Modifikasi Metode *Timestamped Change Data Capture* pada RDBMS *(Relational Database Management System)*

Hendra Suprayogi[1], Harry Soekotjo Dachlan[2], Muhammad Aswin[3]

***Abstract*—A normalization process must be applied in the data structure in database designing process in RDBMS, in order to eliminate data anomaly and redundancy. This procedure increases access performance as well as reduces unnecessary storage space. It transforms a single entity into multiple entities, normalization process is often so-called decomposition. A problem occurs, however, after a normalized database is being implemented. When a new data recorded there is nothing wrong with its retrieval, while the retrieval process goes into the last data updated in the case of updating activity. Moreover, when a deletion occurred, the data can never be retrieved. Timestamped CDC method offers a way of tracking data change. Basically, timestamped CDC uses historical record to include timestamp everytime change occurs. Tracking updated record is simply done by the previous research of timestamped CDC, while keeping track on deleted data is impossible since the data vanished within the deletion process. This paper modifies the timestamped CDC method so that data deletion tracking can be achieved as well as data updating. Furthermore, this research also makes several testing procedures to find out performance of the modified timestamped CDC structures.**

***Intisari*—Proses normalisasi harus dilakukan pada struktur data dalam perancangan basis data pada RDBMS. Proses normalisasi ini akan membersihkan basis data dari anomali dan redundansi, yang pada akhirnya bisa meningkatkan performa dan mengurangi kebutuhan ruang penyimpanan yang tidak diperlukan. Pada proses normalisasi, terjadi pemisahan satu entitas menjadi banyak entitas yang disebut dengan dekomposisi. Tetapi dengan adanya normalisasi, terjadi masalah pada proses pengubahan dan penghapusan data, yang mana proses tersebut tidak bisa dilacak kembali karena tertumpuk data akhir ataupun terhapus. *Timestamped* CDC merupakan sebuah cara untuk melacak hasil pengubahan data tersebut. Pada dasarnya, timestamped CDC menggunakan rekaman runtutan yang menyertakan jejak waktu ditambahkannya atau diubahnya data tersebut. Pada penelitian sebelumnya, telah berhasil dibuat struktur *timestamped* CDC yang mampu melacak hasil pengubahan data, tetapi masih belum berhasil melacak penghapusan data pada waktu dilakukan penghapusan karena rekaman yang bersangkutan tersebut ikut terhapus. Makalah ini memodifikasi metode *timestamped* CDC sehingga mampu melacak penghapusan data. Selain itu makalah ini juga menguji performa struktur modifikasi *timestamped* CDC yang dibuat.**

***Kata Kunci*— Basis Data, *Change Data Capture*, *Timestamp*, Normalisasi, Entitas Runtutan, RDBMS.**

[1]*Mahasiswa, Jurusan Magister Teknik Elektro Minat Sistem Komunikasi dan Informatika Universitas Brawijaya, Jln. MT Haryono 167 Malang 65145 INDONESIA (e-mail: hendrussuprayogus@gmail.com)*

[2,3]*Dosen, Fakultas Teknik Jurusan Teknik Elektro Universitas Brawijaya, Jln. MT Haryono 167 Malang 65145 INDONESIA (tlp:0341-551430)*

I. PENDAHULUAN

Struktur penyimpanan data pada basis data relasional (*Relational Database Management System* - RDBMS) yang ada saat ini bergantung pada relasi antar entitas, yang seringkali disebut dengan *relationship* atau *join*. RDBMS menyimpan data pada entitas-entitas yang terpisah, berbeda dengan sistem penyimpanan yang menaruh semua data pada entitas tunggal, dengna tujuan meningkatkan kecepatan akses dan fleksibilitas.

Pada basis data relasional, basis data harus dinormalisasi untuk membersihkan dari anomali dan redundansi. Pada proses normalisasi, terjadi pemisahan satu entitas menjadi banyak entitas yang disebut dengan dekomposisi. Setelah entitas dipisahkan (didekomposisi), entitas-entitas tersebut bisa digabungkan lagi dengan menggunakan operasi *join* [1].

Hal yang sering dijumpai adalah ketika perancang basis data melakukan proses normalisasi, terjadi masalah pada penyimpanan entitas utama yang ternyata tidak mampu mengatasi pengubahan komponen-komponen data yang bisa diikuti oleh entitas transaksinya sekaligus mempertahankan integrasinya dalam menghasilkan laporan yang akurat seiring terjadinya proses pengubahan atribut yang dinamis pada entitas utama tersebut. Sebagai contoh, bila pada entitas utama terdapat atribut yang bersifat bisa berubah, sedangkan pada entitas transaksi terdapat atribut yang bergantung pada kunci entitas utama, kemudian terdapat serangkaian transaksi dan di suatu saat terjadi proses pengubahan (*update*) pada atribut entitas utama, lalu serangkaian transaksi terjadi kembali. Pada kasus ini, ketika di akhir periode dilakukan proses pelaporan sesuai dengan kondisi data yang ada pada saat tersebut, maka informasi yang keluar dari atribut yang bersifat bisa berubah tersebut akan menghasilkan hanya informasi terakhir (setelah dilakukan pengubahan), baik untuk transaksi sebelum maupun sesudah atribut tadi diubah. Pada dasarnya, informasi hasil dari pelaporan akan memberikan hasil yang salah. Informasi yang benar dari atribut tersebut hanya berasal dari periode setelah dilakukan pengubahan yang terakhir kali.

Kasus yang paling sering terjadi adalah bila sebuah produk memiliki harga jual yang berfluktuasi, maka dengan menggunakan struktur basis data ternormalisasi, proses pelaporan data akan selalu mendapatkan hasil pengubahan harga yang terakhir kali. Harga sebelum-sebelumnya tidak akan tercatat pada tabel, sehingga pelaporan harga tidak lagi akurat. Selain itu, kasus yang juga bisa terjadi (meskipun tidak terlalu sering) adalah suatu produk yang tidak diproduksi lagi dan harus dihapus atau dinonaktifkan, maka dengan menggunakan struktur data ternormalisasi akan mendapatkan hasil pelaporan di mana produk tersebut seolah-olah tidak pernah ada, karena data produk tersebut sudah terhapus.

