APLIKASI MIGRASI DATABASE DAN REPLIKASI BI-DIRECTIONAL

Michael Yoseph Ricky

Computer Science Department, School of Computer Science Binus University
Jl. K.H. Syahdan No. 9, Palmerah, Jakarta Barat 11480
mricky@binus.edu

ABSTRACT

This study aims to analyze and design a migration and replication configurations in an enterprise using several methods such as literary study and direc survey to the company; analysis on hangar systems, process migration and replication as well as existing problems; and a prototype design for migration process implementated with Oracle SQL Developer and replication process implementated with Oracle GoldenGate. The study resluts ini a prototype for migration and replication configuration processes using Oracle's Golden Gate which can produce two sets of identical data for backup and recovery. Also a simple tool is designed to assist active-active replication process as well as active-passive one. The migration process from MySQL database to Oracle database using Oracle GoldenGate can not be done because GoldenGate Oracle has bugs related to the binary log, so database migration is done using Oracle SQL Developer. However, bi-directional replication between Oracle database using Oracle GoldenGate can ensure data availability and reduce the workload of primary database.

***Keywords**: database migration, Oracle Golden Gate, bi-directional replication, database*

ABSTRAK

Penelitian ini bertujuan menganalisis dan merancang suatu konfigurasi migrasi dan replikasi pada suatu perusahaan menggunakan metode studi kepustakaan dan survei langsung di perusahaan; analisis pada sistem hangar, proses migrasi dan replikasi serta masalah yang ada; dan perancangan prototipe untuk proses migrasi dengan implementasi Oracle SQL Developer dan proses replikasi dengan implementasi Oracle GoldenGate. Hasil penelitian berupa prototipe untuk konfigurasi proses migrasi dan replikasi menggunakan Oracle Golden Gate yang dapat menghasilkan dua set data identik untuk keperluan backup dan recovery. Dirancang pula tool sederhana untuk membantu proses replikasi aktif-aktif maupun aktif-pasif. Proses migrasi dari database MySQL ke database Oracle menggunakan Oracle GoldenGate belum dapat dilakukan karena Oracle GoldenGate masih memiliki bug yang berhubungan dengan binary log sehingga dilakukan migrasi database dengan menggunakan Oracle SQL Developer. Namun, replikasi bi-directional antar database Oracle menggunakan Oracle GoldenGate dapat menjamin ketersediaan data dan mengurangi beban kerja dari primary database.

***Kata kunci**: migrasi database, Oracle Golden Gate, replikasi bi-directional, database*

PENDAHULUAN

Integrasi antara teknologi informasi dan proses bisnis akan menghasilkan informasi yang sangat berharga bagi suatu perusahaan atau organisasi dalam proses pengambilan keputusan. Maka dari itu, jaminan akan keamanan serta keandalan teknologi informasi yang ditawarkan menjadi faktor penting bagi suatu perusahaan. Akan tetapi, tidak ada satupun perusahaan penyedia teknologi informasi yang memberikan jaminan seratus persen bahwa dengan sistem yang berjalan tidak akan mengalami gangguan. Oleh karena itu perusahaan menginginkan sistem yang dapat menjaga kelangsungan proses bisnis, keutuhan, dan keamanan penyimpanan data.

Di dalam proses bisnis perusahaan maskapai penerbangan ini, belum semua sistem dari perusahaan terintegrasi ke dalam satu *database* yang sama. Sistem *hangar* pada perusahaan masih menggunakan *database* MySQL. Karena besarnya kebutuhan data dan informasi yang harus disediakan dari sistem ini, perusahaan memutuskan untuk melakukan migrasi *database* dari MySQL menjadi Oracle untuk menjalankan sistem ini. Ketersediaan data dan informasi bisnis yang dihasilkan oleh sistem *hangar* di dalam perusahaan mendapat porsi yang besar dalam keberhasilan perusahaan. Maka dari itu perlu adanya sistem *hangar* yang berjalan secara *realtime* sehingga data dan informasi dapat selalu disalurkan tepat waktu. Selain itu, perusahaan juga menginginkan suatu sistem yang menyediakan *continuous data availability* di mana sistem tersebut bebas dari gangguan yang berpotensi menimbulkan kehilangan data dan terhambatnya proses bisnis. Atas dasar kesadaran inilah, perusahaan menginginkan perancangan replikasi data pada sistem *hangar* yang tepat agar dapat menjaga kelangsungan proses bisnis, keutuhan serta keamanan informasi mereka.

