API agar terhubung ke SQL Server.
10. Konfigurasi Azure VM agar dapat diakses dari Internet.

Pada langkah diatas dapat dilihat bahwa langkah-langkah yang harus dilakukan untuk melakukan migrasi dari aplikasi on-premise kedalam layanan komputasi awan Azure VM menggunakan desain arsitektur basis data terpisah cukup mudah. Salah satu sebab adalah tidak diperlukan untuk mengganti skema basis data, cukup memindahkan basis data pada on-premise kedalam layanan Azure VM. Karena infrastruktur (sistem operasi, basis data, dan web server)

yang ada pada on-premise sama dengan yang ada pada Azure VM maka pada percobaan ini tidak ditemukan kesulitan yang berarti pada proses migrasi.

4.1.2. Migrasi ke Layanan Azure VM dengan Arsitektur Basis Data Berbagi

Pada percobaaan yang kedua peneliti melakukan migrasi dari aplikasi on-premise kedalam layanan komputasi awan yang bertipe IaaS (Infrastructure as a Services) yaitu Azure VM, percobaan ini mirip dengan percobaan sebelumnya hanya saja ada perbedaan yang cukup signifikan pada desain arsitektur yang digunakan. Pada percobaan ini peneliti menerapkan desain arsitektur basis data berbagi. Agar aplikasi yang lama yg berada pada server on-premise dapat dipindahkan maka perlu dilakukan perubahan skema pada basis data. Proses ini cukup kompleks dan memakan waktu cukup lama karena ketika peneliti merubah skema basis data tentu saja ini juga akan berpengaruh pada sisi aplikasi backend, maka aplikasi backend yang berupa Web API juga harus dirubah secara signifikan. Proses migrasi dari basis data yang lama ke basis data yang baru juga cukup kompleks, harus dibuat aplikasi khusus untuk memindahkan data pada basis data yang lama (lebih dari satu basis data) kedalam basis data yang baru (satu basis data saja). Adapun langkah-langkah untuk memasang aplikasi dengan desain arsitektur ini adalah sebagai berikut:

1. Masuk ke portal layanan Azure untuk membuat layanan Azure VM.
2. Pilih paket Windows Server dan SQL Server.
3. Membuat perancangan basis data yang sesuai dengan desain arsiteektur *multitenancy* basis data berbagi.
4. Melakukan migrasi data pada basis data dari server on-premise ke basis data berbagi yang baru.
5. Melakukan perubahan pada aplikasi backend agar dapat menyesuaikan dengan perubahan skema pada basis data.
6. Konfigurasi web server IIS agar dapat digunakan sebagai web hosting.
7. Memasang aplikasi backend Web API pada IIS.
8. Memasang aplikasi client ASP.NET MVC pada IIS.
9. Mengkonfigurasi koneksi basis data pada Web API agar terhubung ke SQL Server.
10. Konfigurasi Azure VM agar dapat diakses dari Internet.

Langkah diatas mirip seperti langkah pemasangan aplikasi pada desain yang sebelumnya, perbedaannya adalah peneliti harus membuat desain basis data baru, melakukan migrasi data, dan melakukan perubahan pada aplikasi backend Web API yang dipasang pada server IIS. Bila dibandingkan dengan proses migrasi pada arsitektur *multitenancy* basis data terpisah pada percobaan sebelumnya, maka percobaan ini jauh lebih kompleks, karena harus melakukan banyak sekali penyesuaian skema maupun kode pada basis data dan aplikasi

4.1.3. Migrasi ke Layanan Azure App dengan Arsitektur Basis Data Terpisah

Percobaan ini berbeda dengan dua percobaan sebelumnya yang menggunakan layanan bertipe IaaS (Infrastructure as a Services), percobaan ini akan menggunakan layanan komputasi awan bertipe PaaS (Platform as a Services) yaitu Azure App untuk memasang aplikasi backend Web API dan layanan SQL Azure untuk memasang basis datanya

Adapun langkah-langkah migrasi data dan aplikasi dari server on-premise yang harus dilakukan adalah sebagai berikut:

1. Buat dua layanan Azure Web Apps pada Azure Portal. Layanan ini akan digunakan untuk menaruh aplikasi backend Web API dan aplikasi client yaitu ASP.NET MVC.
2. Buat layanan SQL Azure pada Azure Portal. Jumlah instan yang dibuat dapat disesuaikan dengan jumlah tenant yang dibuat. Misal jika jumlah tenant adalah 4, maka jumlah instan basis data yang dibuat juga 4.

