ENKRIPSI DATA DAN TABLESPACE PADA ORACLE DATABASE

Iwan Tanto

Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI
Jl. Ir. Juanda 96 Bandung 40132

E-mail : iwantanto@gmail.com

ABSTRAK

Data yang tersimpan di dalam *database* perusahaan mengandung informasi sensitif yang perlu dijaga keamanannya. Sistem keamanan *database* di mesin *server* pada umumnya sudah dikelola dengan baik, namun proses *backup* data seringkali merupakan titik rawan kebocoran data karena data disalin ke media yang kemudian disimpan di luar sistem keamanan *server*. Pengamanan data *backup* ini dapat dilakukan dengan cara melakukan enkripsi pada saat *backup*. Teknologi *Transparent Data Encryption* (TDE) pada *Oracle Database* memudahkan proses enkripsi tersebut.

Kata kunci: enkripsi, *tablespace*

1. PENDAHULUAN

Data yang tersimpan di dalam suatu *database* perusahaan seringkali mengandung informasi yang penting dan sensitif sehingga perlu dijaga keamanannya. Data sensitif bisa berupa nomor rekening nasabah, nomor PIN, nomor kartu kredit dan data nasabah lainnya, atau bisa juga berupa daftar gaji pegawai, hasil riset perusahaan, data keuangan perusahaan dan masih banyak lagi tergantung pada persepsi setiap perusahaan terhadap datanya. Keamanan data ini menjadi penting karena di era informasi ini seluruh data disimpan di dalam sebuah *database* di mesin *server* yang melayani seluruh pengguna di perusahaan tersebut. Bahkan perusahaan multinasional sudah menghubungkan mesin *server*-nya dengan internet sehingga *database* perusahaan dapat diakses oleh karyawannya dari mana saja.

Sistem keamanan *database* di mesin *server* pada umumnya sudah dikelola dengan baik sehingga hanya pihak berwenang saja yang dapat mengakses data

tersebut. Dengan sistem *login* yang dikelola dengan baik, dapat diatur pengguna (*user*) mana saja yang berhak mengakses data sensitif, memonitor kapan data sensitif tersebut diakses oleh pengguna tertentu, dan mengenkripsi data sensitif yang dikirimkan via jalur terbuka seperti internet. Namun proses *backup* data seringkali merupakan titik rawan kebocoran data karena data disalin ke media yang kemudian disimpan di luar sistem keamanan *server*. Salinan data *backup* ini seringkali luput dari pengamanan yang mencukupi sehingga lebih mudah dicuri atau disalin lagi secara ilegal. Salinan ilegal ini kemudian disalahgunakan dengan cara me-*restore* datanya di mesin lain.

Pencegahan kejahatan data *backup* semacam ini dapat dilakukan dengan cara melakukan enkripsi pada saat *backup*, walaupun hal ini seringkali diabaikan oleh perusahaan. Data *backup* yang berhasil dicuri dan di-*restore* tidak dapat dibaca sebelum melakukan proses dekripsi. Dengan teknologi enkripsi yang sudah tersedia saat ini, para pencuri data tidak mungkin mendekripsi data tersebut dengan mudah.

2. *TRANSPARENT DATA ENCRYPTION (TDE)*

Oracle memperkenalkan teknologi *Transparent Data Encryption* (TDE) dalam *Oracle Advanced Security* mulai *Oracle Database* 10g *release* 2. TDE adalah sebuah konsep dimana proses enkripsi dan dekripsi data berjalan secara transparan, artinya pengguna tidak perlu menulis kode atau terlibat dalam proses enkripsi. Pengguna dapat menggunakan *database* sama seperti *database* tanpa enkripsi, proses enkripsi berjalan di *background.*

