ANALISIS DAN PENERAPAN METODE TUNING PADA BASIS DATA FUNDING

Indrajani

Information Systems Department, School of Information Systems, BINUS University
Jl.KH.Syahdan no.9 Kemanggisan Palmerah Jakarta Indonesia
indrajani@binus.ac.id

ABSTRACT

The purpose of this research is to analyze, design, and implement tuning for the funding database, which include data mart processing that will be used in the formation of analytical reports. The research method used is literature study from a variety of journals, books, e-books, and articles on the internet. Fact finding techniques are also done by analyzing, collecting, and examining the documents, interviews, and observations. Other the research methods are also used to analyze and design database such as SQL tuning, Partitioning, and Indexing. The results obtained from this research is an implementation tuning for the funding database, in which if it is implemented, it will improve performance significantly. The conclusion is by implementing tuning for the funding database, it can help the bank to improve the performance of database system and decrease time to produce analytical reports.

Keywords: *Database, Tuning, Funding, Analytical Report*

ABSTRAK

Tujuan penelitian ini adalah menganalisis, merancang, dan mengimplementasikan metode tuning pada database funding. Proses pengolahan data mart yang akan digunakan dalam pembentukan analytical report terdapat di dalamnya. Metode penelitian yang digunakan adalah studi kepustakaan dari berbagai jurnal, buku, e-book, dan artikel-artikel di internet. Selain itu juga dilakukan teknik pengumpulan data, yaitu dengan mempelajari dokumen-dokumen dan wawancara serta observasi. Metode analisis dan perancangan model tuning juga digunakan yaitu SQL Tuning, Partitioning, dan Indexing. Hasil yang diperoleh dari penelitian ini ialah suatu implementasi tuning pada database funding yang jika diimplementasikan akan meningkatkan kinerja secara signifikan. Simpulan yang dapat ditarik dari penelitian ini adalah tuning merupakan salah satu faktor penting dalam meningkatkan kinerja sistem basis data.

Kata kunci: *basis data, tuning, funding, analytical report*

PENDAHULUAN

Ada tiga kelompok besar dalam bisnis perbankan yaitu konsumer, bisnis kredit, serta bisnis tresuri dan internasional. Ketiga bisnis tersebut didukung oleh unit-unit pengendali risiko serta pendukung korporasi. Bisnis perbankan konsumer merupakan *core competence* dari suatu bank. Bisnis ini meliputi jasa perbankan konsumer, kartu kredit, fasilitas pembayaran sarana umum serta perbankan prioritas. Bisnis kredit merupakan *core competence* lainnya yang sedang dikembangkan oleh bank, dengan mengandalkan jangkauan jaringan kantor cabang yang luas, kemampuan distribusi lintas jaringan maupun kegiatan pemasaran dan penjualan yang unggul. Bisnis ini menyalurkan kredit korporasi kepada perusahaan dan institusi terkemuka di berbagai sektor industri, kredit komersial maupun ritel bagi sektor usaha kecil dan menengah (UKM), dan kredit pemilikan rumah dan kendaraan bermotor bagi nasabah perorangan.

Dana pihak ketiga adalah unsur terpenting selain kegiatan pengumpulan dana. Seiring dengan berkembangnya kegiatan dan promosi produk perbankan serta tuntutan ketersediaan data yang cepat dan sesuai dengan kebutuhan pengguna, maka sistem yang sudah pernah dikembangkan perlu ditingkatkan kembali dari sisi kinerja. Saat ini, untuk menghasilkan data-data yang dibutuhkan dalam membuat *analytical report* memerlukan waktu sekitar dua sampai tiga minggu untuk data funding selama satu bulan. Pihak manajemen merasa perlu meningkatkan sistem yang telah ada dari sisi kinerja karena dengan waktu yang terpakai menyebabkan *analytical report* yang dihasilkan tidak terbarukan sehingga banyak terjadi kesalahan dalam pengambilan keputusan yang merugikan pihak perusahaan dan nasabah.

Ruang lingkup penelitian ini adalah meningkatkan kinerja dari sisi *Application Tuning*, yaitu upaya memperbaiki kinerja sistem dengan SQL Tuning, *indexing* dan *partitioning*. Adapun tujuan penelitian ini adalah mengoptimalisasikan waktu yang dibutuhkan dalam pembentukan tabel fakta dari *data mart* sehingga pada akhirnya *analytical report* harian dan bulanan data *funding* dapat dihasilkan secara lebih cepat dan sesuai dengan kebutuhan pengguna. Manfaat penelitian ini adalah menyediakan data untuk mendukung kegiatan analisis sebagai dasar pengambilan keputusan oleh pihak managerial secara lebih cepat