Diperlukan suatu teknik pelacakan hasil pengubahan data atau disebut dengan *change data capture* (CDC) pada kasus-kasus data berubah tersebut dan tetap mendapatkan pelaporan yang 100% akurat tanpa ada kesalahan sedikitpun. Selain itu diperlukan juga kesederhanaan struktur proses dan performa yang tinggi ketika dilakukan akses baik data *input* maupun data *output*.

Metode yang bisa digunakan untuk CDC antara lain adalah *record-based*, *reproduce*, *trigger*, DB *snapshot*, *log-based*, dan *refresh-table*. Di antara metode tersebut, hanya *trigger* dan *log-based* yang bisa diterapkan secara *real-time* [2].

Satu penelitian lain mengatakan, beberapa metode yang diklaim bisa digunakan untuk implementasi *real-time* CDC adalah *log-based*, *tri*gger, dan *timestamped* [3].

Pada *timestamped* CDC, data baru yang datang sebagai *input* diberi *timestamp* (informasi waktu berlaku). Dalam metode ini, tidak mungkin untuk melakukan pendeteksian rekaman yang dihapus, karena informasi waktu tersebut akan terhapus bersama dengan rekamannya [4].

Dalam makalah ini akan dibangun teknik *timestamped* CDC yang dimodifikasi untuk melakukan akses entitas utama dan transaksi dengan antarmuka *stored procedure* sebagai ETL (*Extract-Transform-Load*) *tools* sehingga selain mampu melakukan pendeteksian penambahan rekaman dan pengubahan rekaman, juga dapat digunakan untuk mendeteksi penghapusan rekaman. Selain itu akan dilakukan pengujian kuantitatif pada teknik yang dibangun untuk mengetahui akurasi dan performa sewaktu pengisian data, pengubahan data, penghapusan data, pelaporan data utama, data rincian, data transaksi, dan pelaporan transaksi.

Pemilihan metode *timestamped* pada CDC ini didasarkan pada kesederhanaan penambahan atribut di setiap entitas runtutan. Dengan menggunakan metode ini, secara visual bisa dilihat runtutan data yang terbentuk dengan mudah hanya dengan mengurutkan rekaman menurut kunci utamanya.

II. SISTEM PENGELOLAAN *DATABASE*

A. Relational Database Management System

Relational Database Management System (RDBMS) adalah *Database Management System* (DBMS) yang berdasarkan pada model relasional yang ditemukan oleh Edgar F. Codd (1970), ketika bekerja pada IBM San Jose Research Laboratory [5]. Saat ini RDBMS merupakan pilihan yang paling banyak digunakan untuk menyimpan informasi dalam basis data pada sistem finansial, manufaktur dan logistik, data personal, dan sebagainya.

B. Model Relasional

Model relasional adalah model basis data yang berdasarkan pada logika predikat orde pertama, dan pertama diformulasikan dan diproposalkan pada tahun 1969 oleh Edgar F. Codd. Di dalam model ini, semua data direpresentasikan dalam bentuk *tupple* dan digolongkan dalam relasi [6].

Tujuan utama dari model relasional adalah untuk menyediakan metode deklaratif untuk menunjukkan data dan *query* [7]. Pengguna menyatakan secara langsung informasi yang terkandung dalam basis data dan informasi yang diperlukan darinya, sementara RDBMS akan mendeskripsikan struktur data untuk proses penyimpanan dan pengambilannya.

C. Change Data Capture

Change Data Capture (CDC) digunakan untuk melakukan identifikasi atau penangkapan hasil pengubahan data pada basis data, sehingga hasil pengubahan tersebut dapat diinformasikan ke semua pengguna atau sistem lain yang memerlukan informasi tentangnya [3].

CDC dipakai pada *real-time data warehousing* dan pada *event-driven architectures* [8].

D. Real-Time Change Data Capture

Salah satu tujuan *real-time* CDC adalah untuk memenuhi tujuan utama dari CDC tanpa interupsi dari proses yang terjadi di dalamnya. Selain itu, *real-time* CDC harus secara otomatis mengijinkan pengubahan data untuk berpindah dari sistem sumber menuju sistem tujuan. Jadi secara umum suatu proses CDC harus mengetahui ke mana mengirim data yang berubah tersebut [3].

Ada beberapa metode untuk melakukan *real-time* CDC, seperti *log-scanning* (mencatat hasil pengubahan), *trigger* (menggunakan *trigger* dalam tabel), dan *timestamped* (memberi informasi waktu pada rekaman) [4].

E. Extract-Transform-Load

Extract-Transform-Load (ETL) adalah istilah umum yang digunakan pada *data warehousing* yang berarti melakukan ekstraksi (*extraction*) data dari sistem sumber, transformasi (*transformation*) data sesuai dengan aturan, dan pengambilan (*loading*) data menuju ke sistem tujuan. ETL adalah proses dalam *data warehousing* yang menangani pengumpulan data dari sistem sumber dan menempatkannya pada sistem tujuan [4].

F. ETL Tools

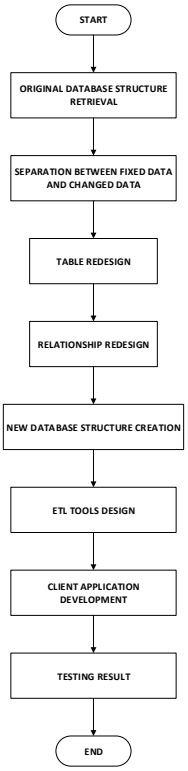
ETL *Tools* adalah sekumpulan *tool* yang dipakai untuk melakukan eksekusi dan manajemen pada proses ETL. Implementasi dari ETL *tools* bisa bermacam-macam, di dalam RDBMS terdapat *tools* seperty *trigger*, *query*, *view*, ataupun *stored procedure* yang bisa digunakan sebagai ETL tools. Dengan adanya ETL *tools*, maka aplikasi atau pemakai dari luar RDBMS tidak perlu untuk melakukan pemrosesan pada entitas secara langsung [4]. Aplikasi atau pemakai luar cukup menjalankan operasi dari ETL *tools* tanpa harus mengetahui langkah-langkah proses yang ada di dalamnya.

III. METODOLOGI

Tahapan penelitian dimulai dari pengambilan struktur basis data awal yang sudah dinormalisasi, kemudian dilanjutkan dengan inventarisasi pemisahan antara atribut tetap dengan atribut berubah pada setiap entitas, dan melakukan transformasi dari setiap entitas yang memiliki atribut berubah (memodifikasi entitas dan relasi), sehingga terbentuk struktur basis data baru.

