TRANSLATOR PL/SQL PADA ORACLE KE TRANSACT-SQL PADA MICROSOFT SQL SERVER TRANSLATOR OF PL/SQL IN ORACLE TO TRANSACT-SQL IN MICROSOFT SQL SERVER

M. Arief Faizal R.[1], -[2]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Beragamnya produk RDBMS (Relational DataBase Management System) memungkinkan bagi pengguna untuk memilih dan menggunakannya sesuai dengan keinginan, bahkan mungkin sesuai dengan kebutuhannya. Produk-produk ini juga tentu memiliki karakteristik yang berbeda dengan kelebihan dan kekurangannya masing-masing. Salah satunya adalah bahasa prosedural yang digunakan.

Perbedaan bahasa prosedural inilah yang dijadikan alasan untuk membuat suatu sistem translator yang akan menerjemahkan suatu bahasa prosedural tertentu ke bahasa prosedural yang lain. Yakni dari PL/SQL pada Oracle ke T-SQL pada Microsoft SQL Server. Translator ini diharapkan akan memudahkan pengguna produk RDBMS untuk mempelajari dan membandingkan bahasa-bahasa prosedural yang berlainan.

Sistem translator ini bisa menangani semua elemen PL/SQL (kecuali yang tidak memiliki padanannya pada T-SQL), tipe data, built-in function, DML (Data Manipulation Language) dan Transaction Control. Di sisi lain, sistem ini tidak menangani analisis semantik PL/SQL, dengan batasan bahasa masukan bebas dari komentar serta batas penggunaan fungsi dan prosedur bersarang tidak lebih dari dua. Adapun pengujian dilakukan pada kedua RDBMS dengan menggunakan SQLPlus pada Oracle dan SQL Query Analyzer pada Microsoft SQL Server secara manual. Dari pengujian ini akan diketahui kebenaran sintaks dan semantik antara kedua bahasa.

Kata Kunci : RDBMS, PL/SQL, T-SQL, translator

Abstract

There are so many RDBMS (Rational DataBase Management System) product that enable user to choose and use them, indeed as their needs. These products have difference characteristics with their advantage or weakness. One of the differences is procedural language that is used.

This differences will be the reason why the translator system is built. The system will translate one procedural language to another. That is, from PL/SQL in Oracle to T-SQL in Microsoft SQL Server. With hope, by using this tool, user will learn and compare both various procedural language easily.

This translator can handle all elements of PL/SQL (except element which don't have comparison in T-SQL), datatype, built-in function, DML (Data Manipulation Language) and Transaction Control. In the other side, this system can't handle semantic analysis of PL/SQL, with restriction, input language should be free from comment and also using limitation of nested function or procedure not more than two. In testing concern, both language will be tested in both of RDBMS by using SQLPlus at Oracle and SQL Query Analyzer at Microsoft SQL Server manually. From this testing, we will know the valid syntax and semantic analysis for both of language.

Keywords : RDBMS, PL/SQL, T-SQL, translator

BAB I
PENDAHULUAN

1.1 Latar Belakang

RDBMS (***Relational DataBase Management System***) merupakan alat bantu utama dalam pengelolaan data. Saat ini, telah tersedia banyak produk RDBMS yang memungkinkan kita untuk memilih dan menggunakannya sesuai dengan keinginan. Produk RDBMS yang populer digunakan, antara lain : *Oracle, Microsoft Access, Microsoft SQL Server* dan *MySQL*. Produk-produk ini memiliki spesifikasi kelebihan dan kekurangan tersendiri.

Karena hal itulah, banyak pengguna yang hanya menggunakan produk RDBMS tertentu dan jarang memperhatikan produk yang lain. Yang paling mendasar dari produk-produk tersebut adalah bahasa prosedural SQL yang digunakan. Banyak di antara kita yang mengetahui **PL/SQL** (pada Oracle) secara mendalam, tetapi tidak pada **T-SQL** (pada Microsoft SQL Server), atau sebaliknya. PL/SQL merupakan bahasa query prosedural yang digunakan untuk mengakses basisdata Oracle. Sedangkan T-SQL (Transact-SQL) merupakan bahasa yang digunakan untuk mengelola basisdata SQL Server. Dari segi sintaks bahasa yang digunakan, bahasa ini memiliki beberapa kesamaan dan juga perbedaan.