METODE

Metode penelitian yang digunakan adalah metode pengumpulan data, metode analisis, dan metode perancangan basis data. Teknik yang digunakan dalam metode pengumpulan data ini mencakup antara lain wawancara, mempelajari dokumen, observasi, dan studi kepustakaan (Indrajani, Wihendro, & Safan, 2013). Wawancara dilakukan pada bagian-bagian yang akan menggunakan sistem basis data yang diusulkan. Pertanyaan-pertanyaan yang akan ditanyakan akan disiapkan terlebih dahulu sebelum wawancara berlangsung. Sifat yang akan digunakan adalah gabungan antara pertanyaan tertutup dan terbuka (Indrajani, 2011). Untuk mendapatkan data dan informasi mengenai kebutuhan pengguna secara lengkap, maka dilakukan juga pengumpulan dokumen-dokumen. Selain itu juga dilakukan observasi langsung ke lapangan (Indrajani, 2011).

Setelah itu desain dan perancangan yaitu merancang alternatif-alternatif solusi, pengembangan yang mencakup proses *create, run, testing, debug* dan *code*. Kemudian implementasi yaitu tahap ketika *code* yang telah di-*tuning* diimplementasikan pada sistem yang berjalan. Terakhir adalah evaluasi yaitu tahap membandingkan *processing time system* lama dengan yang baru.

Data mart adalah bagian dari *data warehouse* yang mendukung kebutuhan informasi dari suatu departemen (Connolly & Begg, 2010). Perbedaan *data mart* dengan *data warehouse* antara lain *Data mart* fokus untuk departemen tertentu saja dan tidak berisi data operasional yang bersifat detail. *Data mart* lebih mudah dipahami karena jumlah data yang lebih sedikit. Alasan Pembuatan *Data mart* adalah untuk memudahkan akses terhadap data-data yang sering digunakan untuk bahan analisis, data yang disajikan *Data mart* lebih spesifik berdasarkan setiap departemen sehingga memudahkan pencarian data, dan biaya yang dibutuhkan untuk membangun *Data mart* lebih kecil dibanding *data warehouse.*

Analytical report adalah laporan yang berisi ringkasan informasi yang ditujukan untuk membantu proses pengambilan keputusan oleh pihak manajemen. Contohnya adalah laporan-laporan yang bersifat multidimensional. Di mana laporan-laporan tersebut dapat dilihat dari berbagai sudut pandang. Misalnya segi nasabah, segi produk, segi waktu, segi cabang, dan sebagainya.

Tuning adalah tindakan memodifikasi sistem dengan tujuan meningkatkan kinerja. Manfaat *tuning* adalah mempercepat *response time* proses dan meningkatkan *throughput* (Chan & Immanuel, 2008). Jenis-jenis *tuning yakni Tuning for response time*, yaitu *tuning* yang ditujukan untuk mempercepat *response time* dan *Tuning for throughput*, yaitu *tuning* yang ditujukan untuk meningkatkan jumlah *throughput process. Application tuning* adalah suatu tindakan yang mayoritas berhubungan dengan perbaikan di sisi sintaks SQL dan memastikan apakah sintaks *query* yang digunakan telah efisien dan optimal.

