APLIKASI MIGRASI *DATABASE* DAN REPLIKASI *BI-DIRECTIONAL*

Michael Yoseph Ricky
Binus University
mricky@binus.edu

ABSTRACT

The objective of this research is to analyze and design a migration and replication configuration in an enterprise. The research methodologies used are data collection, analysis, and design. Data collection is conducted based on literature study and visit to the company. The analysis is performed based on the analysis of the hangar system, migration process analysis, replication, and the occurring problems. The design is conducted by designing a prototype for migration process implementing the Oracle SQL Developer and replication process implementing the Oracle Golden Gate. The result of this research is a prototype for configuring the migration process and replication using Oracle Golden Gate which results in two identical datasets for backup and recovery. A simple tool is also designed for helping the active-active and active-passive replication process. The result of this research shows that the migration process from the MySQL database to Oracle database using the Oracle Golden Gate still cannot be performed, since the Oracle Golden Gate still has a bug related to the binary log. The migration is then performed using the Oracle SQL Developer. However, the bi-directional replication between the Oracle database using the Oracle Golden Gate can assure the availability of data and reduction of workload in the primary database.

Keywords: *Database Migration, Oracle Golden Gate, Replication, Bi-Directional, Database.*

1 Pendahuluan

1.1 Latar Belakang

Pada era globalisasi seperti sekarang ini, dunia bisnis berkembang sangat pesat dan menjadi semakin kompetitif. Seiring dengan berkembangnya dunia bisnis, teknologi informasi pun dituntut untuk berkembang dan berinovasi agar dapat mengimbangi kemajuan dunia bisnis. Perusahaan dan organisasi akan semakin bergantung pada teknologi informasi yang paling efektif dan efisien di dalam proses menjalankan bisnis mereka termasuk di dalamnya proses penyimpanan dan pengolahan informasi bisnis.

Integrasi antara teknologi informasi dan proses bisnis akan menghasilkan informasi yang sangat berharga bagi suatu perusahaan atau organisasi dalam proses pengambilan keputusan yang menyangkut kepentingan-kepentingan bisnis perusahaan tersebut. Maka dari itu jaminan akan keamanan serta keandalan suatu teknologi informasi menjadi faktor penting bagi suatu perusahaan untuk menggunakan teknologi informasi yang ditawarkan. Akan tetapi, tidak ada satupun perusahaan penyedia teknologi informasi yang memberikan jaminan seratus persen bahwa dengan menggunakan teknologi informasi mereka, sistem yang berjalan tidak akan mengalami gangguan, baik yang terencana maupun yang tidak terencana seperti kesulitan dalam mengintegrasikan data-data yang menggunakan *database* yang berbeda, kemungkinan terjadi *downtime* ketika proses migrasi data, kinerja yang tinggi dari *server* karena pengaksesan yang terus menerus sehingga memungkinkan terjadinya kerusakan pada *server*, ataupun adanya kemungkinan terjadi bencana-bencana alam yang tidak terduga. Gangguan-gangguan tersebut mungkin saja berdampak buruk pada proses penyimpanan dan pengolahan informasi bisnis yang mengakibatkan kerugian pada suatu perusahaan. Oleh karena itu perusahaan menginginkan sistem yang dapat menjaga kelangsungan proses bisnis, keutuhan, dan keamanan penyimpanan data.

Di dalam proses bisnisnya, belum semua sistem dari perusahaan terintegrasi ke dalam satu *database* yang sama. Sistem *hangar* pada perusahaan masih menggunakan *database* MySQL. Namun karena besarnya kebutuhan data dan informasi yang harus disediakan dari sistem ini, maka perusahaan memutuskan untuk melakukan migrasi *database* dari MySQL menjadi *Oracle* untuk menjalankan sistem ini.

Dalam perkembangan bisnis perusahaan, ketersediaan data dan informasi bisnis yang dihasilkan oleh sistem *hangar* di dalam perusahaan mendapat porsi yang besar dalam keberhasilan perusahaan sebagai salah satu maskapai penerbangan yang paling diminati oleh masyarakat. Maka dari itu perlu adanya sistem *hangar* yang berjalan secara *realtime* sehingga data dan informasi yang diperlukan dapat selalu disalurkan tepat waktu. Selain itu, perusahaan juga menginginkan suatu sistem yang menyediakan *continuous data availability* dimana 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.