Tahap berikutnya adalah merancang antarmuka menggunakan *stored procedures* sebagai ETL *tools* untuk

melakukan akses khusus pada entitas yang telah ditransformasi, membuat aplikasi contoh untuk melakukan akses *stored procedures* tersebut, dan terakhir melakukan pengujian terhadap performa struktur yang telah dimodifikasi tersebut. Gbr. 1 menunjukkan langkah-langkah yang dilakukan.



Gbr. 1 Diagram metodologi.

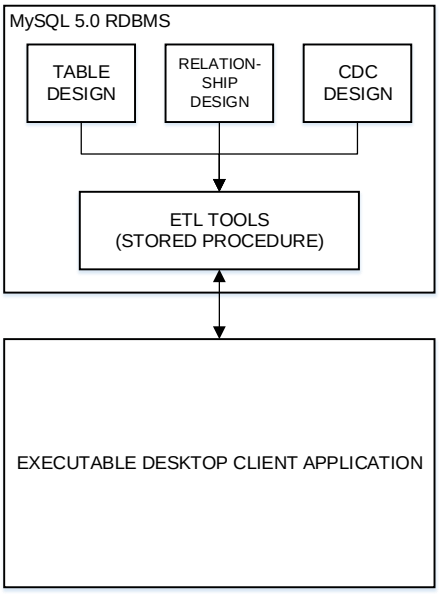
Sedangkan kerangka teori dalam penelitian ini berupa RDBMS menggunakan MySQL 5.0 yang menjadi sentral perancangan *timestamped* CDC dan modifikasinya, berikut ETL *tools* menggunakan *stored procedures*. Gbr. 2 menunjukkan kerangka teori penelitian tersebut. Dari luar RDBMS dilakukan akses melewati *stored procedures* tersebut dan bisa menggunakan aplikasi *desktop*, *mobile*, ataupun *web*.

A. Pengambilan Struktur Basis Data Awal

Setiap rancangan basis data membutuhkan cara yang unik untuk mengimplementasikan *timestamped* CDC. Oleh karena itu harus diambil satu kasus pada rancangan basis data yang ada di dunia nyata. Dalam kasus ini, diambil rancangan basis data pada sistem reservasi hotel atau yang sering disebut dengan *Central Reservation System* (CRS). Pada kasus CRS, basis data harus memiliki informasi tentang paket kebijakan sewa kamar pada periode tertentu, dan akan dipakai pada proses reservasi sesuai dengan tanggal yang berlaku pada paket penjualan tersebut.

B. Pemisahan Atribut Tetap dan Atribut Berubah

Pada sebuah struktur basis data di dalam setiap entitas terdapat atribut tetap dan atribut berubah. Di dalam proses sebelum merancang CDC, diperlukan inventarisasi sifat atribut dari masing-masing entitas sesuai dengan kebijakan (*business rule*) untuk menentukan mana saja yang akan dibuat tetap dan berubah. Proses inventarisasi ini sangat penting karena setiap atribut berubah memerlukan lebih banyak sumber daya dibandingkan atribut tetap.



Gbr. 2 Kerangka teori penelitian.

Objektif dari pemisahan atribut tersebut adalah menerapkan CDC tetapi harus mempertimbangkan beberapa atribut yang memang tidak perlu dimasukkan dalam kategori atribut berubah. Sebagai contoh, untuk data barang dalam minimarket terdapat atribut berubah yaitu harga. Sedangkan kode barang, nama barang, dan satuan merupakan atribut tetap. Tidak perlu semuanya dijadikan atribut berubah karena memakan lebih banyak sumber daya. Sedangkan untuk contoh kasus yang diangkat nantinya ada beberapa kategori atribut berubah di beberapa entitas dan digunakan kode yang sama.

C. CDC Design (Table and Relationship Redesign)

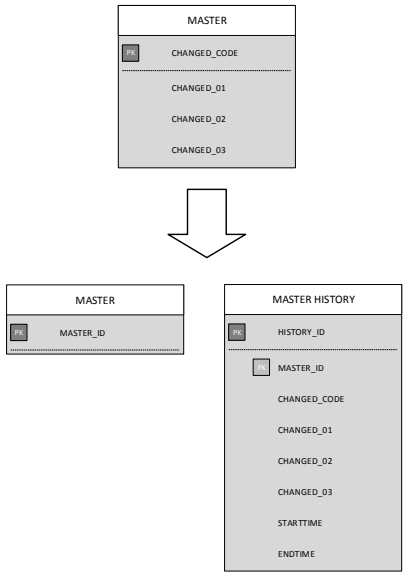
Di dalam sebuah entitas asal yang mana proses CDC akan diimplementasikan, bisa terjadi dua kemungkinan. Kemungkinan pertama semua atribut dalam entitas tersebut merupakan atribut berubah.

Sedangkan kemungkinan kedua dalam entitas tersebut gabungan antara atribut tetap dengan atribut berubah. Transformasi antara entitas asal dengan semua atribut merupakan atribut berubah harus menambahkan satu atribut kunci utama yang harus bernilai tetap. Untuk memudahkan agar pihak pemakai tidak perlu mengetahui bahwa terdapat satu atribut yang ditambahkan maka bisa digunakan *Auto-Increment field*.

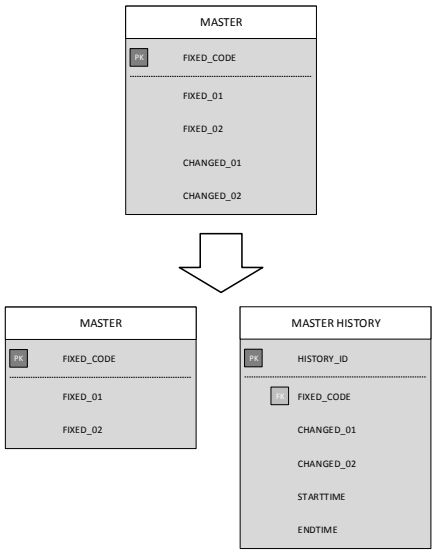
Atribut ini memiliki tipe bilangan bulat (*integer*) dan ditambahi dengan fitur khusus dalam RDBMS yang memungkinkan nilainya secara otomatis akan dibuat penambahan secara urut setiap kali dilakukan penambahan rekaman baru. Gbr. 3 menunjukkan transformasi tersebut.