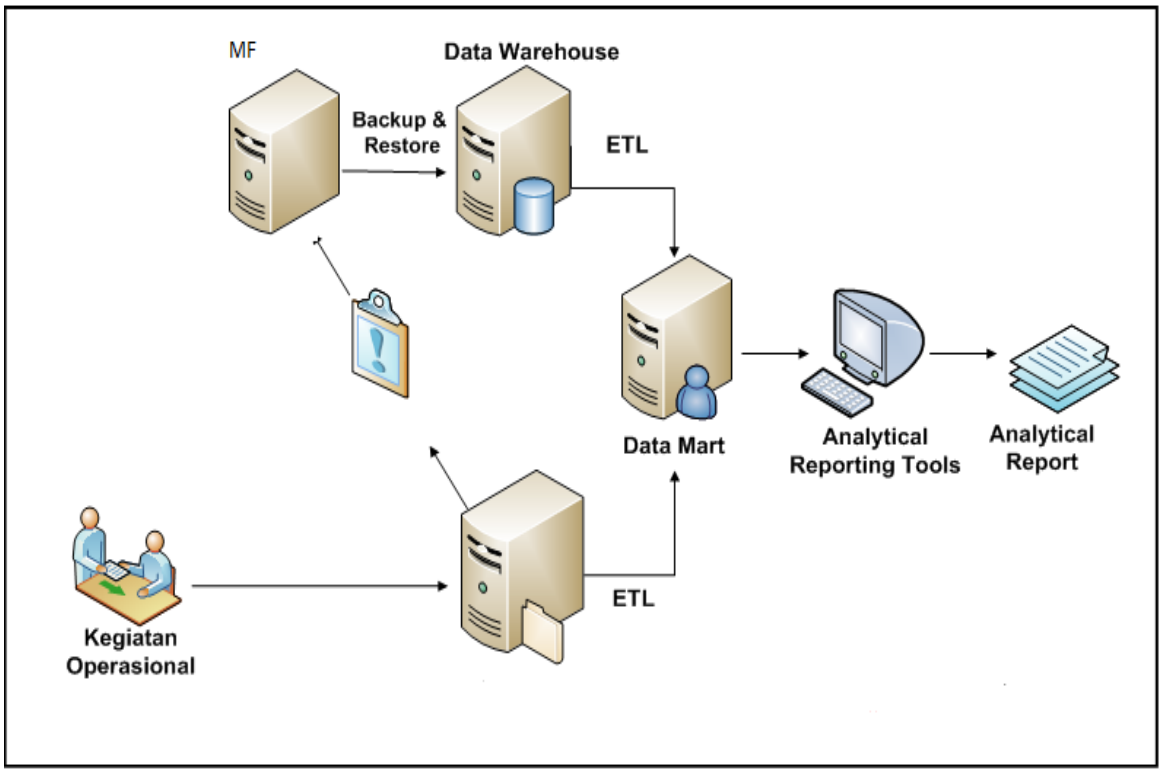
SQL tuning merupakan tindakan *query optimization*. Definisi *Query Optimization* adalah suatu tindakan memilih *execution strategy* yang efisien untuk mengeksekusi sebuah *query*. Kinerja adalah salah satu ukuran terpenting, yang menggambarkan jika proyek ini sukses atau terjadi kesalahan. Berarti desain untuk proses kinerja harus dimulai sejak awal dalam pengembangan basis data. Di sini terdapat beberapa teknik untuk meningkatkan kinerja basis data agar maksimal. Proses *tuning* kinerja mencakup antara lain pengukuran respon waktu sebelum *tuning*, pada waktu *tuning* dilakukan, dan pengukuran respon waktu setelah *tuning* (Cecilia & Mihai, 2011). *Database Administrator* bertanggung jawab untuk meningkatkan kinerja sistem basis data. Deteksi penurunan kinerja dapat dilihat dari parameter kinerja system (Hitesh, Aditya, & Ranjit, 2012).

Partitioning adalah tindakan membagi tabel yang mempunyai *record* data yang sangat besar menjadi segmen-segmen kecil disebut partition berdasarkan sebuah *partitioned key* (Anonim1, 2007). *Oracle* menyediakan lima teknik untuk mempartisi tabel (Indrajani, 2013) yaitu *Range Partitioning, List Partioning, Hash Partitioning, Composite Range-Hash Partitioning,* dan *Composite Range-List Partitioning. Range Partitioning* adalah setiap partisi dispesifikasi oleh serangkaian nilai dari *partitioning key.* Sedangkan *List Partitioning* yaitu setiap partisi dispesifikasikan oleh sebuah daftar nilai dari *key partitioning*. *Hash Partitioning* adalah sebuah algoritma *hash* diterapkan pada *partitioning key* untuk membatasi partisi pada baris yang diberikan. *Composite Range-Hash Partitioning* ialah sebuah kombinasi dari teknik *Range* dan *Hash partitioning*. Pertama kali, tabel akan dipartisi dengan teknik *range partitioning*. Kemudian setiap individu hasil *range partitioning* tersebut akan dipartisi kembali dengan menggunakan *hash partitioning*. Terakhir *Composite Range-List Partitioning* yaitu sebuah kombinasi antara teknik *Range* dan *List Partitioning*. Pertama kali, tabel akan dipartisi dengan teknik *range partitioning*. Kemudian setiap individu hasil *range partitioning* tersebut akan dipartisi kembali dengan menggunakan *list partitioning*. Tidak seperti gabungan *Range-Hash Partitioning,* isi dari setiap subpartisi mewakili sebuah subset logika dari data, dideskripsikan oleh sesuai dengan proses setup pada *range* dan *list partitioning*.

Index adalah *data structure* yang memungkinkan DBMS mengakses *record-record* secara lebih cepat dan meningkatkan *speed response* dari suatu *query* (Millsap & Garry, 2008).

HASIL DAN PEMBAHASAN

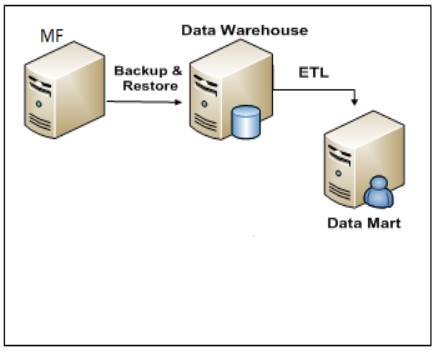
Divisi Pengembanan Produk dan Jasa merupakan salah satu divisi pada perusahaan yang bertugas menangani dana pihak ke tiga oleh nasabah untuk berbagai keperluan, misalnya tabungan, deposito, dan giro. Divisi Teknologi Informasi merupakan salah satu divisi pada perushaan tersebut yang bertugas untuk menangani sisi pengembangan dari aplikasi dan sistem-sistem yang ada. Ke dua divisi tersebut bekerja sama dalam peningkatan kebutuhan laporan-laporan yang dibutuhkan oleh pihak manajemen.