Pada *Oracle Database* 10g *release* 2 diperkenalkan TDE untuk enkripsi kolom pada suatu tabel. Enkripsi kolom memiliki beberapa aturan yang harus diikuti agar tidak berdampak buruk terhadap performansi aplikasi. Enkripsi kolom tidak dianjurkan jika kolom tersebut digunakan dalam *query* yang bersifat mencari rentang nilai, misalnya kolom gaji karyawan. Proses enkripsi akan mengakibatkan nilai data dalam kolom tersebut menjadi acak. Indeks yang dibangun dari kolom terenkripsi juga akan memiliki nilai terenkripsi. Sehingga *query* seperti mencari karyawan yang gajinya 1 juta – 2 juta akan tidak efektif dan sulit. Enkripsi kolom juga tidak dianjurkan untuk kolom yang dipakai sebagai *foreign key* karena proses enkripsi dapat menyebabkan hubungan tersebut terputus. Walaupun enkripsi kolom bekerja sangat cepat dan efektif, keterbatasan pada *query* rentang nilai adalah tantangan utama dalam penerapannya.

3. ENKRIPSI *TABLESPACE*

Opsi *Oracle Adanced Security* pada *Oracle Database* 11g memperkenalkan TDE untuk enkripsi *tablespace* yang memungkinkan seluruh *tablespace* dienkripsi secara utuh. Dengan demikian teknologi ini telah menjawab kelemahan enkripsi kolom. Sebuah *tablespace* dibuat secara terenkripsi, kemudian seluruh data baik tabel maupun indeks disimpan terenkripsi di dalam *tablespace* tersebut. Ketika pengguna mengakses sebuah data, proses *server* memindahkan data tersebut dari media penyimpanan ke *buffer cache* di dalam *session* pengguna tersebut. Data didekripsi pada saat proses pemindahan, sehingga data di *buffer cache* selalu merupakan data yang tidak terenkripsi. Seluruh jenis pengaksesan data, seperti *query*, membuat dan mencari indeks, join tabel, selalu berlangsung di *buffer cache*. Dengan demikian performansi operasi data yang melibatkan *tablespace* terenkripsi adalah sama dengan *tablespace* yang tidak terenkripsi.

Dengan menggunakan enkripsi *tablespace*, sebelum data di dalam *buffer* ditulis kembali ke media penyimpanan, dilakukan enkripsi oleh proses DB *Writer* (DBWn). Operasi lain yang memanipulasi data langsung pada *database* akan dienkripsi secara langsung saat memproses operasi tersebut. Penulisan catatan proses (*log*) juga dilakukan secara terenkripsi.

Enkripsi membutuhkan sedikitnya dua hal, kunci enkripsi dan algoritma enkripsi. TDE menggunakan arsitektur kunci dua-tingkat (*two-tier key*). Kunci enkripsi untuk kolom maupun *tablespace* disimpan di dalam *database* namun dienkripsi dengan menggunakan kunci lain yang disebut kunci utama (*masterkey*). Kunci utama disimpan di luar *database* dalam sebuah wadah khusus yang disebut modul keamanan eksternal (*external security module*) yang dapat berupa *software* seperti *Oracle wallet* atau *hardware* yang dirancang khusus.

Oracle wallet adalah sebuah dokumen yang mengikuti format sesuai standar *Public Key Cryptography Standard* No.12 dan dienkripsi dengan *password*. Untuk menggunakan *wallet* sebagai modul keamanan eksternal, sebuah *password* harus dimasukkan agar kunci utama (*masterkey*) dapat mengakses *database*. Jika *password* salah, maka *wallet* tidak dapat dibuka dan data yang terenkripsi tidak dapat dibaca. *Wallet* akan tertutup otomatis ketika *instance database* dimatikan, yaitu ketika pengguna keluar (*logout*) dari *database* atau *server database* dimatikan. Jika pencuri

data berhasil me-*restore* data *backup*, maka tanpa *wallet* dan *password* yang tepat sang pencuri tidak dapat membaca isi data yang terenkripsi tersebut.