Alasan inilah yang dijadikan pertimbangan untuk membuat suatu translator (penerjemah) antar-bahasa prosedural SQL yang berbeda. Yang nantinya diharapkan, pengguna bisa mempelajari bahasa prosedural SQL yang beragam (khususnya PL/SQL dan T-SQL).

1.2 Perumusan Masalah

Pada tugas akhir ini, penulis akan menitikberatkan bagaimana menerjemahkan bahasa ekstensi SQL yang berbeda, yakni dari PL/SQL pada Oracle ke T-SQL pada Microsoft SQL Server. Untuk Sintaks yang sama (antara keduanya), PL/SQL akan diterjemahkan secara langsung ke T-SQL (*mapping*),

sedangkan yang tidak sama, PL/SQL akan digenerasi berdasarkan T-SQL yang bersesuaian.

1.3 Pembatasan Masalah

Batasan masalah untuk tugas akhir ini adalah sebagai berikut :

- RDBMS yang digunakan adalah Oracle 8i dan Microsoft SQL Server 2000
- Translator ini hanya bersifat satu arah, yakni penerjemahan dari PL/SQL pada Oracle ke T-SQL pada Microsoft SQL Server.
- Hanya terbatas pada basisdata relasional.
- Pengecekan input sintaks (PL/SQL) dilakukan oleh sistem tanpa memperhatikan analisis semantiknya.
- Sintaks PL/SQL yang akan dimasukkan harus bebas dari *comment* (komentar).
- *Nested Function* dan *Nested Procedure* yang ditangani maksimal dua.

1.4 Tujuan Pembahasan

Tujuan yang ingin dicapai dalam tugas akhir ini adalah :

- Memudahkan pengguna untuk mempelajari T-SQL melalui PL/SQL sebagai pembanding.
- Menerjemahkan bahasa prosedural SQL yang berbeda, yakni dari PL/SQL pada Oracle ke T-SQL pada Microsoft SQL Server.

1.5 Metode Penyelesaian Masalah

Metode yang digunakan dalam penyelesaian tugas akhir ini adalah :

- Studi literatur tentang RDBMS dan dua produk yang digunakan pada tugas akhir ini.
- Studi literatur tentang teknik dan metode kompilasi.
- Analisis skema atau struktur bahasa prosedural yang terdapat pada RDBMS yang digunakan dalam translator ini.

- Perancangan perangkat lunak translator PL/SQL pada Oracle ke T-SQL pada Microsoft SQL Server.
- Menguji perangkat lunak dengan cara membandingkan hasil (output) basisdata menggunakan PL/SQL dengan hasil basisdata menggunakan T-SQL yang merupakan terjemahan dari PL/SQL.

1.6 Sistematika Penulisan

Tugas Akhir ini disusun dengan sistematika pembahasan sebagai berikut :

BAB I **PENDAHULUAN**

Bab ini memaparkan latar belakang dilakukannya penelitian, perumusan masalah yang akan dibahas, pembatasan masalah, tujuan yang ingin dicapai melalui penelitian ini, metode penyelesaian masalah dan sistematika pembahasan.

BAB II **DASAR TEORI**

Bab ini membahas teori yang mendukung penyusunan tugas akhir ini yaitu mengenai *basisdata relasional, karakteristik dan perbandingan PL/SQL dan T-SQL,* serta *teknik dan metode kompilasi* yang digunakan untuk menerjemahkan PL/SQL ke T-SQL.

BAB III **PERANCANGAN SISTEM**

Bab ini menguraikan mengenai proses perancangan sistem, spesifikasi kebutuhan sistem, dan perancangan subsistem.

