USING XML DATA FORMAT TO LOAD AND EXTRACT DATA BETWEEN TWO DIFFERENT TYPES OF DATABASES: A STUDY CASE OF DEVELOPMENT CUSTOMER PRICE INDEX (CPI) APPLICATION

Bahagian Pengurusan Maklumat

HASNI MOHD JUFRI AND MUHAMAD AZLAN YUSOFF

Abstract:

This study aims to identify how two different types of databases can share the same data without using the RDBMS data provider.

One case study was conducted to test load and extract the data between SQLite and Access databases by using xml data format on Mobile Customer Price Index (CPI) application and Client-Server based system.

The results of this study indicate that xml data format is a flexible file as a transport to exchange data between two different types of databases.

Introduction

XML (Extensible Markup Language) is a markup language.

XML defines a set of rules for Encoding documents in a format that is both human and machine readable.

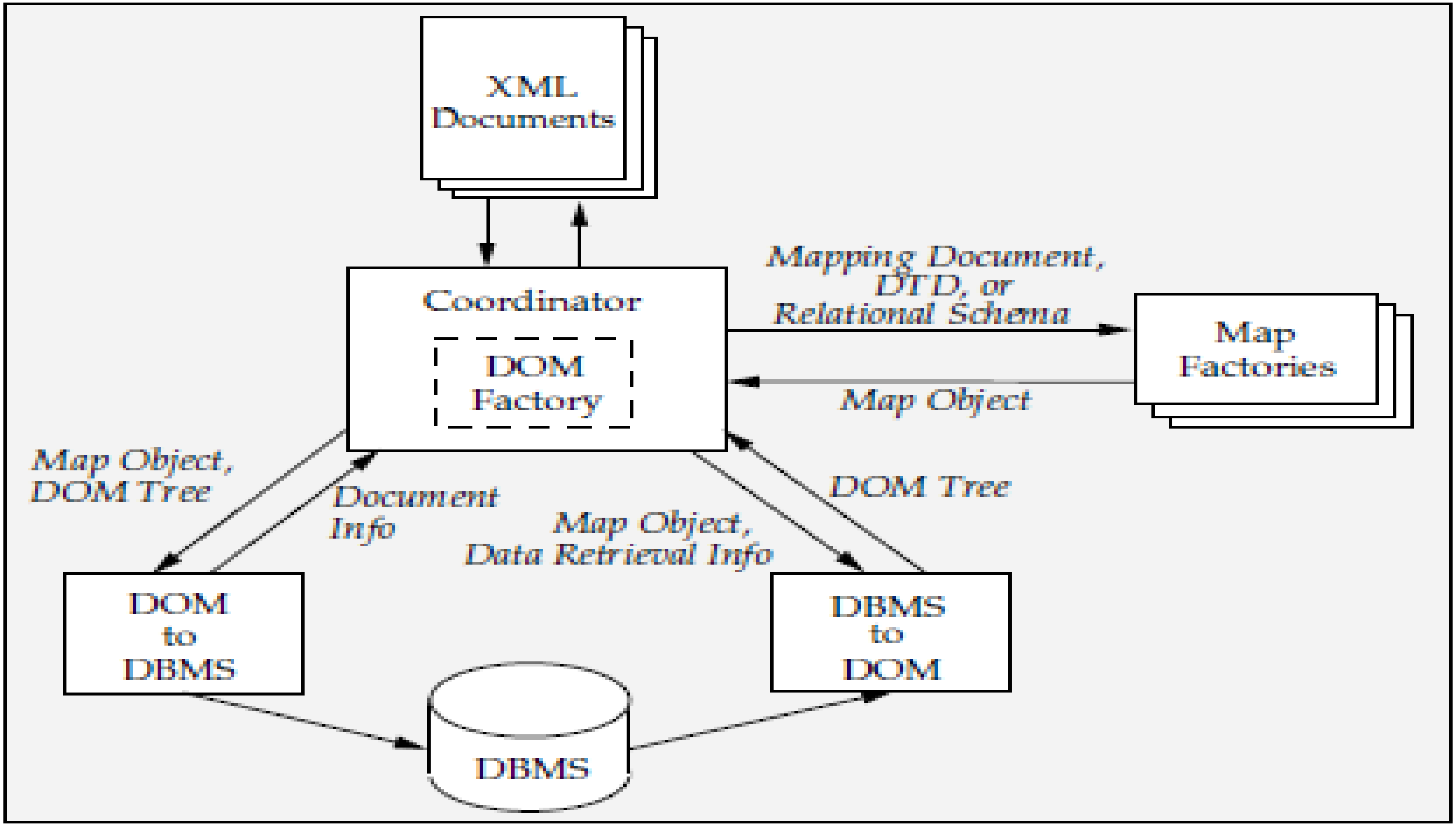
Introduction

- □ Challenges of modern data management is dealing with diverse data
- □ Data exchange - restructure data stored
- □ Specification by a schema mapping