4. **PENGATURAN ENKRIPSI *TABLESPACE***

Berikut ini adalah cara melakukan pengaturan enkripsi *tablespace* pada *Oracle Database* 11 *release* 1 atau lebih tinggi. Pertama dibuat terlebih dahulu sebuah *wallet* dengan cara berikut:

1. Atur variabel ORACLE_BASE dengan menggunakan perintah
   $export ORACLE_BASE=/opt/oracle
2. Pindah ke direktori ORACLE_BASE kemudian ke subdirektori admin dari *instance* yang digunakan, misalnya prolin1
   $cd $ORACLE_BASE/admin/prolin1
3. Buat direktori bernama “wallet” untuk menyimpan *wallet*.
   $mkdir wallet
4. Buat *wallet* dengan *password* pengaman, misalnya “T45rustMe54”. Ingat *password* bersifat *case-sensitive*.
   $ sqlplus/as sysdba
   SQL>alter system set encryption key identified by “T45rustMe54”;

Wallet hanya perlu dibuat satu kali saja. Setelah *database* terbuka, *wallet* akan tetap terbuka hingga *wallet* ditutup secara sengaja atau *database* dimatikan yang akan menutup *wallet* secara otomatis. Untuk membuka *wallet* setelah *database* dinyalakan, gunakan perintah:

SQL>alter system set wallet open identified by “T45rustMe54”;
System altered.

Setelah *wallet* selesai dibuat, maka *tablespace* terenkripsi dapat dibuat.

1. Perintah berikut membuat *tablespace* terenkripsi dengan nama enc128_ts:
   Create tablespace enc128_ts
   Datafile ‘/u01/oracle/database/enc128_ts.dbf’
   Size 1M autoextend on next 1M
   Encryption using ‘AES128’
   Default storage(encrypt)
   /

Perintah Encryption using ‘AES128’ menunjukkan proses enkripsi menggunakan algoritma AES dengan kunci *default* selebar 128-bit. Dapat dipilih AES192 untuk memilih kunci selebar 192-bit atau AES256 untuk kunci selebar 256-bit.

2. Setelah *tablespace* dibuat, objek di dalamnya dapat dibuat. Perintah berikut ini membuat sebuah tabel bernama ACCOUNTS_ENC:

```
Create table accounts_enc(
    ACC_NO      NUMBER              NOT NULL,
    FIRST_NAME      VARCHAR2(30)    NOT NULL,
    ....
    )
    Tablespace enc128_ts;
```

Tidak ada perintah khusus untuk membuat objek seperti tabel di atas. Seluruh kolom dari tabel, dan juga seluruh objek yang dibuat di dalam *tablespace* tersebut akan terenkripsi.

Untuk memastikan telah terjadi enkripsi, cobalah untuk memasukkan sebuah data ‘David’ untuk kolom FIRST_NAME kemudian mencari kembali data tersebut di datafile yang terdapat di *tablespace* enc128_ts.

```
SQL>insert into accounts_enc values(1,’David,...);
$string enc128_ts.dbf|grep David
```

Perintah untuk mencari data ‘David’ tidak akan memberikan hasil karena data ‘David’ telah tersimpan di dalam *tablespace* yang terenkripsi.

5. PERFORMANSI ENKRIPSI *TABLESPACE*

Masalah utama dalam enkripsi data adalah kekhawatiran terjadinya penurunan performansi. Melakukan *query* terhadap kolom yang terenkripsi menimbulkan masalah karena indeks mungkin tidak dapat digunakan. Jika indeks tidak dapat dipakai maka performansi akan turun. Dengan konsep enkripsi *tablespace* dimana isi *buffer cache* adalah teks biasa tanpa enkripsi, maka dapat dipastikan performansi tidak akan turun.

Untuk memastikan hal ini dapat dilakukan pengujian kecil. Buatlah sebuah *tablespace* normal:

```
Create tablespace normal_ts
```

Datafile '/u01/oracle/database/normal_ts.dbf'

Size 1M autoextend on next 1M;

Kemudian buat ACCOUNTS_REG yang identik dengan ACCOUNTS_ENC.