BAB IV **IMPLEMENTASI DAN ANALISIS HASIL**

Bab ini menyajikan implementasi perangkat lunak, skenario pengujian dan analisis hasil penerjemahan yang dilakukan.

BAB V **KESIMPULAN DAN SARAN**

Berisi kesimpulan dari hasil penelitian tugas akhir ini serta saran-saran untuk pengembangan lebih lanjut.

BAB IV
IMPLEMENTASI DAN ANALISIS HASIL PERANGKAT LUNAK

4.1 Lingkungan Implementasi

Pembahasan pada bagian ini meliputi spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam pengembangan sistem ini.

4.1.1 Spesifikasi Perangkat Lunak

Sistem ini dikembangkan pada lingkungan sistem operasi Windows XP Professional, yang diperlukan untuk instalasi Borland Delphi 7.0, RDBMS Oracle 8i (8.1.5) dan Microsoft SQL Server 2000.

4.1.2 Spesifikasi Perangkat Keras

Adapun perangkat keras yang digunakan adalah :

- Processor Intel Pentium III 870 MHz
- SDRAM 256 MB
- Harddisk 20 GB
- VGA Card 32 MB
- Keyboard dan Mouse

4.2 Implementasi Perangkat Lunak

Implementasi perangkat lunak yang dihasilkan adalah sebagai berikut :

a. Masukan *user* berupa struktur atau blok PL/SQL, yakni berupa file teks (.txt). Alasan pemilihan jenis file ini karena sederhana, mudah dibaca atau ditulis serta formatnya fleksibel.
b. Pengecekan sintaks PL/SQL (masukan *user*) dilakukan oleh sistem dan hanya terbatas pada pengecekan sintaktik, tanpa memperhatikan pengecekan semantik dari PL/SQL tersebut.

c. Untuk konversi tipe data, digunakan standar tipe data yang ada, di mana jika tidak terdapat padanan untuk suatu tipe data tertentu, maka tipe data tersebut akan dikonversi ke tipe data yang sesuai dengan modifikasi. Sebagai contoh tipe data pada PL/SQL seperti : *boolean, number, double precision, binary_integer, natural, naturaln, pls_integer, positive, positiven, signtype, long, string, raw, long raw, bfile, lob, clob, nchar, nlob, nclob, blob, rowid,* dan *urowid* akan dikonversi ke dalam bentuk tipe data yang sesuai pada T-SQL (*lampiran b*). Tetapi untuk tipe data yang tidak memiliki padanannya tidak diikutsertakan dalam sistem ini. Contoh tipe data yang tidak memiliki padanannya adalah : *mlslabel, record, varray* dan *ref object type* atau *ref cursor*.
d. Statement SQL yang diperbolehkan dalam struktur PL/SQL hanya statement DML (*Data Manipulation Language*) dan *Transaction Control*. Jadi statement DML yang diikutsertakan dalam sistem adalah *SELECT, INSERT, UPDATE, DELETE* dan *SET TRANSACTION*. Sedangkan Transaction control berupa *COMMIT, ROLLBACK* dan *SAVEPOINT*. Untuk DDL (*Data Definition Language*) tidak diperbolehkan dalam struktur PL/SQL. **[URM97]**
e. Built-in package PL/SQL yang ditangani dalam sistem ini hanya DBMS_OUTPUT.PUT_LINE. Sedangkan untuk built-in package yang lain tidak ditangani dalam sistem ini.
f. Built-in function yang diperbolehkan dalam blok PL/SQL adalah *number function, character function returning character values, character function returning number values, date function, conversion function, miscellaneous single row function* dan *object reference function*. Sedangkan *grouping function* tidak diperbolehkan dalam blok PL/SQL, kecuali dalam struktur *SELECT statement*. (*lampiran c*).
g. Terdapat beberapa hal yang perlu diperhatikan oleh *user* bahwa SQL maupun PL/SQL memiliki *keyword* yang memungkinkan dijadikan

identifier dalam membuat sintaks PL/SQL, yakni *name* dan *c*. Kedua *keyword* ini diusahakan dihindari penggunaannya sebagai identifier.