Literature Review

- Recently, there has been a lot of research in the domain of storing XML documents in databases.
  - develop a specific XML database system.
  - use an OODBMS to store XML or hypertext documents.
  - interpret XML documents as graph structures which are stored in a relational database.
  - map XML documents to relational databases.

XML-DBMS architecture



Ronald Bourret (2000)

How XML help on exchange data between SQLite and Access databases

- Creating XML documents according to a known DTD from data extracted from a database
- Loading data from XML documents into relational tables with a known schema

DTD

- A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.

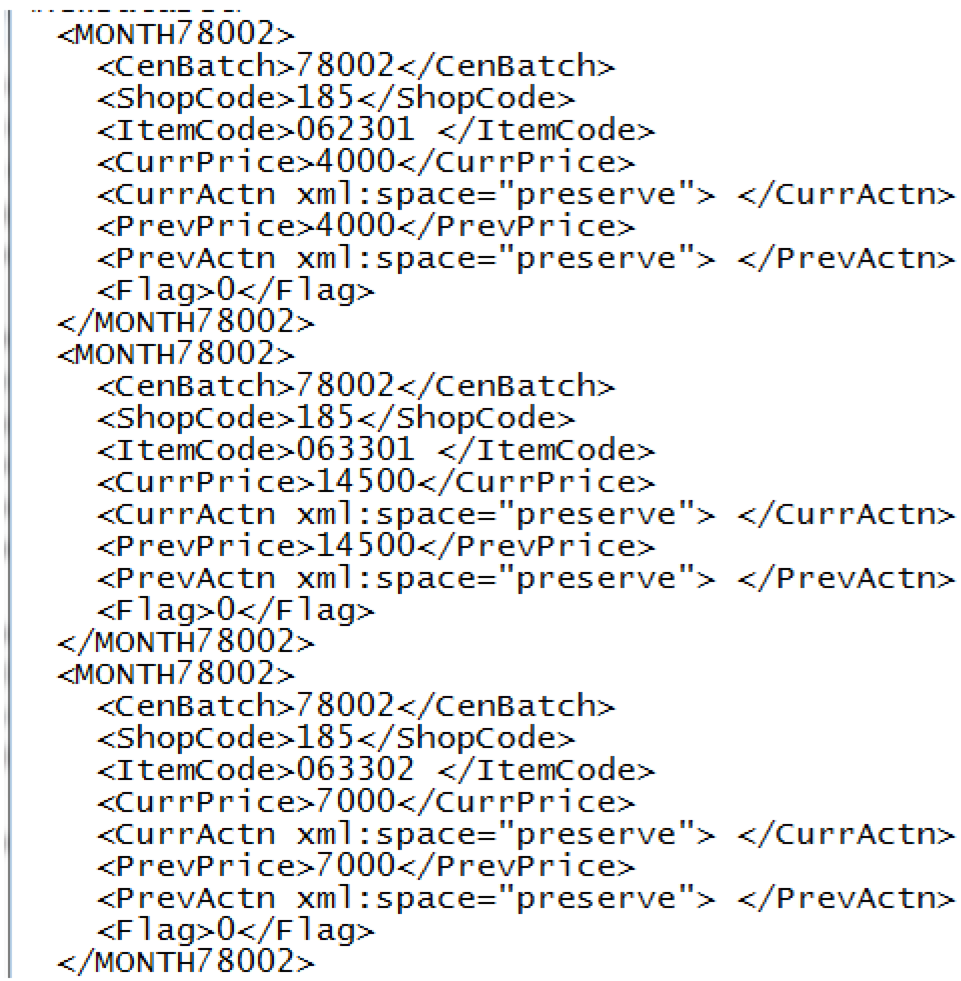
The building blocks of XML documents

- **Elements**
  - "hr", "br" and "img"
- **Tags**
  - <body>body text in between</body>.
- **Attributes**
  - <img src="computer.gif" />
- **PCDATA**
- **CDATA**
- **Entities**

The following entities are predefined in XML:

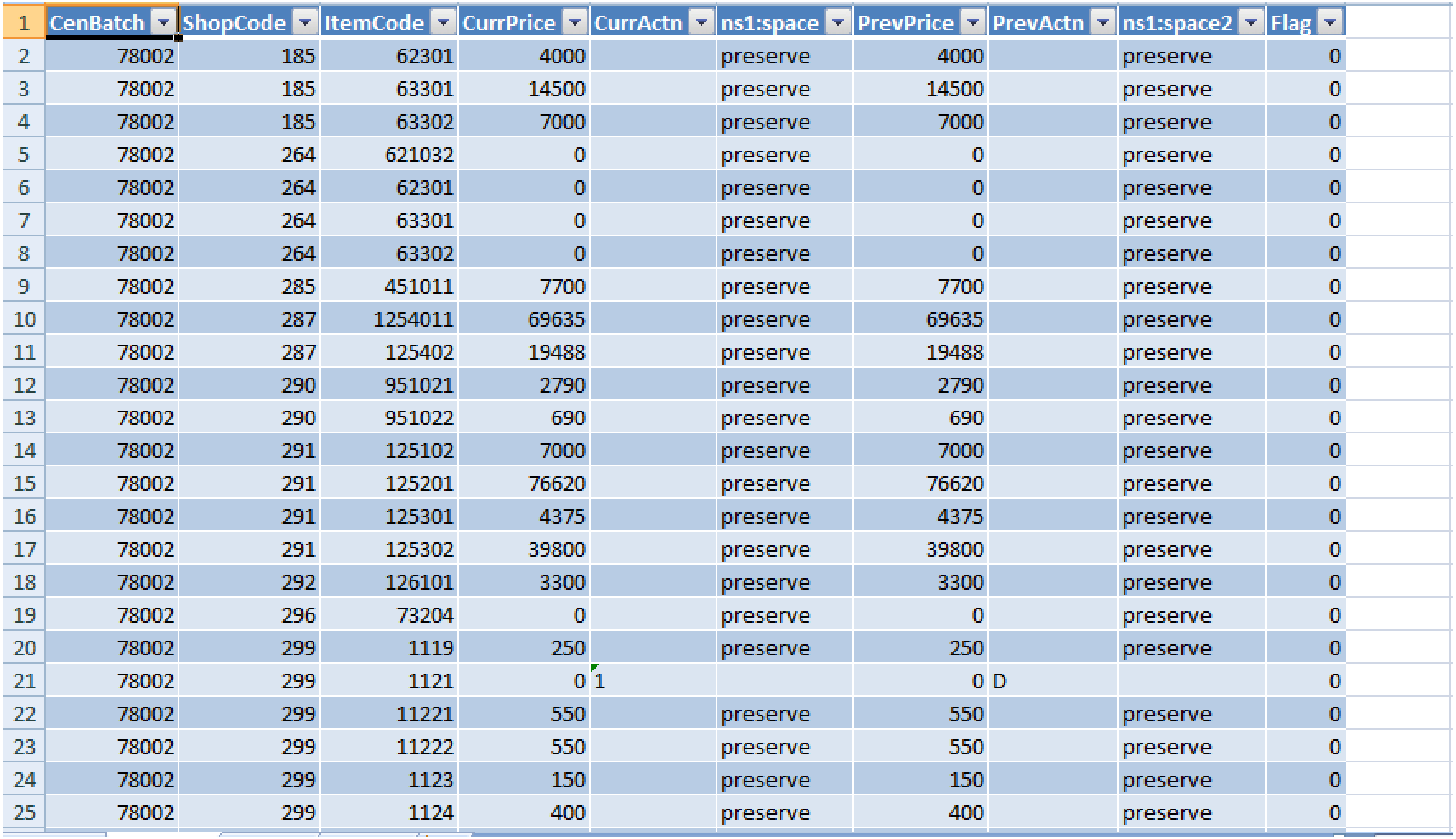
Entity References	Character
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