Penelitian ini dibatasi pada ruang lingkup antara lain: (1) berfokus pada *database* sistem *hangar* yang akan dimigrasi dan direplikasi secara aktif-pasif (*bi-directional replication)*; (2) analisis migrasi menggunakan Oracle *GoldenGate*; (3) migrasi *database* sistem *hangar* dari *database MySQL* ke *Oracle* dengan menggunakan Oracle *SQL Developer*; (4) analisis perbandingan replikasi aktif-pasif dan aktif-aktif; (5) replikasi aktif-pasif *database* sistem *hangar* perusahaan dengan menggunakan Oracle *GoldenGate* untuk menyediakan data secara kontinu *(continuous data availability)*; (6) Objek yang akan direplikasi hanya terbatas pada tabel-tabel *database* tidak termasuk *function, stored procedure*, *view*, *trigger*, dan *security.*

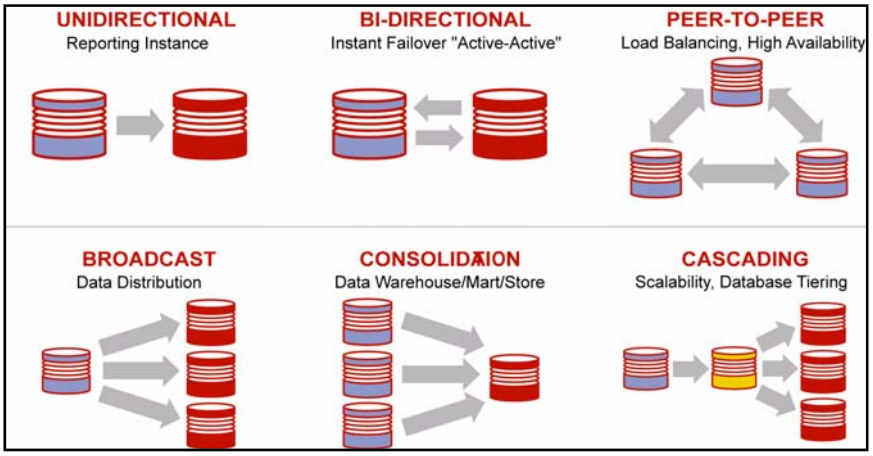
Tujuan dari topik bahasan penelitian ini adalah: (1) melakukan analisis proses migrasi dengan menggunakan Oracle *GoldenGate*; (2) melakukan migrasi *database* dari MySQL menjadi Oracle dengan menggunakan Oracle SQL Developer; (3) melakukan konfigurasi replikasi aktif-pasif data pada *database* Oracle hasil migrasi untuk menyediakan data secara kontinu *(continuous data availability)* dengan menggunakan Oracle *GoldenGate*; (4) memberikan solusi konfigurasi replikasi yang dapat menjaga ketersediaan data. Sedangkan manfaat yang diharapkan pada penelitian ini antara lain: (1) sistem *hangar* perusahaan akan memiliki *database* Oracle yang lebih tangguh daripada MySQL; (2) mencegah terjadinya kehilangan data dengan melakukan replikasi aktif-pasif antar dua *database (primary* dan *backup database*); (3) memberikan strategi konfigurasi replikasi yang dapat menguntungkan perusahaan; (4) hasil penelitian dapat digunakan untuk mengembangkan sistem-sistem yang lain untuk penerapan sistem yang *high availability.*

Landasan Teori

Oracle Data Guard

Oracle GoldenGate memungkinkan perubahan dan manipulasi data pada level transaksi di tengah peron yang lebih dari satu dan berbeda-beda (Hart dan Jesse, 2004). Oracle GoldenGate mengunakan arsitektur modular yang akhirnya memberikan fleksibilitas dalam mengekstrak dan