h. Konversi struktur *JAVA* dan *C* pada *CREATE OR REPLACE PROCEDURE* atau pada *CREATE OR REPLACE FUNCTION* tidak ditangani.

i. Batasan jumlah *procedure bersarang* (*nested procedure*) dan *function bersarang* (*nested function*) yang bisa ditangani oleh sistem ini maksimal dua. Hal ini dikarenakan, nested function atau nested procedure memiliki struktur yang kompleks dalam ruang lingkup PL/SQL yang intensitas pemakaiannya relatif jarang digunakan. Sistem ini merupakan alat pembelajaran bahasa T-SQL dengan PL/SQL sebagai pembanding. *User* cenderung menggunakan dan mempelajari *simple code* T-SQL, dibandingkan mempelajari *complex code*. Karena alasan inilah, sistem hanya membatasi penggunaan nested function dan nested procedure maksimal dua.

j. Pada sistem ini, *comment* (komentar) yang mungkin ada dalam sintaks (bahasa sumber yakni PL/SQL) tidak ditangani. Dengan kata lain, bahasa masukan sudah bebas dari *comment*.

4.3 Skenario Pengujian Perangkat Lunak

Pengujian perangkat lunak dilakukan secara manual ke masing-masing RDBMS. Sebagai inisialisasi awal, untuk setiap RDBMS (Oracle dan Microsoft SQL Server 2000) dibuat basisdata dengan struktur seperti pada *lampiran f* **[URM97]**. Masukan yang berupa sintaks PL/SQL sebelum diterjemahkan menjadi T-SQL diuji di Oracle (menggunakan SQL Plus) untuk melihat hasil eksekusinya. Dengan perangkat lunak ini PL/SQL tersebut diterjemahkan menjadi T-SQL yang kemudian akan diuji di Microsoft SQL Server 2000 (menggunakan SQL Query Analyzer). Dari sini bisa dilihat perbandingan hasil keluaran di Oracle dan Microsoft SQL Server 2000.

4.4 Analisis Hasil Pengujian Perangkat Lunak

Dalam analisis hasil perangkat lunak ini dilakukan analisis penerjemahan dan kesesuaian antara PL/SQL sebagai bahasa sumber dengan T-SQL sebagai bahasa sasaran dengan menggunakan contoh (studi kasus). Studi kasus yang dianalisis merupakan studi kasus yang dipilih untuk mewakili semua elemen pada PL/SQL.

a. Blok PL/SQL

PL/SQL :

```
DECLARE
      v_Num1            NUMBER := 1;
      v_Num2            NUMBER := 2;
      v_String1   VARCHAR2(50) := 'Hello World';
      v_String2   VARCHAR2(50) :=
           'This message brought to you by PL/SQL ';
      v_OutputStr VARCHAR2(50);
BEGIN
      INSERT INTO temp_table (num_col, char_col)
         VALUES (v_Num1, v_String1);
      INSERT INTO temp_table (num_col, char_col)
         VALUES (v_Num2, v_String2);

      SELECT char_col
         INTO v_OutputStr
      FROM temp_table
      WHERE num_col = v_Num1;

      DBMS_OUTPUT.PUT_LINE(v_OutputStr);

      SELECT char_col
         INTO v_OutputStr
      FROM temp_table
      WHERE num_col = v_Num2;

      DBMS_OUTPUT.PUT_LINE(v_OutputStr);
END;
```

