Teaching Tip

Teaching Advanced SQL Skills: Text Bulk Loading

David Olsen
Karina Hauser
Utah State University
Management Information Systems
3515 Old Main Hill
Logan, UT 84322-3515
david.olsen@usu.edu karina.hauser@usu.edu

ABSTRACT

Studies show that advanced database skills are important for student to be prepared for the today's highly competitive job market. A common task for database administrators is to insert large amount of data into a database. This paper illustrates how an up-to-date, advanced database topic, namely bulk insert, can be incorporated into a database class. It gives detailed examples on how to import different file types and discusses associated issues.

Keywords: SQL, Bulk insert, Database systems

1. INTRODUCTION

The job market for database administrator is growing faster than other information technology (IT) related professions with a predicted growth of 18% to 26% over the next ten years (U.S. Department of Labor: Bureau of Labor Statistics, 2006-07 Edition). The vast majority of universities have recognized the need to integrate database education into their core IT curriculum and offer at least one course in database management (Kung, Yang and Zhang, 2006). To take advantage of the expanding job market students have to be equipped with not only fundamental database knowledge and skills, like normalization and standard SQL (Structured Query Language), but also with advanced SQL skills that are not traditionally covered in text books. Up-to-date knowledge and skills in emerging technologies are necessary for students to secure entry-level jobs, whereas fundamental knowledge and skills are valuable for job advancement (Lightfoot, 1999).

In the future, data transfer will be done in not only gigabytes but also terabytes and petabytes. (Babcock, 2006). The ability to quickly, inexpensively, and accurately transfer business and customer information as well as the ability to store, maintain, and analyze it will be a vital link to success. Traditionally, this challenge has been handled with Database Management Systems (DBMS). Unfortunately, database technology falls short of providing efficient automation techniques for the challenges that large collections of data raise. As storage back-end, many applications rely on relational databases, which are designed towards large data volumes. A new age of data transfer technologies are being used to tackle the demanding challenges of bridging backend storage and data transfer. One of the more popular and effective exchange methods for large-scale data transfer is the use of XML (Extensible Markup Language). Further, to open up the content of XML documents to analysis with declarative query languages, efficient bulk loading techniques are necessary. While bulk load and update algorithms for XML data stored in relational format encounter problems, opportunities are prevalent. To get the best performance out of relational database systems, one should make careful use of edit scripts and replace them with bulk operations if more than very small portion of the database is updated.

2. TEXT BULK LOADING EXPLANATION AND THEORY

The BULK INSERT statement was introduced in SQL Server 7 and allows one to interact with the Bulk Copy Program (BCP) via a script. The BCP utility copies data between an instance of Microsoft® SQL Server and a data file in a user-specified format. BULK INSERT makes an "under-the-hood" call to BCP to import the data. BCP is a command-line utility that can be used to import/export data in either character or native SQL Server format.

The BULK INSERT statement supports the use of a specialized format file that stores format information for each field in a data file. A format file might also contain information about the corresponding SQL Server table. The format file can be used to provide all the format information that is required to bulk export data from and bulk import data to an instance of SQL Server.

Format files provide a flexible way to interpret data as it is in the data file during import, and also to format data in the data file during export. This flexibility eliminates the need to write special-purpose code to interpret the data or reformat the data to the specific requirements of SQL Server or the external application. For example, for bulk importing data to be loaded into a database table that requires comma-separated values, a format file to insert commas as field terminators in the exported data can be used.

SQL Server 2005 supports two kinds of format files: Non-XML format files and XML format files. Non-XML format files were supported by earlier versions of SQL Server; XML format files are new in SQL Server 2005. To bulk import data into an instance of SQL Server, the BULK INSERT statement works with the query processor. The BULK INSERT method converts the data in a data file into OLE DB (Object Linking and Embedding DataBase) rowsets. The OLE DB rowsets are inserted into the target table by the query processor, which plans and optimizes each operation.