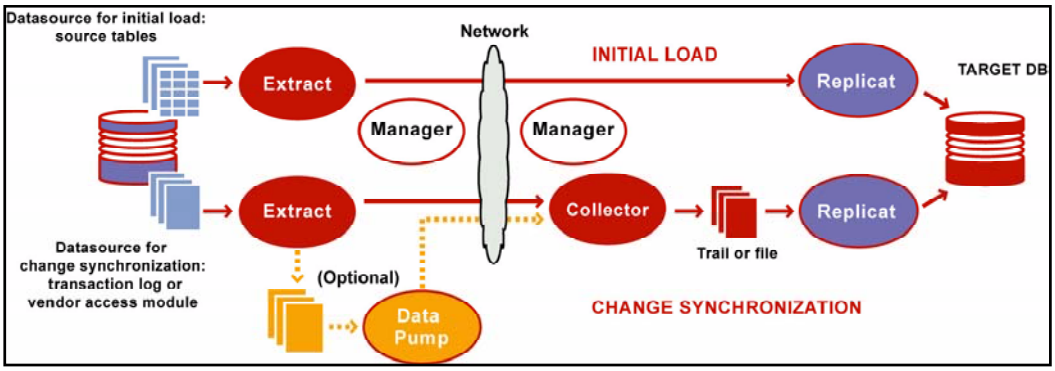
mereplikasi catatan-catatan data yang dipilih, perubahan transaksional dan perubahan-perubahan pada *data definition language* (DLL) melewati topologi-topologi yang bermacam-macam (Gambar 1, Zeis et al, 2009).



Gambar 1. Topologi-topologi yang didukung Oracle GoldenGate.

Arsitektur Oracle GoldenGate

Oracle GoldenGate terdiri atas komponen-komponen berikut (Gambar 2): (1) *extract;* (2) *data pump*; (3) *replicat;* (4) *trails* atau *extract files;* (5) *checkpoints;* (6) *manager;* (7) *collector*.



Gambar 2. Komponen-komponen *Oracle GoldenGate.*

Proses *Extract* berjalan pada sistem sumber dan merupakan mekanisme ekstraksi dari Oracle GoldenGate. Konfigurasi *Extract* dapat dilakukan dengan dua cara. Pertama, *initial loads*. Untuk *initial data loads*, *Extract* mengekstrak sekumpulan data secara langsung dari objek sumber. Kedua, mengubah sinkronisasi. Untuk menjaga data sumber disinkronisasi dengan sekumpulan data lainnya, *Extract* menangkap perubahan-perubahan yang dilakukan pada data (secara khusus transaksional *insert*, *update*, dan *delete*) setelah sinkronisasi awal telah dikerjakan. Perubahan DDL dan pengurutan juga diekstraksi jika sesuai dengan tipe *database* yang digunakan. Proses *Extract* menangkap semua perubahan yang dilakukan pada objek-objek yang dikonfigurasi untuk sinkronisasi. Proses *Extract* menyimpan semua perubahan sampai tahap telah menerima *commit records* atau *rollbacks*. Ketika *rollback* diterima, *Extract* membuang data untuk transaksi tersebut. Ketika *commit* diterima, *Extract* mengirimkan data untuk transaksi tersebut kepada proses *trail* pada sistem target. Semua catatan *log* untuk sebuah transaksi ditulis pada *trail* sebagai suatu unit transaksi yang diatur secara berurut.

Perancangan ini mempertanggungjawabkan kecepatan dan integritas data. *Data pump* adalah *Extract* tambahan yang dikonfigurasi pada sistem sumber. Jika suatu *data pump* tidak digunakan, proses *Extract* harus mengirimkan data ke dalam *remote trail* pada sistem target. Jika *data pump* dikonfigurasikan, grup *Extract* primer akan menuliskan ke dalam *local trail* dan kemudian *data pump* akan membaca *trail* dan menyalinnya ke dalam *remote trail* yang ada di sistem target. *Data pump* menambah fleksibilitas penyimpanan dan juga melayani pengisolasian proses *Extract* primer dari aktivitas TCP/IP.

Proses *replicat* berjalan pada sistem target. *Replicat* membaca perubahan data yang diekstrak dan perubahan DDL (jika ada perubahan) yang dispesifikasikan dalam konfigurasi *Replicat,* dan kemudian mereplikasikannya kembali ke *database* target. Konfigurasi *Replicat* dalam dilakukan dengan cara. Pertama dengan *initial loads*. Untuk *initial data loads, Replicat* dapat mengaplikasikan data ke objek target atau mengirimkannya ke *high-speed bulk-load utility*. Cara kedua yaitu mengubah sinkronisasi. Untuk menjaga sinkronisasi, *Replicat* mengaplikasikan perubahan data yang diekstrak pada objek target menggunakan *native database interface* atau ODBC, bergantung pada tipe *database*. DDL dan urutan-urutan yang direplikasi juga diaplikasikan jika sesuai dengan *database* yang digunakan. *Replicat* mengpalikasikan perubahan-perubahan yang direplikasi dengan perintah yang sama ketika perubahan-perubahan tersebut di-*commit* di *database* sumber.