T-SQL :

```
DECLARE @v_num1 NUMERIC
SET @v_num1 = 1
DECLARE @v_num2 NUMERIC
SET @v_num2 = 2
DECLARE @v_string1 VARCHAR(50)
SET @v_string1 = 'Hello World'
DECLARE @v_string2 VARCHAR(50)
SET @v_string2 =
      'This message brought to you by PL/SQL '
DECLARE @v_outputstr VARCHAR(50)
```

```
BEGIN
      INSERT INTO temp_table (num_col,char_col)
         VALUES(@v_num1,@v_string1)
      INSERT INTO temp_table (num_col,char_col)
         VALUES(@v_num2,@v_string2)

      SELECT @v_outputstr = char_col
      FROM temp_table
      WHERE num_col = @v_num1
      DECLARE @PrintVariable VARCHAR(500)

      SELECT @PrintVariable =
           CONVERT(VARCHAR(500),@v_outputstr)
      PRINT @PrintVariable

      SELECT @v_outputstr = char_col
      FROM temp_table
      WHERE num_col = @v_num2

      SELECT @PrintVariable =
           CONVERT(VARCHAR(500),@v_outputstr)
      PRINT @PrintVariable
END
```

Pengujian pada *Oracle (dengan SQLPlus):*

Untuk inisialisasi :

```
SQL> SET SERVEROUTPUT ON;
SQL> SELECT * FROM temp_table;
```

Output :

```
no rows selected
```

Ketika blok PL/SQL tersebut dieksekusi maka (akan muncul) :

```
Hello World
This message brought to you by PL/SQL

SQL> SELECT * FROM temp_table;
```

Output :

```
NUM_COL CHAR_COL
------- ------------------------------------------------
      1 Hello World
      2 This message brought to you by PL/SQL
```

Pengujian pada *Microsoft SQL Server (dengan SQL Query Analyzer) :*

Untuk inisialisasi :

```
SELECT * FROM temp_table
```

Output :

```
num_col   char_col
-------   --------
```

Ketika blok T-SQL (hasil konversi PL/SQL) dieksekusi maka :

```
(1 row(s) affected)
(1 row(s) affected)
Hello World
This message brought to you by PL/SQL
```

Dan ketika memeriksa tabel temp_table dengan perintah :

```
SELECT * FROM temp_table
```

Output :

```
num_col   char_col
-------   -----------------------------------
1         Hello World
2         This message brought to you by PL/SQL
```

Dari perbandingan kedua bahasa tersebut pada masing-masing RDBMS, menghasilkan keluaran yang sama, dimana setiap statement pada bahasa masukan (PL/SQL) bisa diterjemahkan menjadi T-SQL.

Tabel 4.1 – Tabel Hasil Uji Konversi Elemen PL/SQL Studi Kasus 1

No.	Elemen PL/SQL	Status Konversi
1	Variable Declaration	Sukses
2	Variable Assignment	Sukses
3	PL/SQL block	Sukses
4	INSERT statement	Sukses
5	SELECT statement	Sukses
6	DBMS_OUTPUT.PUT_LINE statement	Sukses

b. Blok PL/SQL dan Exception

PL/SQL :

```
DECLARE
   e_TooManyStudents    EXCEPTION;
   v_CurrentStudents    NUMBER(3);
   v_MaxStudents        NUMBER(3);
   v_ErrorCode          NUMBER;
   v_ErrorText          VARCHAR2(200);

BEGIN
   SELECT current_students, max_students
   INTO v_CurrentStudents, v_MaxStudents
   FROM classes
   WHERE department = 'HIS' AND course = 101;

   IF v_CurrentStudents > v_MaxStudents THEN
      RAISE e_TooManyStudents;
   END IF;

EXCEPTION
   WHEN e_TooManyStudents THEN
      INSERT INTO log_table(info)
         VALUES('History 101 has ' ||
                v_CurrentStudents ||
                'students : max allowed is ' ||
                 v_MaxStudents);
   WHEN OTHERS THEN
      v_ErrorCode := SQLCODE;
      v_ErrorText := SUBSTR(SQLERRM, 1, 200);
      INSERT INTO log_table(code, message, info)
         VALUES(v_ErrorCode, v_ErrorText, 'Oracle
                error occured');

END;
```