Sedangkan bila di dalam entitas asal terdapat gabungan atribut tetap dan atribut berubah, dan atribut kunci utama pada

entitas asal merupakan atribut tetap, maka transformasi yang terjadi ditunjukkan pada Gbr. 4.



Gbr. 3 Transformasi atribut asal menuju struktur *timestamped* CDC, dengan seluruh atribut merupakan atribut berubah.

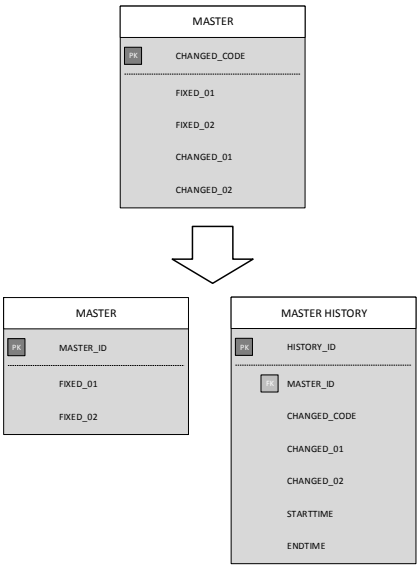


Gbr. 4 Transformasi entitas asal menuju struktur *timestamped* CDC, dengan gabungan atribut tetap dan atribut berubah, dan kunci utama sebagai atribut tetap.

Dengan gabungan atribut tetap dan atribut berubah berada di dalam entitas asal, dan atribut kunci utama pada entitas asal merupakan atribut berubah, maka transformasi yang terjadi ditunjukkan pada Gbr. 5.

D. ETL Tools Creation

Semua akses ke entitas secara fisik diserahkan kepada ETL *tools*, yang dalam hal ini berupa *stored procedure*. Pengguna atau aplikasi menggunakan antarmuka *stored procedure* untuk melakukan akses logis pada entitas.



Gbr. 5 Transformasi entitas asal menuju struktur *timestamped* CDC, dengan gabungan atribut tetap dan atribut berubah, dan kunci utama sebagai atribut berubah.

Pada entitas rincian, tidak disertakan CDC karena hanya diperlukan pada entitas utama. Runtutan pemasukan-pengubahan-penghapusan pada entitas rincian dianggap satu kali akses yang menggunakan proses CDC pada entitas utama. Sedangkan pada entitas transaksi, sebagaimana aturan bisnis yang berlaku, tidak diperbolehkan melakukan pengubahan dan penghapusan data.

```
PROCEDURE LOGIC_INSERT_UNPLANNED(
  IN MasterCode,
  IN MasterStaticComponents,
  IN MasterDynamicComponents,
  OUT TimeSpend DATETIME)
BEGIN
  DECLARE parStartProc, parEndProc AS DATETIME;
  DECLARE parNow, parInfinity AS DATETIME;
  SET parStartProc = SYSDATE();
  SET parInfinity = INFINITETIME();
  SET parNow = NOW();
  INSERT INTO MASTER VALUES
    (MasterCode,
     MasterStaticComponent);
  INSERT INTO MASTER_HISTORY VALUES
    (MasterCode,
     MasterDynamicComponents,
     parNow, parInfinity);
  SET parEndProc = SYSDATE();
  SET TimeSpend = parEndProc - parStartProc;
END
```

Gbr. 6 Antarmuka akses penambahan data baru untuk pengubahan data tidak terencana.

Antarmuka pemasukan data baru pada Gbr. 6 digunakan bila proses pengubahan data tidak terencana. Hal ini dilakukan dengan cara memberi nilai START_TIME dengan *timestamp* sekarang dan nilai END_TIME dengan InfiniteTime. Sehingga akhir berlaku dari rekaman tersebut dianggap tidak terdefinisi.

```
PROCEDURE LOGIC_INSERT_PLANNED(
  IN MasterCode,
  IN MasterStaticComponents,
  IN MasterDynamicComponents,
  IN StartTimePlanned AS DATETIME,
  IN EndTimePlanned AS DATETIME,
  OUT TimeSpend AS DATETIME)
BEGIN
  DECLARE parStartProc, parEndProc AS DATETIME;
  SET parStartProc = SYSDATE();
  INSERT INTO MASTER VALUES
    (MasterCode,
     MasterStaticComponent);
  INSERT INTO MASTER_HISTORY VALUES
    (MasterCode,
     MasterDynamicComponents,
     StartTimePlanned, EndTimePlanned);
  SET parEndProc = SYSDATE();
  SET TimeSpend = parEndProc - parStartProc;
END
```

Gbr. 7 Antarmuka akses penambahan data baru untuk pengubahan data yang terencana.

Sedangkan bila proses pengubahan data sudah direncanakan pada periode tertentu, maka nilai START_TIME diisi dengan *timestamp* di mana masa berlaku rekaman tersebut dimulai, dan nilai END_TIME diisi dengan *timestamp* di mana masa berlaku rekaman tersebut habis. Gbr. 7 merupakan antarmuka pemasukan data baru bila proses pengubahan data direncanakan.

```
PROCEDURE LOGIC_EDIT_UNPLANNED(
  IN MasterCode,
  IN MasterDynamicComponents,
  OUT TimeSpend AS DATETIME)
BEGIN
  DECLARE parStartProc, parEndProc AS DATETIME;
  DECLARE parNow, parInfinity AS DATETIME;
  SET parStartProc = SYSDATE();
  SET parInfinity = INFINITETIME();
  SET parNow = NOW();
  UPDATE MASTER_HISTORY
    SET END_TIME = parNow
    WHERE
      MASTER_HISTORY.MasterCode =
      MasterCode AND
      MASTER_HISTORY.END_TIME =
      parInfinity;
  INSERT INTO MASTER_HISTORY VALUES
    (MasterCode,
     MasterDynamicComponents,
     parNow, parInfinity);
  SET parEndProc = SYSDATE();
  SET TimeSpend = parEndProc - parStartProc;
END
```

Gbr. 8 Antarmuka akses pengubahan data yang sudah ada untuk pengubahan data tidak terencana.