Untuk mendukung proses ekstraksi dan replikasi yang terus-menerus dari perubahan *database*, Oracle GoldenGate menyimpan perubahan-perubahan yang ditangkap ke *disk* dalam *file* yang berseri yang disebut *trail*. Suatu *trail* dapat berada pada sistem sumber ataupun target, atau juga pada sistem *intermediate*, tergantung pada bagaimana konfigurasi Oracle GoldenGate. Pada sistem lokal, *trail* dikenal sebagai *extract trail* atau local *trail*. Pada *remote system trail* dikenal sebagai *remote trail*. Dengan menggunakan *trail* untuk penyimpanan, Oracle GoldenGate mendukung akurasi data dan toleransi kesalahan. Penggunakan *trail* juga mengizinkan aktivitas ekstraksi dan replikasi muncul secara bebas dengan *trail* yang lainnya. Dengan proses yang terpisah ini, ada juga kesempatan untuk mengatur bagaimana data dikirimkan. Contohnya, daripada mengekstrak dan mereplikasi perubahan secara terus-menerus, kita dapat mengekstrak perubahan-perubahan secara terus-menerus tetapi menyimpannya dalam *trail* untuk replikasi pada target nantinya, kapan pun aplikasi target memerlukannya.

Checkpoint menyimpan posisi yang baru dibaca dan disalin dari suatu proses ke *disk* untuk tujuan pemulihan (*recovery*). *Checkpoint* ini memastikan bahwa perubahan data yang ditandai untuk sinkronisasi diekstraksi dengan proses *Extract* dan direplikasi dengan proses *Replicat*, dan juga mencegah proses yang berulang/redundan. *Checkpoint* menyediakan toleransi kesalahan dengan mencegah kehilangan data dan mengharuskan sistem, jaringan atau suatu proses Oracle GoldenGate untuk memulai kembali. Untuk konfigurasi sinkronisasi yang kompleks, *checkpoint* memungkinkan proses *Extract* atau *Replicat* yang lebih dari satu untuk membaca kumpulan *trail* yang sama.

Checkpoint bekerja dengan *inter-process acknowledgement* untuk mencegah kehilangan data pada jaringan. Oracle GoldenGate memiliki teknologi pengirisan pesan yang terjamin. *Extract* menciptakan *checkpoint* untuk posisinya pada sumber data dan pada *trail*. *Replicat* menciptakan *checkpoint* untuk posisinya pada *trail*. Suatu sistem *checkpoint* digunakan oleh proses *Extract* dan *Replicat* yang beroperasi secara kontinu, tetapi tidak diperulkan proses *Extract* dan *Replicat* yang berjalan pada batch mode. Suatu batch process dapat dijalankan kembali dari titik mulainya, dimana proses terus-menerus memerlukan dukungan untuk interupsi yang direncanakan atau tidak direncanakan yang disediakan *checkpoint*. *Manager* adalah pengontrol proses dari Oracle GoldenGate. *Manager* harus berjalan pada kedua sistem di konfigurasi Oracle GoldenGate sebelum *Extract* atau *Replicat* dapat dimulai, dan *Manager* harus tetap berjalan sementara proses-proses tersebut berjalan sehingga fungsi manajemen sumber daya dilakukan. *Manager* melakukan fungsi-fungsi berikut: (1) *monitor* dan *restart* proses Oracle GoldenGat; (2) menerbitkan laporan-laporan permulaan, contohnya ketika *throuhgput* berjalan lambat atau ketika sinkronisasi yang terpendam meningkat; (3) menjaga

trail files dan *log*; (4) mengalokasikan ruang penyimpanan data; (5) melaporkan kesalahan dan kejadian; (6) menerima dan mengirimkan permintaan dari *user* interface