Gambar 1 Proses Pembentukan Analytical Report

Ketika seseorang ingin mengajukan membuka tabungan, deposito, atau giro pada bank, ia harus menjalani prosedur operasional yaitu mengisi formulir pembukaan tabungan deposito, atau giro. Selama permohonan tersebut belum ditandatangani, data tersebut ada di dalam *server* dan terdapat *database server* yang menyimpan informasi calon nasabah bank tersebut.

Data transaksi harian dari database *mainframe* dimasukkan ke *data warehouse* dengan metode *Backup* dan *Restore*. Kemudian dilakukan proses ETL dari *database server operasional* dan *server data warehouse* ke *server data mart* dengan proses ETL (*Extract, Transform, Load*). Proses yang terjadi di dalam *server data mart* adalah pembentukan beberapa tabel fakta dari *data mart* yang kemudian dengan bantuan *Analytical Reporting Tools*, yaitu *Business Object 6.1*, dihasilkan *analytical report* yang digunakan pihak manajemen untuk memantau kinerja dana pihak ke 3 dan mendukung proses pengambilan keputusan.



Gambar 2 Proses Pembentukan *Data Mart*

Data transaksi harian dari *database mainframe* dimasukkan ke *data warehouse* dengan metode *Backup* dan *Restore*. Alasan mengapa harus *backup* dan *restore* adalah karena data yang terdapat dalam *mainframe* terlalu besar, sedangkan *bandwidth* yang disediakan hanya 1 Mbps maka proses tersebut membutuhkan waktu yang cukup lama jika melakukan ETL. Setelah itu dilakukan ETL (*Extract, Transform, Load*) di *data warehouse* dan *database server* sehingga membentuk data mart yang dibutuhkan oleh Divisi Pengembanan Produk dan Jasa.

Untuk membentuk *analytical report* dibutuhkan data histori sedangkan perancangan *data warehouse* berisi data posisi per hari. Proses pembentukan tabel fakta dari *data mart* membutuhkan waktu yang lama yaitu hingga kurang lebih 8-10 jam. Untuk menghasilkan *analytical report* menggunakan *Business Object* membutuhkan waktu yang lama, sehingga pengguna lebih memilih untuk langsung *query* ke *data mart*. Oleh karena itu, pengguna data *funding* harus memiliki keahlian di bidang *SQL code*.

Beberapa Usulan Pemecahan

Melakukan *SQL tuning* pada *procedure-procedure* yang belum optimal. Misalnya mengganti *where clause* menjadi *join clause* untuk menggabungkan beberapa tabel, menggunakan *ROWID* untuk mengakses data, menggunakan *temp table* untuk menggabungkan beberapa statement yang dipergunakan berulang kali.Selain itu dapat juga membuat partisi pada tabel yang mengandung lebih dari 2 GB data dan memberi *index* terhadap data yang sering diakses.

Proses Pengisian Tabel Histori ACCT_DTL pada Sistem yang Berjalan

Data mart pada sistem yang berjalan hanya menyimpan data posisi, maka dibuat sebuah tabel bernama ACCT_DTL dengan struktur penyimpanan *data mart* histori menggunakan logika *bucket-bucket (block column* per bulan) yang menyimpan data bulanan sebanyak 12 *bucket*. Cara pengisian data ke setiap bucket tiap bulannya adalah: (1) Awalnya, *bucket-bucket* pada tabel ACCT_DTL akan digeser terlebih dahulu dengan data lama yang ada pada *bucket* pertama, akan digeser ke *bucket* kedua, *bucket* kedua ke *bucket* ketiga dan begitu seterusnya, meskipun terdapat *bucket* yang belum berisi data. Jadi setiap data pada bulan baru yang masuk akan terjadi 12 kali pergeseran data, jika proses penggeseran tersebut belum pernah dijalankan dengan sukses pada bulan tersebut. (2) Jika data nasabah A belum pernah terdapat dalam sistem, maka data tersebut akan di-INSERT ke dalam tabel ACCT_DTL pada 1 baris baru di *bucket* pertama. Jika pada bulan yang sama nasabah melakukan transaksi, maka data pada *bucket* bulan tersebut akan diubah. (3) Setiap data pada bulan baru akan selalu berada pada *bucket* yang pertama. Alasan mengapa data yang terbaru selalu dimasukkan ke dalam *bucket* pertama adalah karena data yang digunakan untuk menghasilkan *analytical report* bulanan adalah data yang terbaru sehingga dari sisi *scripting*, akan lebih mudah apabila data yang terbaru selalu diletakkan pada *bucket* pertama.