T-SQL :

```
DECLARE @e_toomanystudents INT
DECLARE @error_number INT
DECLARE @v_currentstudents NUMERIC(3)
DECLARE @v_maxstudents NUMERIC(3)
DECLARE @v_errorcode NUMERIC
DECLARE @v_errortext VARCHAR(200)

BEGIN
   SELECT
      @v_currentstudents = current_students,
      @v_maxstudents = max_students
   FROM classes
   WHERE department = 'HIS' AND course = 101

   IF @v_currentstudents > @v_maxstudents
      BEGIN
         SET @error_number = 54301
      END
```

```
-- Manual Intervention Needed for Exception Handler
   IF (@@ERROR <> 0) OR (@error_number <> 0)
      GOTO ExceptionBlock

   GOTO ExitLabel

ExceptionBlock:
   BEGIN --(
      IF @error_number = 54301
         BEGIN --((
            INSERT INTO log_table (info)
            VALUES('History 101 has ' +
                   @v_currentstudents +
                   'students : max allowed is ' +
                   @v_maxstudents)
         END --))
      ELSE
         BEGIN --((
            SET @v_errorcode = @@ERROR
            SET @v_errortext = SUBSTRING(
               CAST((SELECT description
                     FROM sysmessages
                     WHERE error = @@ERROR)
                  AS VARCHAR),1,200)
            INSERT INTO log_table (code,message,info)
            VALUES(@v_errorcode,@v_errortext,'Oracle
                   error occured')
         END --))
   END --)

ExitLabel:
END
```

Pengujian pada *Oracle (dengan SQLPlus):*

Untuk inisialisasi :

untuk tabel classes dengan department = ‘HIS’ dan course = 101,

max_students = 30 dan current_students = 35 maka :

```
SQL> SET SERVEROUTPUT ON;
SQL> SELECT * FROM log_table;
```

Output :

```
no rows selected
```

Ketika blok PL/SQL tersebut dieksekusi maka untuk memeriksa perubahan pada tabel log_table :

```
SQL> SELECT info FROM log_table;
```

Output :

```
INFO
-------------------------------------------
History 101 has 35 students : max allowed is 30
```

Pengujian pada *Microsoft SQL Server (dengan SQL Query Analyzer) :*

Untuk inisialisasi :

untuk tabel classes dengan department = 'HIS' dan course = 101, max_students = 30 dan current_students = 35 maka :

```
SELECT * FROM log_table
```

Output :

```
code   message   info
----   -------   ----
```

Ketika blok PL/SQL tersebut dieksekusi maka untuk memeriksa perubahan pada tabel log_table :

```
SELECT * FROM log_table
```

Output :

```
code   message   info
----   -------   ----
```

Dari perbandingan kedua bahasa tersebut pada masing-masing RDBMS, terdapat perbedaan keluaran. PL/SQL menghasilkan perubahan pada tabel *log*_table, tetapi T-SQL tidak menghasilkan perubahan pada tabel yang bersangkutan. Blok T-SQL (hasil penerjemahan PL/SQL) tidak bisa dieksekusi meskipun secara sintaks benar. Pada blok T-SQL terdapat

ketidakcocokan tipe data antara variabel *@v_currentstudents* dan *@v_maxstudents* dengan kolom *info* pada tabel *log_table*. Hal tersebut berkenaan dengan analisis semantik yang tidak ditangani pada perangkat lunak ini.

Untuk studi kasus di atas, jika pada kedua variabel tersebut diubah menjadi *CAST (@v_currentstudents AS VARCHAR)* dan *CAST (@v_maxstudents AS VARCHAR)*, maka akan menghasilkan keluaran (output) yang sama dengan keluaran pada *SQLPlus* (*Oracle*).

Analisis sintaks PL/SQL dan T-SQL :

Blok PL/SQL di atas terdiri dari bagian *deklarasi variabel*, *eksekusi statement* dan *exception handler*. Perbedaan yang mendasar antara PL/SQL dan T-SQL adalah, PL/SQL membagi blok-bloknya sesuai dengan fungsinya (seperti pada contoh di atas), sedangkan T-SQL tidak membedakan antara blok deklarasi, eksekusi statement dan exception handler.