Collector adalah suatu proses yang berjalan pada layar belakang pada sistem target. *Collector* menerima perubahan *database* yang diekstrak yang dikirim melalui jaringan TCP/IP, dan menuliskannya ke *file trail* atau *extract*. Secara khusus, *Manager* memulai *Collector* secara otomatis ketika suatu koneksi jaringan diperlukan. Ketika *Manager* memulai *Collector*, proses dikenal sebagai *Collector* dinamis, dan pengguna-pengguna Oracle GoldenGate tidak berinteraksi dengannya. Walaupun demikian, *Collector* juga bisa dijalankan secara manual. Ini dikenal sebagai *Collector* statis. Tidak semua konfigurasi Oracle GoldenGate menggunakan proses *Collector*. Suatu *Collector* dinamis dapat menerima informasi hanya dari satu proses *Extract*. Untuk itu harus ada satu *Collector* dinamis di setiap proses *Extract* yang digunakan. Ketika *Collector* statis digunakan, beberapa proses *Extract* dapat berbagi satu *Collector*. Walaupun demikian, perbandingan satu banding satu adalah yang optimal. Proses *Collector* selesai ketika gabungan proses-proses *Extract* selesai. Untuk default, proses *Extract* memulai koneksi TCP/IP dari sistem sumber ke *Collector* pada target, tetapi Oracle GoldenGate dapat dikonfigurasi sehingga *Collector* memulai koneksi dari target. Pemulaian koneksi dari target mungkin diperlukan jika misalnya target ada pada area jaringan yang bisa dipercayai, tetapi sumber ada di area jaringan yang kurang bisa dipercayai.

Replikasi Data

Replikasi adalah penciptaan satu atau lebih salinan *file system* (Munoz dan Syracuse, 2000). Replikasi digunakan untuk melipatgandakan semua perubahan yang terjadi pada suatu *server* yang disebut m*aster server* atau *master*, ke *server* lain yang disebut *slave server* atau *slave* (Howard et al, 2011). Dua hal penting dari replikasi adalah menciptakan *backup* dari *server* utama untuk menghindari kehilangan data jika *master* mengalami kerusakan dan untuk memiliki salinan dari *server* utama untuk menjalankan *reporting* dan analisis kerja tanpa mengganggu jalannya bisnis. Replikasi - seperti migrasi atau sinkronisasi data – dikerjakan dalam *database* antara sumber (*source*) dan tujuan (target) (Hart dan Jesse, 2004).

Migrasi Data

Migrasi adalah pergerakan suatu "*file system*" dari satu *server* ke *server* yang lain (Morales, 2002). Migrasi *database* menunjuk pada koleksi proses-poses dan prosedur-prosedur untuk mengkonversi data dari satu *server database* ke *server database* yang lainnya (Zeis et al, 2009).
Suatu metode untuk replikasi data antara satu sumber dan satu target terdiri dari: (1) Mendefinisikan model fisikal data yang disimpan dari sumber dan target, setiap model fisikal merepresentasi struktur data yang plural; (2) Mendefinisikan model logical data dari sumber dan target, setiap model logical terdiri dari node-node yang plural dan berdasarkan pada struktur data dari model fisikal yang berhubungan (Viv, 2010).

METODE

Pengumpulan Data

Studi Pustaka

Studi pustaka dilakukan untuk memperoleh data tambahan dengan mencari dan mengumpulkan data dan informasi yang berkaitan dengan topik penelitian ini. Penelitian dilakukan dengan mengkaji teori-teori serta data-data ilmiah lain dari berbagai literatur yang akan dijadikan

pedoman penulisan penelitian ini dan sumber-sumber panduan lain seperti buku teks dan situs internet yang berkaitan dengan topik penelitian ini.

Penelitian Lapangan

Penelitian secara langsung dilakukan untuk mendapatkan data utama langsung dari perusahaan. Analisis survei dilakukan dalam bentuk wawancara langsung dengan pihak-pihak yang berkaitan guna mendapatkan informasi yang diperlukan dalam penulisan penelitian ini.

Penulis melakukan mewawancarai langsung *Database Administrator* perusahaan mengenai proses bisnis yang sedang berjalan khususnya pada sistem *hangar* serta kebutuhan-kebutuhan perusahaan.

Analisis

Analisis dilakukan terhadap sistem *hangar* berjalan pada perusahaan. Penulis juga melakukan analisis perbandingan proses migrasi dengan menggunakan Oracle *GoldenGate* dan *Oracle SQLDeveloper* serta analisis proses replikasi secara aktif-pasif *(live standby)* dan aktif-aktif *(bi-directional replication)* pada *database* Oracle hasil migrasi.