Proses pengisian *record-record* data pada ACCT_DTL menggunakan sebuah PL/SQL *procedure* yaitu DBP_PRE_PROCESS. Pada *procedure* DBP_PRE_PROCESS ini dilakukan proses *insert record-record* baru (jika ada *account* baru yang melakukan peminjaman) dan proses *update record-record* (jika nasabah membayar cicilan)

Proses Pembentukan Tabel Fakta

Sebuah tabel fakta dibentuk dari histori *data mart* (termasuk salah satunya adalah data pada tabel histori ACCT_DTL) dan beberapa tabel dimensi (untuk menyediakan kategori-kategori apa saja yang dapat ditampilkan oleh suatu *analytical report*). Kedua sumber data e tersebut (*data mart* dan tabel dimensi) kemudian diolah melalui suatu *procedure* (setiap tabel fakta diolah oleh *procedure* yang berbeda). Setiap tabel fakta digunakan untuk pembentukan sebuah *analytical report*

SQL Tuning - Restrukturisasi tabel ACCT_DTL

Pada perancangan sistem yang lama, tabel ACCT_DTL menggunakan sistem *bucket*. Proses tersebut dapat menyebabkan penurunan kinerja dari *database* dan juga pemborosan *resource* sehingga kami merancang suatu sistem baru untuk menyimpan *data mart*. Pada perancangan sistem yang baru, kami tidak menggunakan 12 *bucket* untuk menyimpan data histori setiap bulannya. Akan tetapi, hanya menggunakan 1 *bucket* dalam partisi bulanan untuk menyimpan data histori. Struktur tabel ACCT_DTL_ENH:

Atribut Account	*Account Status*	*Month_idx*
Mr. A No rek : xxxxx	Rp 6.000.000	01
Mr. B No rek : xxxxx	Rp 18.000.000	01
Mr. A No rek : xxxxx	Rp 4.000.000	02
Mr. B No rek : xxxxx	Rp 15.000.000	02
Mr. A No rek : xxxxx	...	...
Mr. B No rek : xxxxx	...	...
Mr. A No rek : xxxxx	Rp 6.000.000	12
Mr. B No rek : xxxxx	Rp 6.000.000	12

Data selalu di-*insert* ke *row* baru, berbeda dengan P_ACCT_DTL lama yang selalu menggeser data lama ke *bucket* sebelumnya jika ada data baru yang akan masuk.

Gambar 3 Struktur tabel ACCT_DTL_ENH

Keuntungan yang diperoleh dari perubahan ini adalah mempercepat pengaksesan data pada tabel ACCT_DTL_ENH yang berisi data histori karena adanya *range partitioning* berdasarkan month_idx dan *sub-partition* berupa *hash partitioning* berdasarkan acct_no. Selain itu juga menghilangkan proses *update* yang menggunakan logika perpindahan data ke *bucket* berikutnya setiap bulan. Jadi, perubahan ini membuat prosedur SHIFT_ ACCT_DTL dapat dihilangkan.

Mengganti *where clause* menjadi *join clause* untuk menggabungkan *table*

```
SELECT  last_name, job_name, department_name
FROM employees a JOIN jobs b ON a.job_id = b.job_id
JOIN department c ON a.department id = c.department id
```

```
SELECT  last_name, job_name, department_name
FROM employees a, jobs b, department c
WHERE a.job_id = b.job_id AND a.department_id = c.department_id
```

Gambar 4 Mengganti *where clause*

Karena untuk menggabungkan beberapa tabel dengan *non-join clause (where clause)*, hasil akan dievaluasi setelah melakukan *cross join* seluruh tabel. Sedangkan dengan *join clause*, hasil akan dievaluasi setelah *cross join* antar 2 tabel. Sehingga dapat mengurangi jumlah baris yang akan dicocokkan dengan tabel berikutnya

Menggunakan *Temp Table* untuk Menampung *Join Table* yang Digunakan Berulang Kali

Dalam *script* yang kompleks, seringkali ditemukan penggabungan beberapa tabel. Penggabungan beberapa tabel sekaligus ini dapat menurunkan performa. Setelah kami teliti terdapat beberapa jenis penggabungan tabel yang sering diimplementasikan seperti *Join Multiple Table in one statement* dan *Join Multiple Table using temp table.*