Antarmuka untuk proses pengubahan data yang sudah ada pada Gbr. 8 digunakan bila proses pengubahan data tidak terencana. Hal ini dilakukan dengan cara mengubah nilai END_TIME pada rekaman runtutan sebelumnya (END_TIME dengan nilai InfiniteTime) dengan *timestamp* sekarang, kemudian menambahkan rekaman baru yang memiliki nilai START_TIME sebagai *timestamp* sekarang dan nilai END_TIME sebagai *timestamp* dengan nilai InfiniteTime.

Bila proses pengubahan data sudah direncanakan pada periode tertentu, maka dibuat rekaman terbaru pada entitas runtutan utama, dengan nilai START_TIME diisi dengan *timestamp* di mana masa berlaku rekaman tersebut dimulai, dan nilai END_TIME diisi dengan *timestamp* di mana masa berlaku rekaman tersebut habis.

```
PROCEDURE LOGIC_EDIT_PLANNED(
  IN MasterCode,
  IN MasterDynamicComponents,
  IN StartTimePlanned AS DATETIME,
  IN EndTimePlanned AS DATETIME,
  OUT TimeSpend AS DATETIME)
BEGIN
  DECLARE parStartProc, parEndProc AS DATETIME;
  SET parStartProc = SYSDATE();
  INSERT INTO MASTER_HISTORY VALUES
    (MasterCode,
     MasterDynamicComponents,
     StartTimePlanned, EndTimePlanned);
  SET parEndProc = SYSDATE();
  SET TimeSpend = parEndProc - parStartProc;
END
```

Gbr. 9 Antarmuka akses pengubahan data yang sudah ada untuk pengubahan data yang terencana.

```
PROCEDURE LOGIC_DELETE_UNPLANNED(
  IN MasterCode,
  OUT TimeSpend AS DATETIME)
BEGIN
  DECLARE parStartProc, parEndProc AS DATETIME;
  DECLARE parNow, parInfinity AS DATETIME;
  SET parStartProc = SYSDATE();
  SET parInfinity = INFINITETIME();
  SET parNow = NOW();
  UPDATE MASTER_HISTORY
    SET END_TIME = parNow
    WHERE
      MASTER_HISTORY.MasterCode =
      MasterCode AND
      MASTER_HISTORY.END_TIME =
      parInfinity;
  SET parEndProc = SYSDATE();
  SET TimeSpend = parEndProc - parStartProc;
END
```

Gbr. 10 Antarmuka akses penonaktifan data yang sudah ada untuk pengubahan data tidak terencana.

Rekaman yang sebelumnya pada entitas runtutan utama tidak diubah sama sekali. Gbr. 9 merupakan antarmuka untuk proses pengubahan data lama bila proses pengubahan data tersebut direncanakan.

Antarmuka penonaktifan data yang sudah ada pada Gbr. 10 digunakan bila proses pengubahan data tidak terencana. Hal ini dilakukan dengan cara mengubah nilai END_TIME pada rekaman runtutan sebelumnya (END_TIME dengan nilai InfiniteTime) dengan *timestamp* sekarang. Sedangkan bila proses pengubahan data sudah direncanakan pada periode tertentu, maka tidak perlu dilakukan akses sama sekali, karena pengisian nilai END_TIME pada rekaman sebelumnya dengan nilai habisnya masa berlaku sudah didefinisikan pada proses pengubahan terakhir.

Untuk melakukan akses pada pelaporan, ada dua varian yang disajikan, yaitu pelaporan data utama (dengan CDC tampil sesuai dengan periode yang berlaku), dan pelaporan data transaksi (dengan CDC tampil sesuai dengan waktu transaksi yang berupa sub-jangkauan antara awal dan akhir pelaporan).

```
PROCEDURE MASTER_BROWSING(
  OUT TimeSpend AS DATETIME)
BEGIN
  DECLARE parStartProc, parEndProc AS DATETIME;
  DECLARE parNow AS DATETIME;
  SET parStartProc = SYSDATE();
  SET parNow = NOW();
  SELECT
    A.MasterCode, A.MasterStaticComponents,
    B.MasterDynamicComponents
  FROM
    MASTER AS A, MASTER_HISTORY AS B
  WHERE
    (A.MasterCode = B.MasterCode) AND
    (B.START_TIME <= parNow) AND
    (B.END_TIME > parNow)
  SET parEndProc = SYSDATE();
  SET TimeSpend = parEndProc - parStartProc;
END
```

Gbr. 11 Antarmuka akses pelaporan data utama.

Sesuai dengan Gbr. 11, untuk menampilkan pelaporan data utama yang sedang aktif saat ini diperlukan pembanding antara *timestamp* sekarang dengan START_TIME dan END_TIME. Rekaman yang ditampilkan hanyalah yang yang memiliki START_TIME lebih dari sama dengan sekarang dan END_TIME kurang dari sekarang.

```
PROCEDURE TRANSACTION_REPORTING(
  IN StartingTimeStamp,
  IN EndingTimeStamp,
  OUT TimeSpend AS DATETIME)
BEGIN
  DECLARE parStartProc, parEndProc AS DATETIME;
  SET parStartProc = SYSDATE();
  SELECT
    C.TransID, C.TransTimeStamp,
    A.MasterCode,
    B.MasterDynamicComponents
  FROM
    MASTER AS A, MASTER_HISTORY AS B,
    TRANSACTION AS C
  WHERE
    (A.MasterCode = B.MasterCode) AND
    (B.HistoryID = C.HistoryID) AND
    (C.TRANS_TIME >=
     B.StartTimeStamp) AND
    (C.TRANS_TIME <
     B.EndingTimeStamp);
  SET parEndProc = SYSDATE();
  SET TimeSpend = parEndProc - parStartProc;
END
```

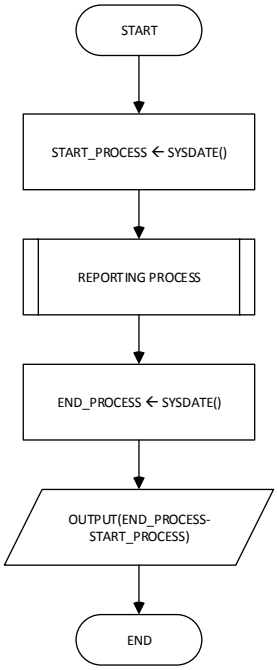
Gbr. 12 Antarmuka akses untuk pelaporan data transaksi.