Perancangan

Metode perancangan yang digunakan dalam perancangan migrasi dan replikasi aktif-pasif ini antara lain: (1) perancangan dan implementasi migrasi menggunakan *Oracle SQLDeveloper* untuk mengubah *database* MySQL menjadi Oracle; (2) perancangan dan implementasi replikasi aktif-pasif dengan menggunakan Oracle *GoldenGate* untuk menyediakan data secara kontinu *(continuous data availability)*; (3) sedangkan untuk memudahkan proses replikasi maka dirancang *tool* sederhana dengan menggunakan bahasa pemrograman *Microsoft Visual Basic .NET 2008 Express Edition.*

HASIL DAN PEMBAHASAN

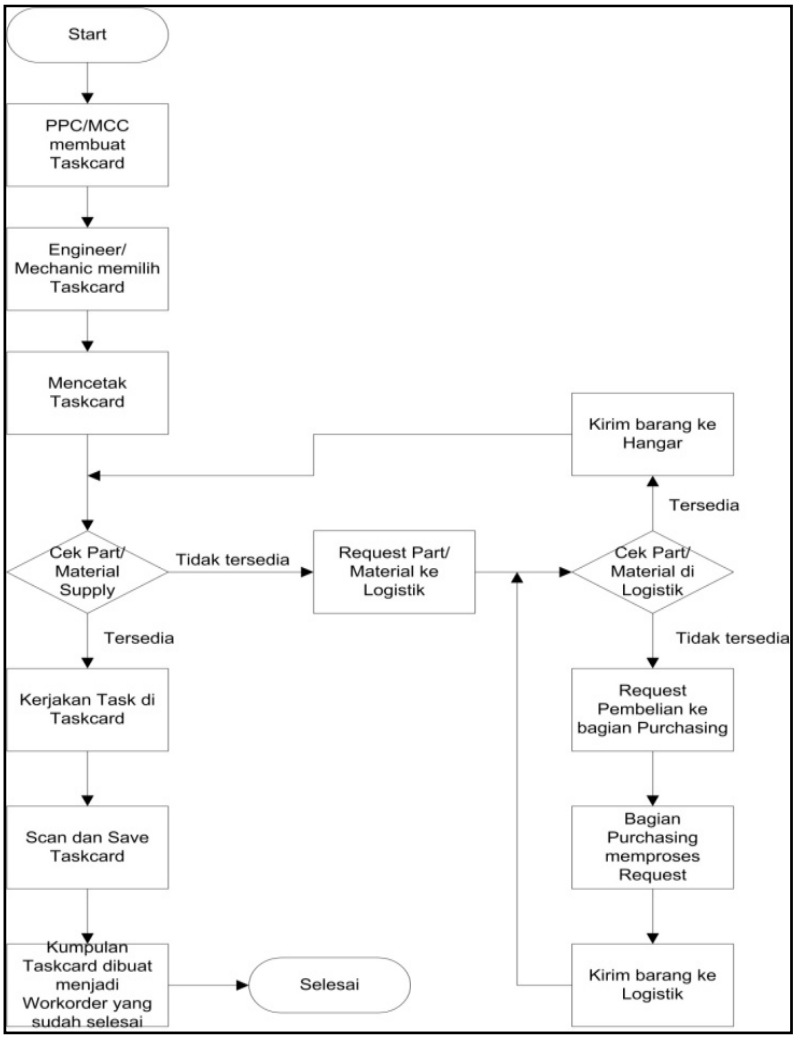
Analisis dan Perancangan

Analisis Sistem Berjalan

Di dalam sistem hangar, ada tiga bagian penting yang saling terkait yaitu *Land Maintenance (LM), Base Maintenance (BM),* dan *PPC & MMC Login.* LM dan BM bertugas untuk melakukan reparasi pesawat, pengecekan kondisi pesawat, reparasi pesawat serta perawatan berkala untuk tiap pesawat. Prosedur pada sistem hangar dimulai dengan pembuatan *Taskcard* oleh PPC/MMC. Kemudian dari *taskcard* yang sudah ada, *engineer/mechanic* akan memilih dan mencetak beberapa taskcard untuk dikerjakan. Taskcard tersebut berisi pekerjaan-pekerjaan yang harus diselesaikan teknisi. Sebelum memulai mengerjakan taskcard, teknisi harus terlebih dahulu memeriksa persediaan *part* dan *material* yang akan digunakan. *Part* dan *material* yang akan digunakan tertera pada taskcard yang telah dicetak. Jika *part* dan *material* yang akan digunakan tidak tersedia, maka teknisi harus melakukan request ke bagian logistik. Jika barang-barang yang diperlukan tersedia di bagian logistik, maka barang-barang tersebut akan langsung dikirim ke bagian hangar. Jika tidak, bagian logistik akan melakukan request pembelian barang ke bagian puchasing. Bagian puchasing akan memproses request tersebut dan *part* dan *material* yang direquest akan dikirim ke bagian logistik yang kemudian akan dikirim ke bagian hangar.