3. Melakukan migrasi basis data dari server on-premise kedalam layanan SQL Azure untuk setiap basis data. Karena basis data pada layanan SQL Azure sedikit berbeda dengan basis data SQL Server yang ada pada server on-premise, maka tidak semua fitur yang ada pada SQL Server dapat dimigrasikan kedalam layanan SQL Azure. Sebagai contoh SQL Azure tidak mendukung fitur "select into". Untuk melakukan migrasi peneliti harus membuat beberapa penyesuaian pada basis data.
4. Melakukan migrasi aplikasi backend Web API kedalam layanan Azure Web App.
5. Melakukan migrasi aplikasi client kedalam layanan Azure Web App.

Langkah yang dilakukan pada proses migrasi dari aplikasi on-premise ke layanan komputasi awan PaaS (Platform as a Services) cukup mudah, terutama jika pada basis data tidak terdapat fitur khusus yang tidak didukung pada layanan SQL Azure.

4.1.4. Migrasi ke Layanan Azure App dengan Arsitektur Basis Data Berbagi

Percobaan mirip dengan percobaan sebelumnya yang menggunakan layanan bertipe PaaS (Platform as a Services) yaitu Azure App untuk memasang aplikasi backend Web API dan layanan SQL Azure, hanya saja perbedaannya terletak pada desain arsitektur basis data yang akan digunakan untuk semua tenant (basis data berbagi). Karena desain basis data tidak sama dengan desain basis data yang ada pada server on-premise, maka serupa dengan percobaan yang kedua basis data harus disesuaikan skemanya agar dapat menyimpan informasi banyak tenant pada sebuah basis data saja. Kemudian diperlukan juga perubahan pada sisi aplikasi backend Web API agar dapat menggunakan desain arsitektur *multitenancy* basis data berbagi.

Adapun langkah-langkah migrasi data dan aplikasi dari server on-premise yang harus dilakukan adalah sebagai berikut:

1. Buat dua layanan Azure Web Apps pada Azure Portal. Layanan ini akan digunakan untuk menaruh aplikasi backend Web API dan aplikasi client yaitu ASP.NET MVC.
2. Buat layanan SQL Azure pada Azure Portal. Berbeda dengan percobaan yang sebelumnya, jumlah instan yang dibuat hanya satu saja untuk menyimpan semua data dari tenant.
3. Melakukan perubahan skema basis data untuk mendukung arsitektur *multitenancy* basis data berbagi.
4. Melakukan perubahan pada aplikasi backend Web API untuk menyesuaikan dengan arsitektur *multitenancy* basis data berbagi.
5. Melakukan migrasi basis data dari server on-premise kedalam layanan SQL Azure. Harus membuat program khusus untuk migrasi karena harus memindahkan data beberapa tenant yang ada pada basis data terpisah (di server on-premise) kedalam satu instan basis data pada layanan SQL Azure.
6. Melakukan migrasi aplikasi backend Web API kedalam layanan Azure Web App.
7. Melakukan migrasi aplikasi client kedalam layanan Azure Web App.

Langkah yang dilakukan pada proses migrasi dari aplikasi on-premise ke layanan komputasi awan PaaS (Platform as a Services) cukup kompleks, karena harus membuat penyesuaian pada basis data dan aplikasi backend Web API. Kelebihan aplikasi yang dibangun menggunakan desain arsitektur *multitenancy* ini akan lebih mudah untuk dikembangkankan lebih lanjut karena memiliki skalabilitas yang lebih baik.