Untuk menampilkan pelaporan data transaksi, seperti pada Gbr. 12, digunakan atribut HistoryID yang ada pada entitas transaksi yang langsung dihubungkan dengan atribut HistoryID yang ada pada entitas runtutan utama.

E. Strategi Pengujian

Proses pengujian dilakukan dengan menambahkan pencacah waktu pada awal setiap proses pelaporan yang dibandingkan dengan pencacah waktu pada akhir setiap proses pelaporan. Gbr. 13 menyajikan strategi pengujian yang dilakukan pada setiap pelaporan.

Dengan membandingkan hasil antara END_PROCESS dengan START_PROCESS, maka didapatkan waktu (kecepatan) yang diperlukan proses pelaporan tersebut. Pada Gbr. 6 hingga Gbr. 12 proses pembandingan ini sudah disertakan dan hasil pengujian bisa diakses oleh pemakai atau aplikasi sebagai parameter *output* dari *stored procedure* yang bersangkutan.



Gbr. 13 Strategi pengujian untuk setiap proses pelaporan.

Pada variabel START_PROCESS dan END_PROCESS, digunakan SYSDATE() untuk mendapatkan *timestamp* sekarang. Hal ini berbeda dengan NOW() yang digunakan pada setiap antarmuka untuk mengambil *timestamp* sekarang. Perbedaannya adalah: SYSDATE() mengembalikan nilai *timestamp* sewaktu dieksekusi, sedangkan NOW() mengembalikan nilai *timestamp* sewaktu rentetan perintah dalam *stored procedure* mulai dieksekusi.

IV. HASIL DAN PEMBAHASAN

Sesuai dengan metode penelitian, langkah awal yang dilakukan adalah mengambil struktur basis data awal. Struktur basis data ini diambil dari kasus CRS pada Purnama Hotel, Batu.

A. Struktur Basis Data Awal

Pada kasus CRS ini, telah diambil struktur basis data awal dengan total jumlah entitas mencapai 62 entitas. Di antara semuanya ini terdapat beberapa entitas yang memerlukan CDC, tetapi ada juga yang tidak memerlukannya. Berikut ini kelompok entitas yang terbentuk di awal.

1. Entitas yang berkaitan dengan manajemen pemakai: um_modules, um_groups, um_groupmodules, dan um_users.
2. Entitas yang berkaitan dengan ketersediaan kamar (*rooms*): t_m_roomtypes dan t_m_rooms.
3. Entitas yang berkaitan dengan ketersediaan fasilitas bukan kamar (*facilities*): t_m_facilitytypes, t_m_setupcapabilities, dan t_m_facilities.

4. Entitas yang berkaitan dengan paket standar (*standard rates*): t_d_roomrates, t_d_roomrate_bd, t_d_facilityrates, dan t_d_facilityrate_bd.
5. Entitas yang berkaitan dengan paket grup (*pax-based rates*): t_d_grouprates, t_d_grouprate_bd, t_r_permittedroomtypes, t_r_permittedfacilitytypes, dan t_r_bonusrooms.
6. Entitas yang berkaitan dengan barang: t_m_goods.
7. Entitas yang berkaitan dengan inventaris: t_m_invtypes dan t_m_invs.
8. Entitas yang berkaitan dengan produk restoran (*food and beverage products*): t_m_fbproducts dan t_r_fbingredients.
9. Entitas yang berkaitan dengan layanan jasa (*services*): t_m_services.
10. Entitas yang berkaitan dengan ketersediaan *outlet*: t_m_outlets.
11. Entitas yang berkaitan dengan reservasi, pemesanan kamar/fasilitas, dan penjadwalan pemeliharaan kamar/fasilitas, termasuk proses *check-in* dan *check-out*: t_t_reservations, t_d_reservations, t_d_reservationguests, t_d_roomblockings, t_d_guestlistroom, t_d_facilityblockings, t_d_guestlistfacility, t_d_reservationmemos, t_d_reservationpayments, dan t_d_reservationbalances.
12. Entitas yang berkaitan dengan *supplier* barang: t_m_suppliers.
13. Entitas yang berkaitan dengan pembelian barang: t_t_goodpurchasings dan t_d_goodpurchasings.
14. Entitas yang berkaitan dengan pembelian inventaris: t_t_invpurchasings dan t_d_invpurchasings.
15. Entitas yang berkaitan dengan *order* dan penjualan barang: t_t_salesgoodorders, t_d_salesgoodorders, t_t_goodsales, dan t_d_goodsales.
16. Entitas yang berkaitan dengan *order* dan penjualan produk restoran: t_t_salesproductorders, t_d_salesproductorders, t_t_productsales, dan t_d_productsales.
17. Entitas yang berkaitan dengan *order* dan pengerjaan layanan: t_t_salesserviceorders, t_d_salesserviceorders, t_t_servicesales, dan t_d_servicesales.
18. Entitas yang berkaitan dengan mutasi barang dalam hotel: t_t_goodflows dan t_d_goodflows.
19. Entitas yang berkaitan dengan mutasi inventaris dalam hotel: t_t_invflows dan t_d_invflows.
20. Entitas yang berkaitan dengan pembuangan barang: t_t_gooddumps dan t_d_gooddumps.
21. Entitas yang berkaitan dengan pembuangan produk restoran: t_t_productdumps dan t_d_productdumps.
22. Entitas yang berkaitan dengan pembuangan inventaris: t_t_invdumps dan t_d_invdumps.

B. Pemisahan Data Tetap dan Data Berubah

Basis data CRS yang memiliki 62 entitas tersebut telah diinventarisasi dan ditemukan enam bagian yang memerlukan *timestamped* CDC, di antaranya:

1. Manajemen paket standard.
2. Manajemen paket grup.
3. Manajemen harga barang.
4. Manajemen harga produk restoran.
5. Manajemen harga layanan.
6. Manajemen nilai penyusutan inventaris.

Oleh karena itu, diprediksi akan diperlukan enam entitas tambahan sehingga total semua entitas setelah implementasi *timestamped* CDC adalah 68 entitas.

C. Transformasi Entitas (CDC Design)

Enam bagian pada basis data CRS tersebut telah diinventarisasi, dan disimpulkan terdapat tambahan enam entitas runtutan utama sesuai dengan kaidah transformasi yang telah dibahas sebelumnya.