Untuk deklarasi *exception*, T-SQL tidak menangani hal ini. Pada T-SQL *User defined exception* bertipedata *integer* yang nantinya akan diinisialisasi dengan *error number* (*lampiran d*).

Tabel 4.2 – Tabel Hasil Uji Konversi Elemen PL/SQL Studi Kasus 2

No.	Elemen PL/SQL	Status Konversi
1	Variable Declaration	Sukses
2	PL/SQL block	Sukses
3	SELECT statement	Sukses
4	IF statement	Sukses
5	Exception Handler	Sukses
6	INSERT statement	Sukses secara sintaks,tapi tidak secara semantik

c. Cursor

PL/SQL :

```
DECLARE
      v_StudentID       NUMBER(5);
      v_FirstName       VARCHAR2(20);
      v_LastName        VARCHAR2(20);
      CURSOR c_HistoryStudents IS
         SELECT id, first_name, last_name
         FROM students
         WHERE major = 'History';
BEGIN
   OPEN c_HistoryStudents;
   FETCH c_HistoryStudents
      INTO v_StudentID, v_FirstName, v_LastName;
   WHILE c_HistoryStudents%FOUND LOOP
      INSERT INTO registered_students
         (student_id, department, course)
      VALUES(v_StudentID, 'HIS', 301);
      INSERT INTO temp_table(num_col, char_col)
      VALUES(v_StudentID,
          v_FirstName || ' ' || v_LastName);
      FETCH c_HistoryStudents
         INTO v_StudentID, v_FirstName, v_LastName;
   END LOOP;
   CLOSE c_HistoryStudents;
   COMMIT;
END;
```

T-SQL :

```
DECLARE @v_studentid NUMERIC(5)
DECLARE @v_firstname VARCHAR(20)
DECLARE @v_lastname VARCHAR(20)
DECLARE @c_historystudents CURSOR
BEGIN
   SET @c_historystudents = CURSOR FOR
   SELECT id ,first_name ,last_name
   FROM students
   WHERE major = 'History'
   OPEN @c_historystudents
   FETCH NEXT FROM @c_historystudents
      INTO @v_studentid, @v_firstname, @v_lastname
   WHILE @@FETCH_STATUS = 0
      BEGIN
         INSERT INTO registered_students
            (student_id,department,course)
         VALUES(@v_studentid,'HIS',301)
         INSERT INTO temp_table (num_col,char_col)
         VALUES(@v_studentid,
                @v_firstname + ' ' + @v_lastname)
         FETCH NEXT FROM @c_historystudents
            INTO @v_studentid, @v_firstname,
               @v_lastname
      END
   CLOSE @c_historystudents
```

```
        DEALLOCATE @c_historystudents
        COMMIT
    END
```

Pengujian pada *Oracle (dengan SQLPlus):*

Untuk inisialisasi :

```
SQL> SELECT * FROM registered_students;
```

Output :

```
STUDENT_ID DEP    COURSE G
---------- --- --------- -
     10000 CS        102 A
     10002 CS        102 B
     10003 CS        102 C
     10000 HIS       101 A
     10001 HIS       101 B
     10002 HIS       101 B
     10003 HIS       101 A
     10004 HIS       101 C
     10005 HIS       101 C
     10006 HIS       101 E
     10007 HIS       101 B
     10008 HIS       101 A
     10009 HIS       101 D
     10010 HIS       101 A
     10008 NUT       307 A
     10010 NUT       307 A
     10009 MUS       410 B
     10006 MUS       410 E
```

Ketika blok PL/SQL tersebut dieksekusi maka untuk memeriksa perubahan pada tabel registered_students :

```
SQL> SELECT * FROM registered_students;
```