Performance considerations can also be significant when large amounts of data are being imported. In some cases, performance can be improved by changing how a bulk-import or bulk-export operation handles one or more of the following:

- Batch switches
- Constraint checking of CHECK constraints
- How bulk transactions are logged. This is relevant for databases that typically use the full recovery model.
- Ordering exported data
- Parallel data importing
- Table locking
- Trigger execution

2.1 Using the BULK INSERT Function for Non-XML Format Files

The process consists of 3 steps:

1. Preparation of file for insert
2. Writing the query
3. Running the query

2.1.1 Preparation of file for insert

In preparation for the insert, field need to be separated by a comma or another unused character. This can be done through an export wizard or with the find-and-replace feature in a text editor. The end of each record needs to have a different unique identifier. This is commonly a carriage return (\n). Field headings in the first line of the file are optional. Figure 1 shows an example of text prepared for a bulk insert. The file needs to be saved on the same drive as the database. The example file was saved as 'C:\bulkloadtemp.txt'.

2.1.2 Writing the query

The following query is used to bulk insert the file created in the first step (bulkloadtemp.txt) into the table bulkExampleTable:

```
BULK INSERT bulkExampleTable
        FROM 'C:\bulkloadtemp.txt'
        WITH
        (
            ROWTERMINATOR = '\n',
            FIELDTERMINATOR = ',',
            FIRSTROW = 2
        )
```

ROWTERMINATOR specifies the character chosen to indicate the end of each row. FIELDTERMINATOR specifies the character chosen for the separator between fields. FIRSTROW = 2 specifies that the first data row is row 2, since the first row contains field headings. Parameters need to be separated by commas.

2.1.3 Running the query

After executing the query the table with the inserted data can be viewed and the result should look similar to Figure 2.

2.2 Using the BULK INSERT Function for XML Format Files

The format and data of the XML file, named xml_book.xml, used in our example is shown in Figure 3.

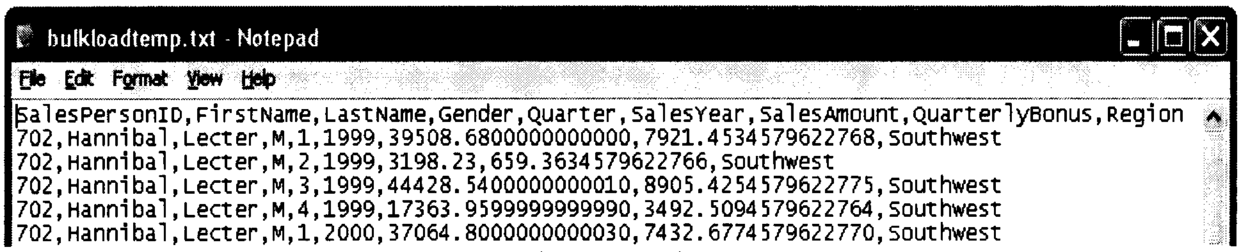
2.2.1 Creating the XML import script

Both the XML file and the database, that the file should be imported to, must reside on the same computer. There are several ways of accomplishing the XML import, and only one has been provided as an example. For other ways to import the XML please look at the provided references (Singh, 2002; Microsoft, n.d.).

```
DECLARE @FileName VARCHAR(255)
DECLARE @ExecCmd VARCHAR(255)
DECLARE @rc INT
DECLARE @FileContents VARCHAR(8000)

DROP TABLE tempXML

CREATE TABLE tempXML (
        PK INT NOT NULL IDENTITY(1,1),
        ThisLine VARCHAR(255)
SET @FileName = 'C:\xml_books.xml'
SET @ExecCmd = 'type ' + @FileName
SET @FileContents = ''

EXEC @rc = master.dbo.xp_cmdshell @ExecCmd
INSERT INTO tempXML (ThisLine)
        EXEC @rc = master.dbo.xp_cmdshell @ExecCmd

SELECT @FileContents =
        @FileContents + COALESCE(ThisLine,'') FROM
#tempXML
        ORDER BY pk
SELECT @FileContents
IF @rc = 1
       BEGIN
       RAISERROR('Could not execute "%s" because %s',
          16,1,@ExecCmd,@filecontents)
END
```