TABEL I
ENTITAS HASIL CDC DARI MANAJEMEN PAKET STANDARD

Nama Field	Tipe Data	Ukuran	Foreign Key
StandardRatePlannerID	int	AUTO	PRIMARY KEY
PublishedPeriodName	varchar	50	
InternalPeriodName	varchar	50	
Description	varchar	255	
StartTime	datetime		
EndTime	datetime		
UserName	varchar	50	um_users

Tabel I menunjukkan entitas hasil *timestamped* CDC untuk manajemen paket standard. Entitas runtutan utama tersebut disambungkan ke dua entitas utama, yaitu_d_roomrates dan t_d_facilityrates.

TABEL II
ENTITAS HASIL CDC DARI MANAJEMEN PAKET GRUP

Nama Field	Tipe Data	Ukuran	Foreign Key
GroupRatePlannerID	int	AUTO	PRIMARY KEY
PublishedPeriodName	varchar	50	
InternalPeriodName	varchar	50	
Description	varchar	255	
StartTime	datetime		
EndTime	datetime		
UserName	varchar	50	um_users

TABEL III
ENTITAS HASIL CDC DARI MANAJEMEN HARGA BARANG

Nama Field	Tipe Data	Ukuran	Foreign Key
GoodPricingID	int	AUTO	PRIMARY KEY
GoodCode	varchar	15	t_m_goods
SellPrice	float		
StartTime	datetime		
EndTime	datetime		
UserName	varchar	50	um_users

Tabel II menunjukkan entitas hasil *timestamped* CDC untuk manajemen paket grup. Entitas runtutan utama tersebut disambungkan ke satu entitas utama, yaitu t_d_grouprates.

Tabel III menunjukkan entitas hasil *timestamped* CDC untuk manajemen harga barang. Entitas runtutan utama tersebut disambungkan ke satu entitas utama, yaitu t_m_goods.

TABEL IV
ENTITAS HASIL CDC DARI MANAJEMEN HARGA PRODUK RESTORAN

Nama Field	Tipe Data	Ukuran	Foreign Key
FBProductPricingID	int	AUTO	PRIMARY KEY
FBProductCode	varchar	15	t_m_fbproducts
SellPrice	float		
StartTime	datetime		
EndTime	datetime		
UserName	varchar	50	um_users

Tabel IV menunjukkan entitas hasil *timestamped* CDC untuk manajemen harga produk restoran. Entitas runtutan utama tersebut disambungkan ke satu entitas utama, yaitu t_m_fbproducts.

TABEL V
ENTITAS HASIL CDC DARI MANAJEMEN HARGA LAYANAN

Nama Field	Tipe Data	Ukuran	Foreign Key
ServicePricingID	int	AUTO	PRIMARY KEY
ServiceCode	varchar	15	t_m_services
UnitQuantum	float		
PricePerQuantum	float		
SellPrice	float		
StartTime	datetime		
EndTime	datetime		
UserName	varchar	50	um_users

Tabel V menunjukkan entitas hasil *timestamped* CDC untuk manajemen harga layanan. Entitas runtutan utama tersebut disambungkan ke satu entitas utama, yaitu t_m_services.

TABEL VI
ENTITAS HASIL CDC DARI MANAJEMEN NILAI PENYUSUTAN INVENTARIS

Nama Field	Tipe Data	Ukuran	Foreign Key
InvValueID	int	AUTO	PRIMARY KEY
InventoryCode	varchar	20	t_m_invs
CurrentValue	float		
StartTime	datetime		
EndTime	datetime		
UserName	varchar	50	um_users

Tabel VI menunjukkan entitas hasil *timestamped* CDC untuk manajemen nilai penyusutan inventaris. Entitas runtutan utama tersebut disambungkan ke satu entitas utama, yaitu t_m_invs.

D. Pengujian Performa

Hasil transformasi tersebut diuji performanya untuk proses penambahan data baru, pengubahan data, penonaktifan data, dan pelaporan. Karena masing-masing atribut START_TIME dan END_TIME dalam struktur basis data setelah implementasi *timestamped* CDC harus diberi indeks, maka untuk proses penambahan data, pengubahan data, dan penonaktifan data akan mengalami sedikit *delay* dibandingkan bila tanpa menggunakan CDC.

Pengujian akan dilakukan dua kali, pertama untuk struktur awal, dan kedua untuk struktur yang telah ditransformasi. Kemudian dilakukan perbandingan antara keduanya.

Pengujian untuk penambahan data, pengubahan data, dan penonaktifan data masing-masing dilakukan menggunakan 1000 data acak dan dihitung akumulasi waktunya karena jika hanya menggunakan satu data perbedaan performa tidak terdeteksi (sela waktu kurang dari 1 ms).

TABEL VII
PENGUJIAN AKSES 1000 DATA ACAK UNTUK PENAMBAHAN DATA

Subjek Uji	Non-CDC	Modified Timestamped CDC
Paket Standard	250 ms	1203 ms
Paket Grup	217 ms	1140 ms
Harga Barang	196 ms	1016 ms
Harga Produk Restoran	180 ms	1020 ms
Harga Layanan	186 ms	1072 ms
Penyusutan Inventaris	185 ms	1023 ms

Tabel VII menunjukkan akumulasi waktu yang diperlukan untuk penambahan data sebanyak 1000 data acak. Terlihat dari hasil tersebut, rata-rata *timestamped* CDC memerlukan waktu sekitar lima kali lebih lama dibandingkan struktur konvensional ternormalisasi.

TABEL VIII
PENGUJIAN AKSES 1000 DATA ACAK UNTUK PENGUBAHAN DATA

Subjek Uji	Non-CDC	Modified Timestamped CDC
Paket Standard	210 ms	1866 ms
Paket Grup	208 ms	1609 ms
Harga Barang	201 ms	1712 ms
Harga Produk Restoran	205 ms	1802 ms
Harga Layanan	209 ms	1693 ms
Penyusutan Inventaris	196 ms	1565 ms

Tabel VIII menunjukkan akumulasi waktu yang diperlukan untuk melakukan pengubahan data sebanyak 1000 data acak. Terlihat dari hasil tersebut, rata-rata *timestamped* CDC memerlukan waktu sekitar delapan kali lebih lama dibandingkan struktur konvensional ternormalisasi. Pada struktur *timestamped* CDC, proses pengubahan data melakukan dua kali proses (menambah rekaman runtutan dan mengubah rekaman runtutan terdahulu), sehingga proses ini lebih lama dibandingkan proses penambahan data baru.