1.2 Ruang Lingkup

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 Golden Gate*
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 Golden Gate* 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.*

1.3 Tujuan dan Manfaat

Tujuan dari topik bahasan penelitian ini adalah:
1. Melakukan analisis proses migrasi dengan menggunakan *Oracle Golden Gate.*
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 Golden Gate.*
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.*

1.4 Metodologi

1.4.1 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 wawancara langsung dengan *Database Administrator* perusahaan mengenai proses bisnis yang sedang berjalan khususnya pada sistem *hangar* serta kebutuhan-kebutuhan perusahaan.

1.4.2 Metode Analisis

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

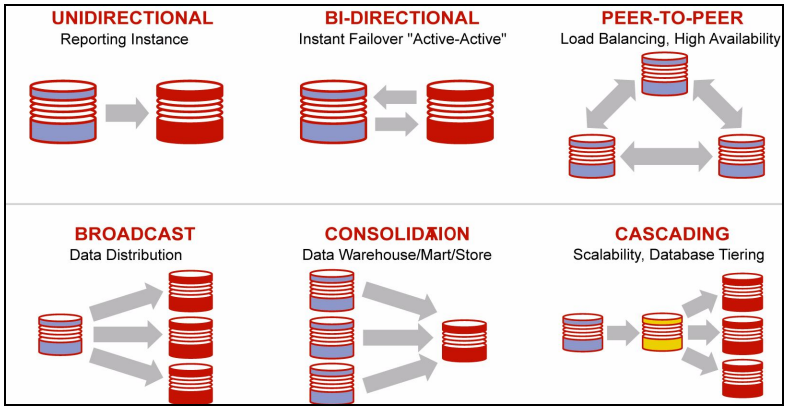
1.4.3 Metode Perancangan

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

2 Landasan Teori

2.1 *Oracle Data Guard*

Oracle Golden Gate memungkinkan perubahan dan manipulasi data pada level transaksi di tengah peron yang lebih dari satu dan berbeda-beda[2]. *Oracle Golden Gate* mengunakan arsitektur modular yang akhirnya memberikan fleksibilitas dalam mengekstrak dan mereplikasi catatan-catatan data yang dipilih, perubahan transaksional dan perubahan-perubahan pada DLL (*data definition language*) melewati topologi-topologi yang bermacam-macam[7].

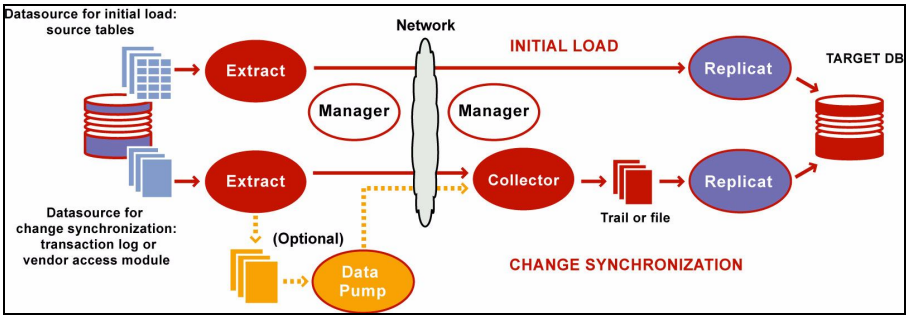


Gambar 1. Topologi-Topologi Yang Didukung *Oracle Golden Gate*

2.2 Arsitektur *Oracle Golden Gate*

Oracle Golden Gate terdiri atas komponen-komponen berikut:

- *Extract*
- *Data pump*
- *Replicat*
- *Trails atau extract files*
- *Checkpoints*
- *Manager*
- *Collector*



Gambar 2. Komponen-Komponen *Oracle Golden Gate*

2.3 *Extract*

Proses *Extract* berjalan pada sistem sumber dan merupakan mekanisme ekstraksi dari *Oracle Golden Gate*. Konfigurasi *Extract* dapat dilakukan dengan cara:

- *Initial loads*: Untuk *initial data loads*, *Extract* mengekstrak sekumpulan data secara langsung dari objek sumber.
- 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.