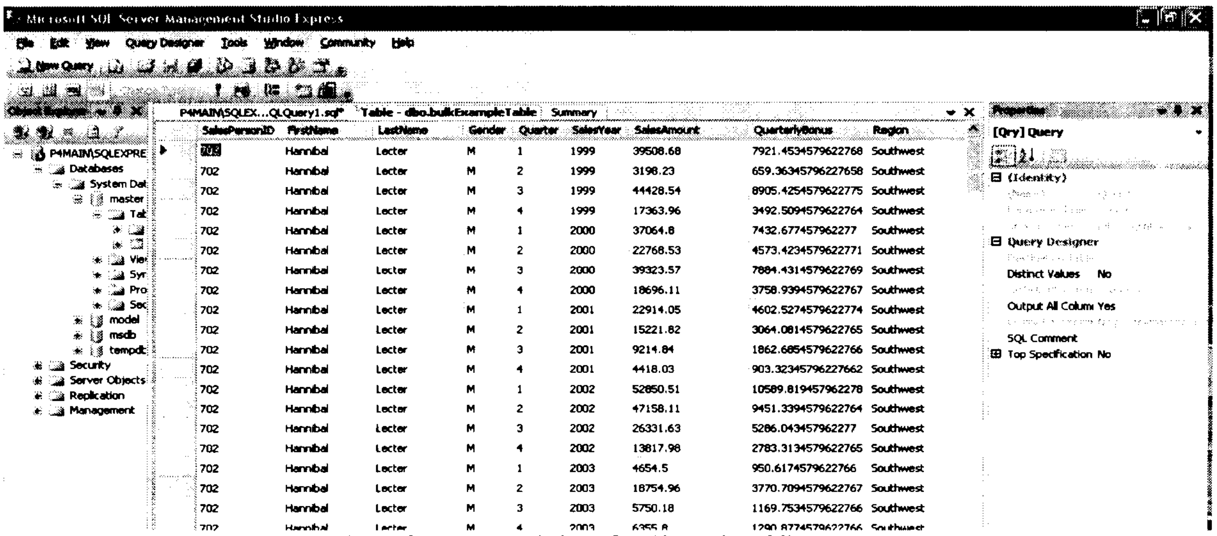
2.2.2 Running the XML import script

After running the script the table with the inserted data can be viewed and the result should look similar to Figure 4.



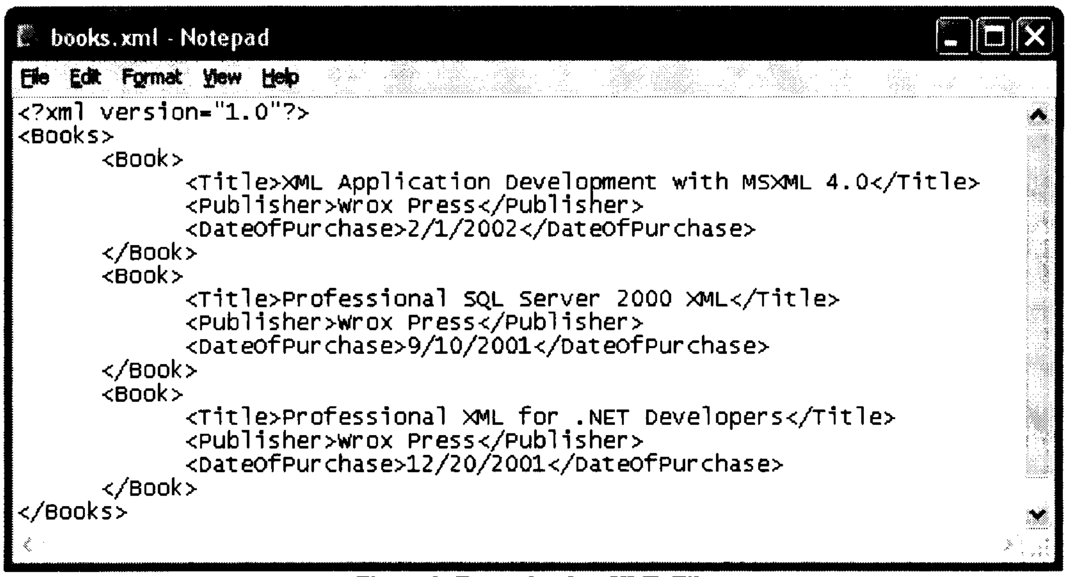
```
SalesPersonID,FirstName,LastName,Gender,Quarter,SalesYear,SalesAmount,QuarterlyBonus,Region
702,Hannibal,Lecter,M,1,1999,39508.6800000000000,7921.4534579622768,Southwest
702,Hannibal,Lecter,M,2,1999,3198.23,659.3634579622766,Southwest
702,Hannibal,Lecter,M,3,1999,44428.5400000000010,8905.4254579622775,Southwest
702,Hannibal,Lecter,M,4,1999,17363.9599999999990,3492.5094579622764,Southwest
702,Hannibal,Lecter,M,1,2000,37064.8000000000030,7432.6774579622770,Southwest
```