Setelah semua *part* dan *material* ada, teknisi akan segera mulai bekerja. Setiap task yang telah diselesaikan akan ditandai. Kemudian setelah semua selesai, teknisi akan melakukan *scan* dan *save taskcard* yang telah dikerjakan. *Inspector* akan mengecek apakah benar semua pekerjaan telah selesai dikerjakan. Kumpulan taskcard yang telah selesai akan dibuat ke dalam satu workorder yang berisi tugas-tugas yang telah diselesaikan, waktu mulai dan waktu akhir sebagai bagian dari perhitungan gaji teknisi.

Untuk program *hangar*, digunakan tiga server yang memenuhi kebutuhan data logistik, fungsi login, dan data *hangar*. Dan setiap server yang ada itu memiliki *database* masing-masing. Database yang digunakan yaitu MySQL. Masalah yang ditemui adalah bahwa MySQL tidak menyediakan fasilitas dblink sehingga ketika *database* ada di dua server yang berbeda, perlu membangun dua koneksi terlebih dulu. Selain itu juga MySQL tidak mendukung fasilitas *interjoin* dan *leftjoin*. Sistem yang sedang berjalan terangkum dalan bagan di bawah ini (Gambar 3).



Gambar 3. Sistem yang sedang berjalan.

Proses Migrasi

Proses migrasi *database* adalah proses untuk mengkonversi data dari satu *database* ke *database* lain baik dalam environment yang sama maupun di dalam environment yang berbeda.

Environment yang dimaksud adalah *hardware* dan *software* yang digunakan bersamaan dengan *database*. Di dalam proses ini, tidak hanya data saja yang dipindahkan, tetapi juga objek-objek yang ada di dalam *database* tersebut termasuk *table, view, trigger, stored procedure, function* dan objek-objek lainnya. beberapa alasan mengapa suatu perusahaan ingin melakukan proses migrasi pada *database* mereka, antara lain: (1) Suatu perusahaan ingin meng-upgrade *database* yang sedang mereka pakai dengan versi yang lebih baru; (2) Suatu perusahaan ingin mengganti *database* mereka dengan *database* lain yang lebih handal dan mampu melakukan proses pengolahan data yang lebih baik; (3) Hardware dan *software* yang sedang digunakan akan diganti, sehingga perusahaan dituntut untuk mengganti *database* mereka ke versi yang lebih tinggi atau ke vendor lain.

Proses Replikasi

Proses replikasi *database* adalah proses menggandakan semua perubahan yang terjadi pada suatu *database* yang disebut master server atau master ke server lain yang disebut *slave* server atau *slave* (Howard, 2010). Dengan proses ini, suatu perusahaan dapat membuat replika dari *database* yang sedang mereka pakai sebagai *backup* bila mana terjadi kegagalan pada sistem baik yang disengaja maupun yang tidak agar perusahaan tersebut masih tetap memiliki salinan dari server utama untuk menjalankan reporting dan analisis kerja tanpa mengganggu jalannya proses bisnis.

Beberapa alasan suatu perusahaan ingin melakukan konfigurasi replikasi pada *database* utamanya, antara lain: (1) Dengan melakukan konfigurasi replikasi, ketersediaan data (data availability) sehingga proses bisnis pun lebih terjamin; (2) Konfigurasi replikasi juga memungkinkan untuk mengurangi beban suatu *database* karena *database* target replikasi bisa digunakan untuk menangani reporting dan read-only queries yang pada dasarnya tidak berat, tetapi karena banyak dan sering dilakukan sehingga membebani *database* utama; (3) Perusahaan akan memiliki *database backup* yang dapat berguna bilamana terjadi kegagalan yang disengaja maupun yang tidak (*planned*/*unplanned outage*).

Analisis Masalah

Masalah yang kini sedang dihadapi oleh berkaitan dengan proses migrasi dan replikasi khususnya untuk sistem hanggar, antara lain: (1) Ukuran penampungan data pada *database* MySQL sudah tidak cukup untuk menampung banyaknya data yang harus disimpan; (2) Proses pengolahan data menjadi lambat; (3) Tidak mendukung fitur DB-Link; (4) Diperlukan *tool* yang tangguh, hemat biaya dan dapat melakukan migrasi pada berbagai platform; (5) Pengaksesan pada *database* secara berlebihan *(overload transaction)* menyebabkan pengolahan data menjadi lambat; (6) Sistem hanggar belum mempunya *database backup*.