Join Multiple Table in one statement, pada *join* ini terdapat beberapa penggabungan tabel dalam satu *statement* yang sering dieksekusi pada beberapa *query* berbeda. Untuk mengeksekusi satu *query* penggabungan beberapa tabel yang rumit saja memerlukan waktu yang cukup lama, bayangkan jika penggabungan tabel-tabel tersebut sering dieksekusi pada *query* yang berbeda-beda, tentu akan memakan waktu yang lebih lama dan memboroskan *resource*. Untuk mengatasi masalah kinerja yang dihadapi, maka pada pada tahap perancangan ini, dirancang suatu alternatif solusi yang dapat meningkatkan kinerja dan menghemat *resource*, yaitu *Join Multiple Table using temp table*. Dengan mengimplementasikan solusi tersebut, penggabungan beberapa tabel yang sering dieksekusi ditempatkan dalam satu *temp table*, sehingga ketika hasil penggabungan tersebut hendak dipergunakan kembali dapat langsung mengakses ke dalam *temp table*, tidak perlu melakukan penggabungan beberapa tabel yang kompleks lagi.

Menggunakan ROWID untuk Mengakses Data

Berdasarkan tingkatan *access path* tercepat, *ROWID* merupakan cara akses tercepat, sehingga kami menggunakan *cursor* untuk menunjuk *ROWID* agar dapat digunakan untuk proses DML. Berikut contoh *query* yang menggunakan *cursor* dan *ROWID*:

```
PROCEDURE INS_FROM_LOAN IS
CURSOR INS_ACCT IS
SELECT DAD.ROWID
FROM   D_ACCOUNT_DTL DAD
WHERE  DAD.RECORD_TO_DATE  = I_INDATE
AND    NOT EXISTS (SELECT 'X' FROM P_ACCT_DTL_ENH PAD
                   WHERE  PAD.ACCT_NO       = DAD.ACCT_NO
                   AND    PAD.MONTH_IDX     = TO_CHAR(I_INDATE,'MM'));

  OPEN INS_ACCT;
  FETCH INS_ACCT BULK COLLECT INTO  OBJ_DROWID ;
  CLOSE INS_ACCT;

V_CTR_LOOP := TRUNC(OBJ_DROWID.COUNT/V_REC_COMMIT);

FOR CTR_CNT IN 1..V_CTR_LOOP LOOP
     FORALL CTR IN V_INITIAL_CNT..V_FINAL_CNT
     INSERT
     INTO   P_ACCT_DTL_ENH(
          ACCT_NO, CUST_NO, LOS_APPLN_NO, HOST_APPLN_NO, FACILITY_CD,  FACILITY_SEQ,
          LOAN_TYPE, ORIG_BAL, VINTAGE, ACCT_OPEN_DATE, LOAN_TENURE, ORIG_LOAN_DATE,
         PAYMENT_FREQUENCY, SALES_EXEC_ID, PRG_CODE, EMPLOYMENT_TYPE, OCC_CD,
         REGION_CD, BRANCH_CD, HUB_CD, DEVELOPER_CD, CHANNEL_CD, POSTAL_CD,
         TOTAL_INCOME, CREATED_USER, CREATED_DTM, AMT_DPD1,
          INT_DPD1, PROCESS_SEQ_NO, MONTH_IDX)

     SELECT ACCT_NO, CIF_NO, HOST_APPLN_NO, HOST_APPLN_NO, FACILITY_CD,  FACILITY_SEQ,
               LOAN_TYPE, ORIG_BAL, TO_NUMBER(TO_CHAR(DATE_OPENED,'RRRRMM')),
               DATE_OPENED, LOAN_TERM, ORIG_LOAN_DATE, PAYMENT_FREQUENCY,  '99',  '99',
               '99', '99', '99','99', '99', '99','99', '99', 0,UPPER(USER),  SYSDATE, 0, 0, PROCESS_SEQ_NO,
               TO_CHAR(RECORD_TO_DATE,'MM')
     FROM   D_ACCOUNT_DTL
     WHERE  ROWID = OBJ_DROWID(CTR);
```

Gambar 5 Menggunakan ROWID

Menghindari Penggunaan *Data Type Convertion*

Yang dimaksud dengan *data type convertion* adalah mengubah tipe data suatu *field*. Misalnya ingin menampilkan tanggal dalam format lain, seperti contoh berikut ini:

```
SELECT last_name, TO_CHAR(hire_date, 'DD Month YYYY')AS HIREDATE
FROM   employees;
```

Gambar 6 *Data Type Convertion*

Sehingga menghasilkan output seperti dibawah ini:

```
LAST_NAME          HIREDATE
===============    ================
John Travolta      10 January 2004
Paris Hilton       23 February 2004
Britney Spears     05 January 2004
Craig David        15 June 2004
```

Gambar 7 *Output Data Type Convertion*

Bentuk *code* di atas merupakan contoh konversi data secara eksplisit karena *keyword data type convertion* disebutkan secara langsung dalam *query*.