Figure 1: Example of Non-XML File Prepared for Insert



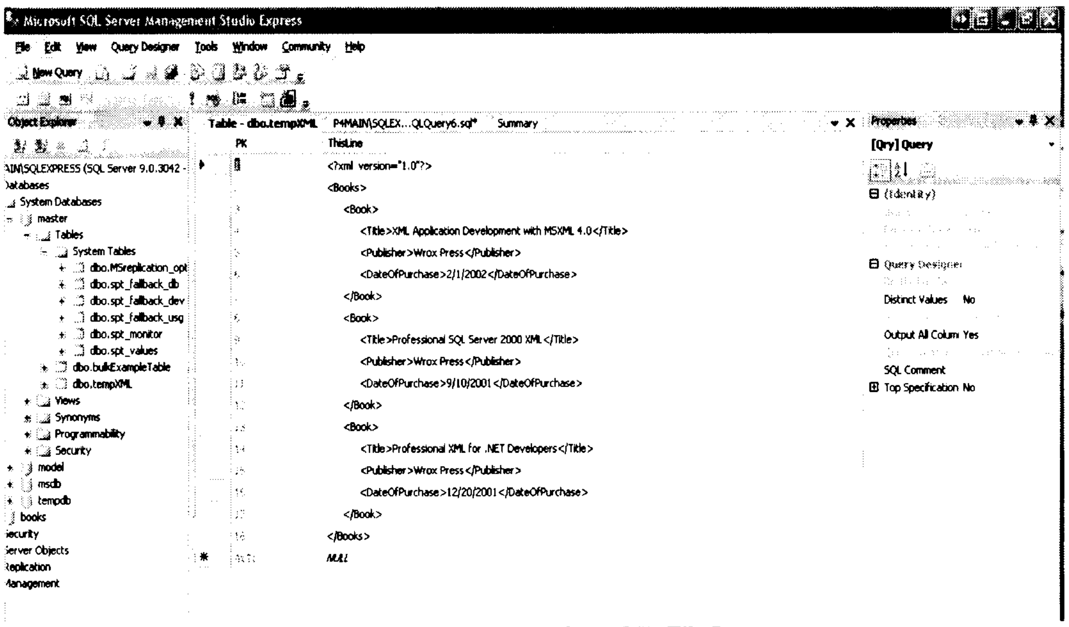
SalesPersonID	FirstName	LastName	Gender	Quarter	SalesYear	SalesAmount	QuarterlyBonus	Region
702	Hannibal	Lecter	M	1	1999	39508.68	7921.4534579622768	Southwest
702	Hannibal	Lecter	M	2	1999	3198.23	659.36345796227658	Southwest
702	Hannibal	Lecter	M	3	1999	44428.54	8905.4254579622775	Southwest
702	Hannibal	Lecter	M	4	1999	17363.96	3492.5094579622764	Southwest
702	Hannibal	Lecter	M	1	2000	37064.8	7432.677457962277	Southwest
702	Hannibal	Lecter	M	2	2000	22768.53	4573.4234579622771	Southwest
702	Hannibal	Lecter	M	3	2000	39323.57	7884.4314579622769	Southwest
702	Hannibal	Lecter	M	4	2000	18696.11	3758.9394579622767	Southwest
702	Hannibal	Lecter	M	1	2001	22914.05	4602.5274579622774	Southwest
702	Hannibal	Lecter	M	2	2001	15221.82	3064.0814579622765	Southwest
702	Hannibal	Lecter	M	3	2001	9214.84	1862.6854579622766	Southwest
702	Hannibal	Lecter	M	4	2001	4418.03	903.32345796227662	Southwest
702	Hannibal	Lecter	M	1	2002	52850.51	10589.819457962278	Southwest
702	Hannibal	Lecter	M	2	2002	47158.11	9451.3394579622764	Southwest
702	Hannibal	Lecter	M	3	2002	26331.63	5286.043457962277	Southwest
702	Hannibal	Lecter	M	4	2002	13817.98	2783.3134579622765	Southwest
702	Hannibal	Lecter	M	1	2003	4654.5	950.6174579622766	Southwest
702	Hannibal	Lecter	M	2	2003	18754.96	3770.7094579622767	Southwest
702	Hannibal	Lecter	M	3	2003	5750.18	1169.7534579622766	Southwest
702	Hannibal	Lecter	M	4	2003	6355.8	1290.8774579622766	Southwest