Create table accounts_reg(

ACC_NO NUMBER NOT NULL,

FIRST_NAME VARCHAR2(30) NOT NULL,

....

)

Tablespace normal_ts;

Selanjutnya isi kedua tabel tersebut dengan data yang sama. Lalu buat indeks untuk kolom FIRST_NAME di masing-masing tabel. Untuk setiap tabel, lakukan *query* untuk mencari semua nama yang dimulai dari huruf D. *Query* ini akan menggunakan indeks pada kolom FIRST_NAME. Bandingkan hasil eksekusi *query* tersebut, tampak tidak ada perbedaan performansi antara *tablespace* terenkripsi dan *tablespace* normal (Listing 1).

Listing 1 – Pengujian performansi *tablespace* normal dan terenkripsi.

```
/* Run a query on the account_reg (unencrypted tablespace) */

SQL>set autot on explain stat
SQL>set timing on
SQL>select first_name from accounts_reg
 2   where first_name like 'D%';

... hasil query ...

Elapsed: 00:05:36.38
-------------------------------------------------------------------------------
---
| Id | Operation         | Name           | Rows  | Bytes  | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------
---
|  0 | SELECT STATEMENT  |                |  210K |  1442K |  513     (1)|
00:00:07 |
|* 1 | INDEX RANGE SCAN  | IN_ACC_REG_FN  |  210K |  1442K |  513     (1)|
00:00:07 |
-------------------------------------------------------------------------------
---

Predicate Information (identified by operation id):
---------------------------------------------------
1 – access("FIRST_NAME" LIKE 'D%')
    filter("FIRST_NAME" LIKE 'D%')

Statistics
----------------------
   1 recursive calls
   0 db block gets
3458 consistent gets
 127 physical reads
```

```
...

/* Run a query on the account_enc (encrypted tablespace) */

SQL>select first_name from accounts_enc
 2   where first_name like 'D%';

... hasil query ...

Elapsed: 00:05:33.85
-------------------------------------------------------------------------------
---
| Id | Operation         | Name           | Rows  | Bytes  | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------
---
|  0 | SELECT STATEMENT |                  |  210K |  1442K |  513     (1)|
00:00:07 |
|* 1 | INDEX RANGE SCAN | IN_ACC_ENC_FN |  210K |  1442K |  513     (1)|
00:00:07 |
-------------------------------------------------------------------------------
---

Predicate Information (identified by operation id):
---------------------------------------------------
1 – access("FIRST_NAME" LIKE 'D%')
     filter("FIRST_NAME" LIKE 'D%')

Statistics
----------------------
   1 recursive calls
   0 db block gets
3427 consistent gets
 127 physical reads
...
```

6. PERBANDINGAN ENKRIPSI *TABLESPACE* DAN ENKRIPSI KOLOM

TDE level kolom memungkinkan *user* untuk mengenkripsi data pada kolom tertentu saja. Berikut ini adalah cara mengenkripsi kolom FIRST_NAME pada tabel ACCOUNTS_REG_ENC yang dibuat dari tabel ACCOUNTS_REG:

Create table accounts_reg_enc as

Select * from accounts_reg;

Alter table accounts_reg_enc

Modify first_name encrypt using 'AES128';

Setelah modifikasi ini, kolom FIRST_NAME akan disimpan sebagai data terenkripsi di dalam tabel ACCOUNTS_REG_ENC, sedangkan kolom lainnya disimpan sebagai teks biasa. Pada saat tabel ini dimuat ke dalam *buffer cache*, kolom FIRST_NAME tetap terenkripsi. Ini menyebabkan penurunan performansi jika menggunakan indeks yang mengacu ke kolom FIRST_NAME.