Tabel IX menunjukkan akumulasi waktu yang diperlukan untuk penghapusan (penonaktifan) data sebanyak 1000 data acak. Terlihat dari hasil tersebut, rata-rata *modified timestamped* CDC memerlukan waktu sekitar delapan kali lebih lama dibandingkan struktur konvensional ternormalisasi. Pada struktur *modified timestamped* CDC, proses penghapusan data melakukan dua kali proses (menambah rekaman runtutan dan mengubah rekaman runtutan terdahulu) hanya pada proses pengubahan yang tak terencana (harga barang, harga produk restoran, harga layanan, dan penyusutan

inventaris), sehingga proses ini lebih lama dibandingkan proses penambahan data baru. Sedangkan pada proses pengubahan terencana (paket standard dan paket grup), proses hanya membutuhkan satu kali pengubahan, sehingga rata-rata memerlukan waktu hampir sama dengan penambahan data baru.

TABEL IX
PENGUJIAN AKSES 1000 DATA ACAK UNTUK PENGHAPUSAN (PENONAKTIFAN) DATA

Subjek Uji	Non-CDC	Modified Timestamped CDC
Paket Standard	190 ms	1121 ms
Paket Grup	188 ms	1240 ms
Harga Barang	209 ms	1925 ms
Harga Produk Restoran	175 ms	1790 ms
Harga Layanan	182 ms	1926 ms
Penyusutan Inventaris	170 ms	1998 ms

TABEL X
PENGUJIAN PENAMPILAN 1000 DATA ACAK UNTUK PELAPORAN

Subjek Uji	Non-CDC	Modified Timestamped CDC
Paket Standard	5 ms	20 ms
Paket Grup	9 ms	18 ms
Harga Barang	8 ms	17 ms
Harga Produk Restoran	9 ms	19 ms
Harga Layanan	6 ms	18 ms
Penyusutan Inventaris	8 ms	18 ms

TABEL XI
KEUNTUNGAN YANG DIDAPATKAN

Jenis Keuntungan	Non-CDC	Previous Timestamped CDC	Modified Timestamped CDC
Pelacakan Penambahan Data Baru	YA	YA	YA
Pelacakan Pengubahan Data	TIDAK	YA	YA
Pelacakan Penghapusan Data	TIDAK	TIDAK	YA
Akurasi 100% untuk Pelaporan	TIDAK	YA	YA
Kecepatan Akses	CEPAT	LAMBAT	LAMBAT

Sedangkan pengujian untuk pelaporan dilakukan untuk menampilkan data sebagaimana subjek uji dan diperlukan bila melakukan proses *real-time* pada *back office* dan *front office*. Dari semua data acak yang telah dimasukkan sebanyak 1000, dilakukan proses penampilan semua data dengan *timestamp* dalam kisaran 31 hari (1 bulan).

Karena menampilkan data dilakukan secara bersamaan (tidak ada proses *indexing*), maka rata-rata jauh lebih cepat daripada akumulasi proses penambahan, pengubahan, dan penghapusan data. Dari Tabel X didapatkan hasil pengujian rata-rata *timestamped* CDC untuk setiap subjek uji dua hingga tiga kali lebih lambat dibandingkan yang konvensional.

Secara kualitatif, Tabel XI menunjukkan perbandingan keuntungan yang didapatkan dari penerapan metode non-CDC, *timestamped* CDC yang telah ada pada penelitian sebelumnya, dan *modified timestamped* CDC yang diulas dalam makalah ini.

V. KESIMPULAN

Berdasarkan hasil pengujian, dapat diambil kesimpulan. Setiap entitas pada metode *modified timestamped* CDC yang berisi data berubah perlu ditambahkan satu entitas untuk mencatatkan runtutan hasil pengubahan data, dan minimal empat *stored procedures* untuk melakukan akses, yaitu penambahan, pengubahan, penghapusan, dan pelaporan. Setiap penerapan metode *modified timestamped* CDC hanya akan mengurangi performa maksimal delapan kali daripada struktur konvensional. Bila dilihat dari waktu pengujian untuk 100 data yang maksimal mendapatkan 2000 ms untuk setiap pengubahan dan 20 ms untuk setiap pelaporan, maka pengurangan performa tersebut masih dalam batas toleransi, dibandingkan dengan manfaat yang diperoleh. Metode *modified timestamped* CDC mampu melacak penghapusan data, menyempurnakan *timestamped* CDC dari penelitian sebelumnya..

REFERENSI

[1] Bagui, Sikha, Earp, Richard, *Database Design Using Entity-Relationship Diagrams*, CRC Press., 2011.
[2] JinGang Shi, YuBin Bao, FangLing Leng, & Ge Yu, "Study on Log-Based Change Data Capture and Handling Mechanism in Real-Time Data Warehouse", *International Conference on Computer Science and Software Engineering*, 479-481, 2008.
[3] Eccles, Mitchell J., Evans, David J., & Beaumont, Anthony J., "True Real-Time Change Data Capture WithWeb Service Database Encapsulation", *IEEE 6th World Congress on Services*: 128-131, 2010.
[4] Tank , Darshan M., Ganatra, Amit, Kosta, Y. P., & Bhensdadia, C K., "Speeding ETL Processing in Data Warehouses Using High-Performance Joins For Changed Data Capture (CDC)", *International Conference on Advances in Recent Technologies in Communication and Computing*, 365-368, 2010.
[5] Hussain, Tauqeer, Shamail, Shafay, & Awais, Mian M., "Eliminating Process of Normalization in Relational Database Design", *Proceedings IEEE INMIC*, 408-413, 2003.
[6] Abiteboul, Serge, Hull, Richard B., & Vianu, Victor, *Foundations of Databases*, Addison-Wesley, 1995.
[7] Springsteel, Frederick N., "Entity-Relationship Logical Design of Database Systems", *IEEE*: 581-582, 1988.
[8] Microsoft Developer Network, 2014, Track Data Changes (SQL Server). [Online] https://msdn.microsoft.com/en-us/library/bb933994.aspx, tanggal akses: 2 Juli 2014.