esProc

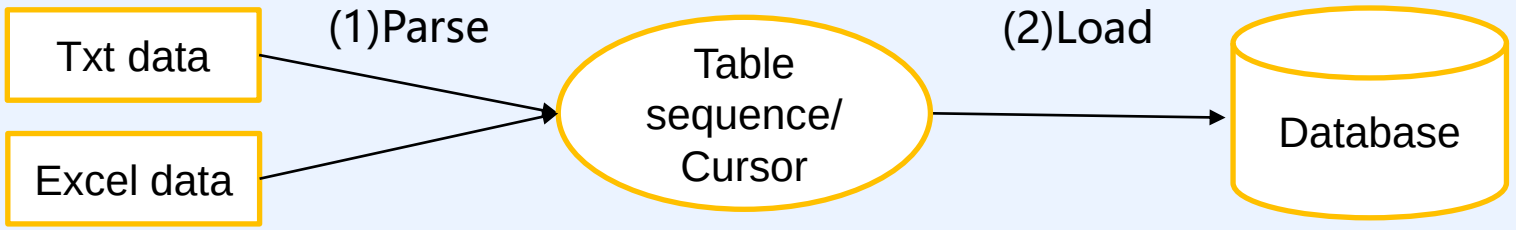
- Your data processing expert

Loading Text & Excel Data into the Database

By Raqsoft

Contents

1. The flowchart



(1)Parse

Read Structured Text Computing with esProc to learn how to parse a txt file into a table sequence or a cursor(http://c.raqsoft.com/article/1571711703952)

Read SPL parsing and exporting Excel learn how to parse an Excel file into a table sequence or a cursor (http://c.raqsoft.com/article/1571712027010)

(2)Load

Here we' ll illustrate how SPL stores the parsing result (a table sequence or a cursor) to the database.

2. Write to an empty table

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.update(A1,table1)	Update data in table sequence A1 to the empty table *table1*
4	=A2.close()	Close database connection

A1's table sequence is the parsing result to be updated to database

A1 and *table1* have same field names and values are populated by field names

table1 needs to be created in advance

2. Write to an empty table

Big data: If the table sequence or the cursor contains a huge volume of data that cannot fit into the memory, read data from them with cursor

	A	Note
1	=file("data.txt").cursor@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.update(A1,table1)	Update data in table sequence A1 to the empty table *table1*
4	=A2.close()	Close database connection

Field mapping: If A1 and *table1* have different field names

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=A1.new(F1:col1,F2:col2,F3:col3)	Fi is A1's field name and coli is *table1*'s field name
3	=connect("db")	Connect to database
4	=A3.update(A2,table1)	Update data in table sequence A1 to the empty table *table1*
5	=A3.close()	Close database connection

3. Write to an existing table by insert and modify

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.update(A1,table1;key1,key2)	Update data in table sequence A1 to *table1*
4	=A2.close()	Close database connection

In A3,key1 and key2 are *table1*'s composite primary key

The update matches *table1*'s records according to its primary key. Insert a record to the database table if Aa value existing in A1 but doesn't exist in *table1*; for a value existing in both A1 and *table1*, compare other field values and update A1's field values to database if there are any differences.

In A3, parameter key1 and key2 can be omitted and the program will auto-read *table1*'s primary key. If the key cannot been found and error is reported, you need to explicitly specify the primary key.

Note: You can only insert or modify data but cannot delete data from the database. A record whose key value doesn't exist in A1 won't be deleted.

4. Replace table by delete

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.query("select key1,key2 from table1")	Query existing records in *table1* by reading primary key values only
4	=A2.update(A1:A3,table1;key1,key2)	Update data in table sequence A1 to *table1*
5	=A2.close()	Close database connection

Delete records from database if key values don’t exist in A1.

In this case you need to read in database table’s primary key values (A3) and delete records whose key values don’t exist in A. The insert and modify are handled in same way as previous.

5. Insert only to increase performance

Method 1:

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.update@i(A1,table1)	Insert new records in table sequence A1 into *table1*
4	=A2.close()	Close database connection

The use of @i option won't make a comparison but generates an INSERT statement directly on A1's records to append them to *table1*. This makes the write action faster.

Method 2: Call db.execute() to execute an ready SQL to make the write even faster.

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.execute(A1,"insert into table1 (col1,col2,col3) values (?,?,?)",#1,#2,#3)	Insert new records in table sequence A1 into *table1*
4	=A2.close()	Close database connection

6. UPDATE only to increase performance

Method 1:

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.update@u(A1,table1)	Update records in table sequence A1 into *table1*
4	=A2.close()	Close database connection

The use of @u option won't make a comparison but generates an UPDATE statement directly on A1's records to append them to *table1*. This makes the write action faster.

Method 2: Call db.execute() to execute an ready SQL to make the write even faster.

	A	Note
1	=file("data.txt").import@t()	Same syntax for importing other data sources
2	=connect("db")	Connect to database
3	=A2.execute(A1,"update table1 set col2=?,col3=? where col1=?)",#2,#3,#1)	Update records in table sequence A1 into *table1* by key value col1
4	=A2.close()	Close database connection

7. Parallel data write

Use parallel processing to increase efficiency if the data source is a huge txt file.

	A	B	Note
1	>n=5		Define the number of parallel threads
2	fork to(n)	=connect("db")	Connect to database
3		=file("data.txt").cursor@t(;A2:n)	Divide the txt file into n segments and read in each one as a cursor
4		=B2.update(B3,table1)	Update the current segment to database
5		=B3.close()	Close database connection

You can reference the syntax in the above examples to write B4's code as needed

Note: An Excel file cannot be segmented

8. Batch write using a database tool

Using a database tool to update data in a txt file to a database table is much faster than using JDBC-based SPL db.update() and db.execute().

If there are multiple txt file of same structures, use multithreaded processing to call the database tool in each thread to get the best performance possible.

Take Oracle as an example:

There are ten txt files under a same directory. They are data_1.txt, data_2.txt, ..., data_10.txt. The Oracle SQL for writing them to the database is as follows (*data.ctl*):

```
LOAD DATA APPEND INTO TABLE table1 (id terminated by '|',amount terminated by '|')
```

SPL script:

	A	B
1	="sqlldr user/password@pdborcl control=/home1/data.ctl direct=true data="	
2	="/home1/data_"	
3	fork to(10)	=A2+string(A3)+".txt"
4		>system(A1+B3)