2.4 *Data Pump*

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.

2.5 *Replicat*

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:

- *Initial loads*: Untuk *initial data loads*, *Replicat* dapat mengaplikasikan data ke objek target atau mengirimkannya ke *high-speed bulk-load utility*.
- 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* mengaplikasikan perubahan-perubahan yang direplikasi dengan perintah yang sama ketika perubahan-perubahan tersebut di-*commit* di *database* sumber.

2.6 *Trails*

Untuk mendukung proses ekstraksi dan replikasi yang terus-menerus dari perubahan *database*, *Oracle Golden Gate* 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 Golden Gate*. 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 Golden Gate* mendukung akurasi data dan toleransi kesalahan. Penggunaan *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, kapanpun aplikasi target memerlukannya.

2.7 *Checkpoints*

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, mengharuskan sistem, jaringan atau suatu proses *Oracle Golden Gate* 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 Golden Gate* 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 diperlukan 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*.

2.8 *Manager*

Manager adalah pengontrol proses dari *Oracle Golden Gate*. *Manager* harus berjalan pada kedua sistem dikonfigurasi *Oracle Golden Gate* 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:

- *Monitor* dan *restart* proses *Oracle Golden Gate.*
- Menerbitkan laporan-laporan permulaan, contohnya ketika *throuhgput* berjalan lambat atau ketika sinkronisasi yang terpendam meningkat.
- Menjaga *file-file trail* dan *log.*
- Mengalokasikan ruang penyimpanan data.
- Melaporkan kesalahan dan kejadian.
- Menerima dan mengirimkan permintaan dari *user interface.*

2.9 *Collector*

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 Golden Gate* tidak berinteraksi dengannya. Walaupun demikian, *Collector* juga bisa dijalankan secara manual. Ini dikenal sebagai *Collector* statis. Tidak semua konfigurasi *Oracle Golden Gate* menggunakan proses *Collector*. Ketika suatu *Collector* dinamis digunakan, *Collector* dapat menerima informasi hanya dari satu proses *Extract*, maka 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 Golden Gate* 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.

2.10 Replikasi Data

Replikasi adalah penciptaan satu atau lebih salinan "*file system*"[5]. Replikasi digunakan untuk melipatgandakan semua perubahan yang terjadi pada suatu *server* yang disebut *master server* atau *master*, ke *server* lain yang disebut *slave server* atau *slave*[3]. 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*)[2].

2.11 Migrasi Data

Migrasi adalah pergerakan suatu "*file system*" dari satu *server* ke *server* yang lain[4]. Migrasi *database* menunjuk pada koleksi proses-proses dan prosedur-prosedur untuk mengkonversi data dari satu *server database* ke *server database* yang lainnya[7]. Suatu metode untuk replikasi data antara 1 sumber dan 1 target[6], terdiri dari:

- Mendefinisikan model fisikal data yang disimpan dari sumber dan target, setiap model fisikal merepresentasi struktur data yang plural.
- 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.

3 Analisis dan Perancangan

3.1 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 *purchasing*. Bagian *purchasing* akan memproses *request* tersebut dan *part* dan *material* yang di*request* 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 3 *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, karena MySQL tidak menyediakan fasilitas *db-link*, maka ketika *database* ada di 2 *server* yang berbeda, perlu membangun 2 koneksi terlebih dulu, selain itu juga MySQL tidak mendukung fasilitas *interjoin* dan *leftjoin*.

Gambar 3. Sistem Yang Sedang Berjalan

3.2 Proses Migrasi

Proses migrasi *database* adalah proses untuk mengkonversi data dari satu *database* ke *database* yang 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. Ada 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.

3.3 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*[3]. 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 tak disengaja (*planned*/*unplanned outage*).