Pada perancangan SQL tuning dengan tujuan untuk menghasilkan *code* yang lebih optimal, penggunaan konversi data secara eksplisit diminimalisasi karena dapat menurunkan kinerja. Sehingga beberapa *code* yang menggunakan konversi data secara eksplisit diubah menjadi implisit seperti dibawah ini:

```
SELECT COUNT(*)
FROM   D_ACCOUNT_DTL DAD
WHERE  TO_DATE(DAD.RECORD_TO_DATE,'DD-MON-YY') = '14-MAY-08';
```

Gambar 8 Sebelum SQL Tuning

Menjadi:

```
SELECT COUNT(*)
FROM   D_ACCOUNT_DTL DAD
WHERE  DAD.RECORD_TO_DATE = '14-MAY-08';
```

Gambar 9 Setelah SQL Tuning

Partitioning

Tabel yang diberi partisi adalah tabel yang berisi data lebih dari 2 GB dan data yang menunjukkan waktu (misalnya: kolom *year_month* dan *month_idx*). Jenis partisi yang diimplementasikan adalah: *range*, *hash* dan *interval partition*. Jenis partisi yang diimplementasikan tersebut disesuaikan dengan jenis data yang terdapat dalam tabel-tabel milik *database funding*.

```
CREATE TABLE P_ACCT_DTL_ENH (
  ACCT_NO       NUMBER(19)      NOT NULL,
  CUST_NO       NUMBER(19)      NOT NULL,
  LOS_APPLN_NO  VARCHAR2(20)    NOT NULL,
  MONTH_IDX     VARCHAR(2)      NOT NULL
)

PARTITION BY RANGE (MONTH_IDX) SUBPARTITION BY HASH (ACCT_NO)
SUBPARTITIONS 4 (
        PARTITION P_JAN VALUES LESS THAN ('02'),
        PARTITION P_FEB VALUES LESS THAN ('03'),
        PARTITION P_MAR VALUES LESS THAN ('04'),
        PARTITION P_APR VALUES LESS THAN ('05'),
        PARTITION P_MAY VALUES LESS THAN ('06'),
        PARTITION P_JUN VALUES LESS THAN ('07'),
        PARTITION P_JUL VALUES LESS THAN ('08'),
        PARTITION P_AUG VALUES LESS THAN ('09'),
        PARTITION P_SEP VALUES LESS THAN ('10'),
        PARTITION P_OCT VALUES LESS THAN ('11'),
        PARTITION P_NOV VALUES LESS THAN ('12'),
        PARTITION P_DEC VALUES LESS THAN ('13')
);
```

Gambar 10 *Partitioning*

Indexing

Pada perancangan ini, data yang diberi *index* adalah kolom dalam suatu tabel yang mengandung banyak data dan kolom tersebut sering diakses baik dalam *select statement* maupun dalam *conditional statement (where clause)*. Contoh: kolom *record_to_date* dalam tabel D_ACCOUNT_DTL. Kolom tersebut sering diakses untuk membentuk *analytical report* harian maupun bulanan seperti: KPI, AVG TA dan P & L report.

Index yang diimplementasikan adalah B*Tree karena kolom yang diberi *index* sering menggunakan DML *statement (insert, update, delete)*, sehingga tidak cocok jika menggunakan *bitmap index*.

Berikut contoh *code indexing* pada salah satu tabel :

```
CREATE TABLE D_ACCOUNT_DTL(
  BRANCH_NO                     NUMBER(5),
  ACCT_NO                       NUMBER(19) NOT NULL,
  ACCT_TYPE                     VARCHAR2(1),
  HOST_APPLN_NO                 VARCHAR2(20),
  CIF_NO                        NUMBER(19),
  STATUS_CD                     NUMBER(1),
  LOAN_TYPE                     CHAR(10 BYTE),
  ORIG_BAL                      NUMBER(17,2),
  RELEASE_AMT                   NUMBER(17,2),
  CURR_BAL                      NUMBER(17,2),
  MISC_INSURANCE_REBATE         NUMBER(15,2),
  RECORD_TO_DATE                DATE,
)

CREATE INDEX IDX_RECORD_TO_DATE ON D_ACCOUNT_DTL(RECORD_TO_DATE);
CREATE INDEX IDX_LOS_APPLN_NO ON D_ACCOUNT_DTL(LOS_APPLN_NO);
CREATE UNIQUE INDEX PK_D_ACCOUNT_DTL ON D_ACCOUNT_DTL(ACCT_NO);
```