Listing 2 – Pengujian performansi enkripsi kolom dan enkripsi *tablespace*

```
/* Run a query on the account_reg_enc (unencrypted tablespace but */
/* with the encrypted first_name column)                          */

SQL>set autot on explain stat
SQL>select count(1) from accounts_reg_enc
 2   where first_name like 'D%';

... hasil query ...

-------------------------------------------------------------------------------
---
| Id | Operation            | Name          | Rows  | Bytes | Cost(%CPU)| Time
|
-------------------------------------------------------------------------------
---
|  0 | SELECT STATEMENT     |               |     1 |     7 |  686    (5)|
00:00:09 |
|  1 | SORT AGGREGATE       |               |     1 |     7 |            |
|
|* 2 | INDEX FAST FULL SCAN|IN_ACC_REG_FN | 50000 |  341K |  686    (5)|
00:00:09 |
-------------------------------------------------------------------------------
---

Predicate Information (identified by operation id):
---------------------------------------------------
2 – filter(INTERNAL_FUNCTION("FIRST_NAME") LIKE 'D%')

Statistics
----------------------
    0 recursive calls
    0 db block gets
13963 consistent gets

/* Run a query on the account_enc (encrypted tablespace) */

SQL>select count(1) from accounts_enc
 2   where first_name like 'D%';

... hasil query ...

-------------------------------------------------------------------------------
---
| Id | Operation            | Name          | Rows  | Bytes | Cost(%CPU)| Time
|
-------------------------------------------------------------------------------
---
|  0 | SELECT STATEMENT     |               |     1 |     7 |  513    (1)|
00:00:07 |
|  1 | SORT AGGREGATE       |               |     1 |     7 |            |
|
|* 2 | INDEX RANGE SCAN     |IN_ACC_ENC_FN |  210K | 1442K |  513    (1)|
00:00:07 |
-------------------------------------------------------------------------------
---

Predicate Information (identified by operation id):
---------------------------------------------------
2 – access("FIRST_NAME" LIKE 'D%')
    filter("FIRST_NAME" LIKE 'D%')

Statistics
----------------------
   0 recursive calls
   0 db block gets
 120 consistent gets
...
```

Selanjutnya dapat dilakukan pengujian kecil untuk melihat seberapa jauh pengaruh enkripsi kolom terhadap penurunan performansi. Untuk itu lakukan hal

yang sama yaitu melakukan *query* untuk mencari semua nama yang dimulai dari huruf D pada tabel ACCOUNTS_REG_ENC yang dibuat dari tabel ACCOUNTS_ENC. Perhatikan bahwa enkripsi kolom telah menghambat Oracle untuk memanfaatkan indeks, sehingga proses *query* berjalan lambat (*Listing* 2).

7. KESIMPULAN

Secara umum enkripsi menyelesaikan masalah keamanan, namun menimbulkan masalah baru yaitu penurunan performansi. Penurunan performansi seringkali tidak dapat diterima dalam situasi nyata, sehingga banyak perusahaan yang memilih untuk mengorbankan keamanan demi performansi yang baik. Dengan menggunakan *transparent tablespace encryption*, penurunan performansi bukan menjadi masalah lagi. Performansi *tablespace* yang terenkripsi sama dengan performansi *tablespace* yang tidak terenkripsi. Keuntungan lain adalah proses enkripsi berlangsung transparan tanpa membutuhkan penulisan program.

8. DAFTAR PUSTAKA

[1]. Oracle Magazine, September/October 2005, *Transparent Data Encryption*, otn.oracle.com/oramag/oracle/05-sep/o55security.html

[2]. Oracle Magazine, January/February 2009, *Encrypting Tablespaces*, otn.oracle.com/oramag/oracle/09-jan/o19tte.html

[3]. *Oracle Database Advanced Security Administrator's Guide*, download.oracle.com/docs/cd/B28359_01/network.111/b28530/asotrans.htm #CJAFEAII *Using Transparent Database Encryption in Oracle Database 11g*, otn.oracle.com/obe/11gr1_db/security/tde/tde.htm