3.4 Analisis Masalah

Masalah yang kini sedang dihadapi berkaitan dengan proses migrasi dan replikasi khususnya untuk sistem *hangar*, antara lain:

- Ukuran penampungan data pada *database* MySQL sudah tidak cukup untuk menampung banyaknya data yang harus disimpan.
- Proses pengolahan data menjadi lambat.

- Tidak mendukung fitur *DB-Link*.
- Diperlukan *tool* yang tangguh, hemat biaya, dan dapat melakukan migrasi pada berbagai *platform*.
- Pengaksesan pada *database* secara berlebihan *(overload transaction)* menyebabkan pengolahan data menjadi lambat.
- Sistem *hangar* belum mempunyai *database backup*.

3.5 Hasil Analisis

Ditemukan bahwa ada masalah antara MySQL atau *Oracle Golden Gate*. Ada 2 kemungkinan yang dapat penulis berikan:

1. *Database* MySQL memang memiliki masalah dengan *binary-log*, di tengah keadaan masih dalam tahap *development*.
2. *Oracle Golden Gate* versi 11.0.0 baru diluncurkan pada bulan Oktober 2010 yang lalu, sehingga ada kemungkinan juga bahwa *Oracle Golden Gate* masih dalam tahap pengembangan. Karena *software* yang terlalu baru maka ada kemungkinan masih ada *bug* yang belum terselesaikan.

4 Kesimpulan dan Saran

4.1 Kesimpulan

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, maka dapat ditarik kesimpulan sebagai berikut:

1. Setelah melakukan percobaan penggunaan *Oracle Golden Gate* 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 Golden Gate* ketika pengaksesan status dari *binary log* yang sudah dalam keadaan *enabled* sebagai *disabled*.
2. Replikasi aktif-aktif (*bi-directional replication*) dilakukan untuk membangun suatu konfigurasi antar dua *database* agar kedua *database* tersebut saling tersinkronisasi dan menghasilkan dua *database* yang memiliki set data yang identik. Replikasi ini dapat menjamin ketersediaan data pada sistem *hangar* 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.

4.2 Saran

Beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut, antara lain:

1. 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 *hangar* yang akan dimigrasi dan direplikasi.
2. Perusahaan disarankan untuk melakukan konfigurasi peng-*install*-an *database Oracle* dan *Oracle Golden Gate* mengikuti *best practice* untuk menghindari kesalahan-kesalahan yang mungkin terjadi.
3. Konfigurasi dengan menggunakan *Oracle Golden Gate* sangatlah luas sehingga penulisan penelitian ini dapat digunakan sebagai dokumentasi awal untuk mengembangkan konfigurasi *Oracle Golden Gate* lainnya seperti *database zero downtime migration, disaster recovery* dan *data reporting.* Meskipun begitu, tidak disarankan untuk melakukan migrasi *database* dari MySQL dengan menggunakan *Oracle Golden Gate* 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

[1] Connolly, T. Dan Begg, C. (2005). *Database System, 4th Edition*. California, Addison Wesley Publishing Company, Inc.
[2] Hart, M., Jesse S. (2004). *Oracle Database 10g: High Availablity with RAC Flashback & Data Guard*. Osborne, McGraw-Hill Companies, Inc.
[3] Howard, G., Irving, S.M., Sceales, A.M., Sauvage, A.M.F. (2010). *Error prevention for data replication.* http://www.ipo.gov.uk/p-find-publication-getPDF.pdf?PatentNo=GB2468742&DocType=A&JournalNumber=6331
[4] Morales, T. (2002). *Oracle9i Database Migration, Release 2 (9.2) Tony Morales 2002*. United States of America, *Oracle* Corporation
[5] Munoz, J., Syracuse, N. (2000). *Replication and Migration.* https://www.ietf.org/proceedings/49/slides/NFSV4-4.pdf
[6] Viv, S. (2010). *Oracle® Database High Availability Overview 11g Release 1 (11.1)*. United States of America, *Oracle* Corporation
[7] Zeis, C., Ruel, C., Wessler, M. (2009). *Oracle® 11g For Dummies*. Indianapolis, Wiley Publishing, Inc.