5. Kesimpulan

Dari hasil penelitian ini didapatkan beberapa kesimpulan yaitu:

1. Layanan komputasi awan dengan tipe IaaS (Infrastructure as a Services) yaitu Azure VM adalah tipe layanan komputasi awan yang paling mudah dalam proses migrasinya,

ini dikarenakan infrastruktur yang digunakan sama dengan infrastruktur yang ada pada *server on-premise* baik sistem operasi, basis data server, maupun web server.

2. Desain arsitektur multitinancy basis data terpisah adalah tipe yang paling mudah untuk dimigrasikan karena tidak perlu melakukan modifikasi basis data dan penyesuaian aplikasi ketika dipasang pada layanan berbasis komputasi awan.
3. Desain arsitektur *multitenancy* basis data berbagi memiliki keuntungan dari sisi skalabilitas dan kemudahan pengembangan aplikasi kedepan. Karena jika diperlukan penambahan atau perubahan skema pada basis data tidak perlu menambahkan di setiap basis data seperti pada arsitektur basis data terpisah.
4. Dari sisi biaya, penggunaan desain arsitekur mutitenancy basis data terpisah pada layanan SQL Azure membutuhkan lebih banyak biaya dibandingkan dengan arsitektur basis data berbagi, karena harus membuat instan basis data untuk setiap tenant.

Dari hasil kesimpulan tersebut, penulis memiliki beberapa saran yang dapat digunakan untuk penelitian lebih lanjut, yaitu:

1. Faktor biaya harus dilihat sebagai faktor yang penting dan dibahas secara khusus karena bervariasinya paket layanan komputasi awan yang tersedia.
2. Faktor performa juga merupakan salah satu faktor yang harus dipertimbangkan untuk menentukan desain arsitektur *multitenancy* mana yang harus dipilih.

Ucapan Terima Kasih

Terima kasih kepada LPPM (Lembaga Penelitian dan Pengabdian Masyarakat) Universitas Kristen Duta Wacana yang telah mendanai penelitian ini. Penulis juga tak lupa ucapkan kepada rekan – rekan mahasiswa yang terlibat untuk membantu dalam penelitian ini.

Daftar Pustaka

Azeez, A., Perera, S., Gamage, D., Linton, R., Siriwardana, P., Leelaratne, D., … Fremantle, P. (2010). Multi-tenant SOA middleware for cloud computing. In *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*(pp. 458–465).

Bezemer, C. P., Zaidman, A., Platzbeecker, B., Hurkmans, T., & Hart, A. (2010). Enabling multi-tenancy: An industrial experience report. *IEEE International Conference on Software Maintenance, ICSM*.

Fiaidhi, J., Bojanova, I., Zhang, J., & Zhang, L. J. (2012). Enforcing *multitenancy* for cloud computing environments. *IT Professional*, *14*(1), 16–18.

Gao, B., An, W. H., Sun, X., Wang, Z. H., Fan, L., Guo, C. J., & Sun, W. (2011). A non-intrusive multi-tenant database software for large scale SaaS application. *Proceedings - 2011 8th IEEE International Conference on E-Business Engineering, ICEBE 2011*, (1), 324–328.

Krebs, R., Momm, C., & Kounev, S. (2012). Architectural Concerns in Multi-tenant SaaS Applications. *Closer*, 426–431.

Leandro, M. (2012). Multi-tenancy authorization system with federated identity for cloud-based environments using shibboleth. *ICN 2012, The …*, (c), 88–93.

Peng, J., Zhang, X., Lei, Z., Zhang, B., Zhang, W., & Li, Q. (2010). Comparison of several cloud computing platforms. *2nd International Symposium on Information Science and Engineering, ISISE 2009*, 23–27. http://doi.org/10.1109/ISISE.2009.94

Qian, L., Luo, Z., Du, Y., & Guo, L. (2009). Cloud Computing : An Overview. *Cloud Computing*, 626–631.

Zhang, F., Chen, J., Chen, H., & Zang, B. (2011). Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. *SOSP '11: Proceedings of the*

Twenty-Third ACM Symposium on Operating Systems Principles, 203–216