Figure 2: Database Table after Non-XML File Import



```xml
<?xml version="1.0"?>
<Books>
      <Book>
            <Title>XML Application Development with MSXML 4.0</Title>
            <Publisher>Wrox Press</Publisher>
            <DateOfPurchase>2/1/2002</DateOfPurchase>
      </Book>
      <Book>
            <Title>Professional SQL Server 2000 XML</Title>
            <Publisher>Wrox Press</Publisher>
            <DateOfPurchase>9/10/2001</DateOfPurchase>
      </Book>
      <Book>
            <Title>Professional XML for .NET Developers</Title>
            <Publisher>Wrox Press</Publisher>
            <DateOfPurchase>12/20/2001</DateOfPurchase>
      </Book>
</Books>
```

Figure 3: Example of an XML File



```
PK    ThisLine
      <?xml version="1.0"?>
      <Books>
         <Book>
            <Title>XML Application Development with MSXML 4.0</Title>
            <Publisher>Wrox Press</Publisher>
            <DateOfPurchase>2/1/2002</DateOfPurchase>
         </Book>
         <Book>
            <Title>Professional SQL Server 2000 XML</Title>
            <Publisher>Wrox Press</Publisher>
            <DateOfPurchase>9/10/2001</DateOfPurchase>
         </Book>
         <Book>
            <Title>Professional XML for .NET Developers</Title>
            <Publisher>Wrox Press</Publisher>
            <DateOfPurchase>12/20/2001</DateOfPurchase>
         </Book>
      </Books>
      NULL
```

```
Distinct Values     No
Output All Colum    Yes
SQL Comment
Top Specification   No
```

Figure 4: Database Table after XML File Import

4. REFERENCES

Babcock, C. (2006). "Data, Data, Everywhere." Retrieved 12/26/2006,from http://www.informationweek.com/shared/printableArticle.jhtml?articleID=175801775.

Kung, M., Yang, S. C. and Zhang, Y. (2006). "The Changing Information Systems (IS) Curriculum: A Survey of Undergraduate Programs in the United States." Journal of Education for Business, Vol. 81, No. 6, pp. 291-300.

Lightfoot, J. M. (1999). "Fads Versus Fundamentals: The Dilemma for Information Systems Curriculum Design." Journal of Education for Business, Vol. 75, No. pp. 43-50.

Microsoft. (n.d.). "MSDN: Examples of Bulk Importing and Exporting XML Documents." Retrieved 12/15/06, from http://msdn2.microsoft.com/engb/library/ms191184.aspx.

Singh, D. (2002). "PerfectXML: Importing XML into SQL Server 2000." Retrieved 12/28/06, from http://www.perfectxml.com/articles/xml/importxmlsql.asp.

U.S. Department of Labor: Bureau of Labor Statistics. (2006-07 Edition). "Occupational Outlook Quarterly." Retrieved: 12/27/06, from http://www.bls.gov/oco/ocos042.htm.

AUTHOR BIOGRAPHIES

David Olsen received his Ph.D. in Management Information Systems from and joined the MIS department at Utah State University in 1998. He teaches primarily in the database area as well as the MBA strategy and management course. His research interests include database concurrency control, accounting information systems, the integration of SQL, XML and XBRL, and database modeling. His research has been published in journals such as Communications of the ACM, Issues in Accounting Education, and the Journal of Database Management.

Karina Hauser is an assistant professor in the Management Information Systems department at Utah State University. She received her PhD in Decision Science and Information Technology at the University of Kentucky on a Toyota Fellowship. Her research interests are Systems Analysis and Design, Webdesign and Lean Manufacturing. Before going into academia, Karina spent 16 years in industry, first as a programmer and later as a consultant and project manager for Enterprise Resource Planning systems, mainly in the automotive sector.

Information Systems & Computing
Academic Professionals

Serving Information Systems Educators

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096