Output :

```
STUDENT_ID DEP    COURSE G
---------- --- --------- -
     10000 CS        102 A
     10002 CS        102 B
     10003 CS        102 C
     10000 HIS       101 A
     10001 HIS       101 B
     10002 HIS       101 B
     10003 HIS       101 A
     10004 HIS       101 C
     10005 HIS       101 C
     10006 HIS       101 E
     10007 HIS       101 B
     10008 HIS       101 A
```

```
10009 HIS       101 D
10010 HIS       101 A
10008 NUT       307 A
10010 NUT       307 A
10009 MUS       410 B
10006 MUS       410 E
10001 HIS       301
10004 HIS       301
10005 HIS       301
```

Pengujian pada *Microsoft SQL Server (dengan SQL Query Analyzer) :*

Untuk inisialisasi :

```
SELECT * FROM registered_students
```

Output :

```
student_id   department   course   grade
----------   ----------   ------   -----
10000        CS           102      A
10002        CS           102      B
10003        CS           102      C
10000        HIS          101      A
10001        HIS          101      B
10002        HIS          101      B
10003        HIS          101      A
10004        HIS          101      C
10005        HIS          101      C
10006        HIS          101      E
10007        HIS          101      B
10008        HIS          101      A
10009        HIS          101      D
10010        HIS          101      A
10008        NUT          307      A
10010        NUT          307      A
10009        MUS          410      B
10006        MUS          410      E
```

Ketika blok T-SQL (hasil konversi PL/SQL) dieksekusi maka untuk memeriksa perubahan pada tabel registered_students :

```
SELECT * FROM registered_students
```

Output :

```
student_id   department   course   grade
----------   ----------   ------   -----
10000        CS           102      A
10002        CS           102      B
10003        CS           102      C
10000        HIS          101      A
10001        HIS          101      B
```

```
10002          HIS            101       B
10003          HIS            101       A
10004          HIS            101       C
10005          HIS            101       C
10006          HIS            101       E
10007          HIS            101       B
10008          HIS            101       A
10009          HIS            101       D
10010          HIS            101       A
10008          NUT            307       A
10010          NUT            307       A
10009          MUS            410       B
10006          MUS            410       E
10001          HIS            301       NULL
10004          HIS            301       NULL
10005          HIS            301       NULL
```

Dari perbandingan kedua bahasa tersebut pada masing-masing RDBMS, menghasilkan keluaran yang sama, dimana setiap statement pada bahasa masukan (PL/SQL) bisa diterjemahkan menjadi T-SQL.

Analisis sintaks PL/SQL dan T-SQL :

Sintaks PL/SQL di atas adalah bentuk *cursor* dengan segala atributnya. Cursor memiliki beberapa bagian penting, yakni *deklarasi cursor*, *open cursor*, *fetch cursor* dan *close cursor*. Kesemuanya juga dimiliki oleh T-SQL, meskipun penggunaannya ada perbedaan. Dalam *open cursor statement*, T-SQL harus menginisialisasi nama variabel cursor (atau nama cursor) dengan *select statement* (dalam PL/SQL hal ini dilakukan pada bagian deklarasi).

Fetch cursor statement memiliki struktur yang hampir sama, kecuali pada penanganan *bind variable* (*bisa dilihat pada lampiran e*). T-SQL tidak menangani segala bentuk *bind variable*. Dan untuk *close cursor statement*, kedua bahasa memiliki struktur yang sama, kecuali pada penanganan *bind variable* seperti halnya *fetch cursor statement*.

Tabel 4.3 – Tabel Hasil Uji Konversi Elemen PL/SQL Studi Kasus 3

No.	Elemen PL/SQL	Status Konversi
1	Variable Declaration	Sukses
2	Cursor Declaration	Sukses
3	PL/SQL block	Sukses
4	OPEN statement	Sukses
5	FETCH statement	Sukses
6	WHILE LOOP Statement	Sukses
7	INSERT statement	Sukses
8	CLOSE statement	Sukses
9	COMMIT statement	Sukses

Sukses pada *status konversi* dalam tabel hasil uji menunjukkan bahwa elemen bisa diterjemahkan menjadi bahasa T-SQL, yang benar/valid secara sintaks dan semantik.