Hasil Analisis

Ditemukan bahwa ada masalah antara MySQL atau Oracle GoldenGate. Ada dua kemungkinan yang dapat penulis berikan: (1) Database MySQL memang memiliki masalah dengan *binary-log*, di tengah keadaan masih dalam tahap *development*; (2) Oracle GoldenGate versi 11.0.0 baru diluncurkan pada bulan Oktober 2010 yang lalu, sehingga ada kemungkinan juga bahwa Oracle GoldenGate masih dalam tahap pengembangan. Karena *software* yang tergolong baru, ada kemungkinan masih ada *bug* yang belum terselesaikan.

PENUTUP

Simpulan

Analisis dan perancangan migrasi dan replikasi data pada sistem *hangar* dimaksudkan untuk meningkatkan performa *database* sistem *hangar* akibat dari banyaknya jumlah transaksi yang ada serta untuk membangun suatu konfigurasi aktif-aktif yang berguna ketika terjadi *outage* pada *database* tersebut.

Setelah melakukan analisis dan perancangan migrasi dan replikasi data ini, dapat ditarik dua simpulan. Pertama, setelah melakukan percobaan penggunaan Oracle GoldenGate untuk migrasi *database* dari MySQL ke Oracle, penulis mendapati bahwa masih ada masalah yang terjadi pada saat migrasi, yaitu masalah pada binary log di *database* MySQL. Terjadi kesalahan dari Oracle GoldenGate ketika pengaksesan status dari *binary log* yang sudah dalam keadaan *enabled* sebagai *disabled*. Kedua, replikasi aktif-aktif (bi-directional replication) dilakukan untuk membangun suatu konfigurasi antar dua *database* agar keduanya saling tersinkronisasi dan menghasilkan dua *database* yang memiliki set data yang identik. Replikasi ini dapat menjamin ketersediaan data pada sistem hanggar serta mengurangi beban kerja pada primary *database*. Selain itu set data hasil replikasi (secondary *database*) dapat digunakan untuk menggantikan *primary database* apabila terjadi *outage* pada *database* tersebut.

Saran

Berikut beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut. Pertama, perusahaan disarankan untuk mendokumentasikan table-tabel pada setiap *database* yang ada serta alur proses bisnis di dalam perusahaan itu sendiri agar dapat memudahkan *developer* untuk merancang konfigurasi migrasi dan replikasi di masa yang akan datang. Hal ini didasarkan pada kesulitan penulis mendapatkan struktur *database* sistem hanggar yang akan dimigrasi dan direplikasi. Kedua, perusahaan disarankan untuk melakukan konfigurasi peng-*install*-an *database* Oracle dan Oracle *GoldenGate* mengikuti *best practice* untuk menghindari kesalahan-kesalahan yang mungkin terjadi. Ketiga, konfigurasi dengan menggunakan Oracle *GoldenGate* sangatlah luas sehingga penulisan penelitian ini dapat digunakan sebagai dokumentasi awal untuk mengembangkan konfigurasi Oracle *GoldenGate* lainnya seperti *database zero downtime migration, disaster recovery* dan *data reporting.* Meskipun begitu, tidak disarankan untuk melakukan migrasi *database* dari MySQL dengan menggunakan Oracle *GoldenGate* karena *tool* ini masih sangat baru dan masih ada *bug* yang belum diperbaiki. Begitu pula dengan MySQL-nya sendiri yang masih dalam proses *development.*

DAFTAR PUSTAKA

Hart, M., Jesse S. (2004). *Oracle Database 10g: High Availablity with RAC Flashback & Data Guard*. California: McGraw-Hill Osborne.

Howard, G., Irving, S. M., Sceales, A. M., Sauvage, A. M. F. (2010). *Error prevention for data replication*. Diakses dari http://www.ipo.gov.uk/p-find-publication-getPDF.pdf?PatentNo=GB2468742&DocType=A&JournalNumber=6331.

Morales, Tony. (2002). *Oracle9i Database Migration, Release 2 (9.2)*. Diakses dari http://download.oracle.com/docs/cd/B10501_01/server.920/a96530.pdf

Munoz, J., Syracuse, N. (2000). *Replication and Migration*. Diakses dari https://www.ietf.org/proceedings/49/slides/NFSV4-4.pdf.

Viv, S. (2010). *Oracle® Database High Availability Overview 11g Release 1 (11.1)*. diakses dari http://isu.ifmo.ru/docs/doc112/server.112/e10804.pdf.

Zeis, C., Ruel, C., Wessler, M. (2009). *Oracle® 11g For Dummies*. Indianapolis: Wiley Publishing.