Gambar 11 *Indexing*

Evaluasi

Tabel 1 Evaluasi Sebelum *Tuning*

No.	Log	Date	Odate	Posting_Date_Key	Jml Record	Process Time (Menit)	Process Time (Jam)
1	[12/19/2013 09:37:34 AM] Task started at: Thursday, December 19, 2013 09:37:34	12/19/2013 09:37:34 AM	18 Juni 2013	2456462	5760773	274.75	4.58
	[12/19/2013 02:12:19 PM] Task finished at: Thursday, December 19, 2013 14:12:19	12/19/2013 02:12:19 PM					
2	[12/19/2013 03:02:30 PM] Task started at: Thursday, December 19, 2013 15:02:30	12/19/2013 03:02:30 PM	19 Juni 2013	2456463	5772319	239.73	4.00
	[12/19/2013 07:02:14 PM] Task finished at: Thursday, December 19, 2013 19:02:14	12/19/2013 07:02:14 PM					
3	[12/20/2013 09:04:02 AM] Task started at: Friday, December 20, 2013 09:04:02	12/20/2013 09:04:02 AM	20 Juni 2013	2456464	5940463	272.87	4.55
	[12/20/2013 01:36:54 PM] Task finished at: Friday, December 20, 2013 13:36:54	12/20/2013 01:36:54 PM					
4	[12/20/2013 03:29:36 PM] Task started at: Friday, December 20, 2013 15:29:36	12/20/2013 03:29:36 PM	21 Juni 2013	2456465	16517495	536.23	8.94
	[12/21/2013 12:25:50 AM] Task finished at: Saturday, December 21, 2013 00:25:50	12/21/2013 12:25:50 AM					

Proses ini adalah proses pembentukan *data mart*. Tampak terlihat kolom *process time* dalam menit dan jam, ada yang memerlukan waktu lebih dari 8 jam.

Tabel 1 Evaluasi setelah *tuning*

No.	Plan	Server	Minute	Second
1	Add Column 2	01	4	40
2	KL EXC COST	01	5	10
3	UN	01	5	35
4	KL COST FIX	01	5	50
5	KL COST%	01	6	20

Dari hasil evaluasi sebelum dan setelah *tuning*, terlihat ada pengurangan waktu yang siginifikan. Proses yang tadinya memerlukan waktu dalam jam, dapat dikurangi menjadi menit.

SIMPULAN

Untuk menghasilkan *analytical report*, Divisi Pengembanan Produk dan Jasa membentuk data histori dari data posisi setiap harinya dengan menggunakan *procedure* yang belum optimal sehingga membuat proses pembentukan *analytical report* memakan waktu yang lama.

Setelah menganalisis permasalahan yang menyebabkan lamanya pembentukan *analytical report* pada *database funding*, solusi yang diimplementasikan adalah Model Aplikasi *Tuning* yang terdiri atas tiga solusi, yaitu: SQL Tuning *procedure* yang membentuk data histori, *indexing* kolom yang sering diakses pada sebuah tabel serta *partitioning* tabel-tabel yang mengandung data di atas 2 GB dan kolom yang mengandung data waktu.

Hasil dari *tuning* yang diimplementasikan adalah meningkatnya kecepatan dalam pembentukan table fakta sehingga pada akhirnya *analytical report* data funding dapat dihasilkan secara lebih cepat dan sesuai dengan kebutuhan pengguna, Business Object dipergunakan kembali sebagai *reporting tools* sehingga *analytical report* dapat terbentuk tanpa membutuhkan kemampuan pengguna untuk *query* langsung ke *data mart.*

DAFTAR PUSTAKA

Anonim1. (2007). *Partioning in Oracle Database 11g.* USA: Oracle Corporation.

Cecilia, C., Mihai, G. (2011). Increasing Database Performance using Indexes. *Database Systems Journal, II*(2):13.

Chan & Immanuel. (2008). *2 days + Performance Tuning Guide 10g Release 2 (10.2).* USA: Oracle Corporation.

Connolly, T., Begg, C. (2010). *Database System; a Practical Approach to design, Implementation and Management* (5 ed.). England: Pearson Education.

Hitesh, K. S., Aditya, S., Ranjit, B. (2012). A Framework for Automated Database TuningUsing Dynamic SGA Parameters and Basic Operating System Utilities. *Database Systems Journal, III*(4): 25.

Indrajani. (2011). *Bedah Kilat 1 Jam – Pengantar dan Sistem Basis Data.* Jakarta, Jakarta, Indonesia: Elex Media Computindo.

_____. (2011). *Perancangan Basis Data Dalam All In 1.* Jakarta, Jakarta, Indonesia: Elex Media Computindo.

_____. (2014). Rancang Bangun Model Aplikasi Tuning Pada Unit Bisnis Kredit Konsumer. In C. L. Bandung (Ed.). Bandung: JBPTITBPP.

Indrajani, Wihendro, & Safan. (2013). Rancang Bangun Konseptual Basis Data Klinik 24 Jam. *Seminar Nasional Teknologi Informasi & Multimedia 2013.* Yogyakarta: STMIK AMIKOM.

Millsap & Garry. (2008). *Optimizing Oracle Performance: A Practitioner's Guide to Optimizing Response Time.* Cambridge: O'Reilly.