XML Code



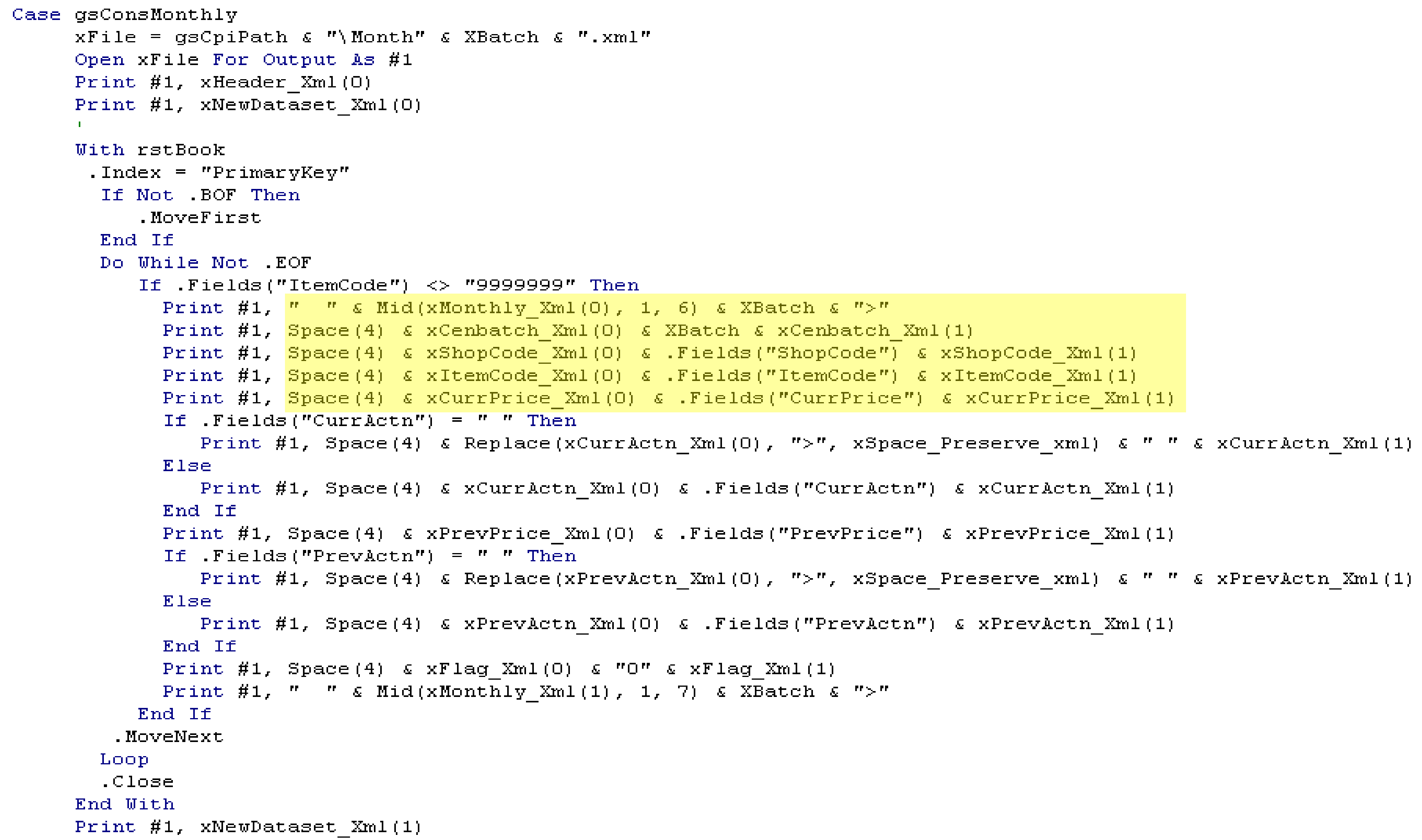
```
<MONTH78002>
  <CenBatch>78002</CenBatch>
  <ShopCode>185</ShopCode>
  <ItemCode>062301 </ItemCode>
  <CurrPrice>4000</CurrPrice>
  <CurrActn xml:space="preserve"> </CurrActn>
  <PrevPrice>4000</PrevPrice>
  <PrevActn xml:space="preserve"> </PrevActn>
  <Flag>0</Flag>
</MONTH78002>
<MONTH78002>
  <CenBatch>78002</CenBatch>
  <ShopCode>185</ShopCode>
  <ItemCode>063301 </ItemCode>
  <CurrPrice>14500</CurrPrice>
  <CurrActn xml:space="preserve"> </CurrActn>
  <PrevPrice>14500</PrevPrice>
  <PrevActn xml:space="preserve"> </PrevActn>
  <Flag>0</Flag>
</MONTH78002>
<MONTH78002>
  <CenBatch>78002</CenBatch>
  <ShopCode>185</ShopCode>
  <ItemCode>063302 </ItemCode>
  <CurrPrice>7000</CurrPrice>
  <CurrActn xml:space="preserve"> </CurrActn>
  <PrevPrice>7000</PrevPrice>
  <PrevActn xml:space="preserve"> </PrevActn>
  <Flag>0</Flag>
</MONTH78002>
```

XML Table



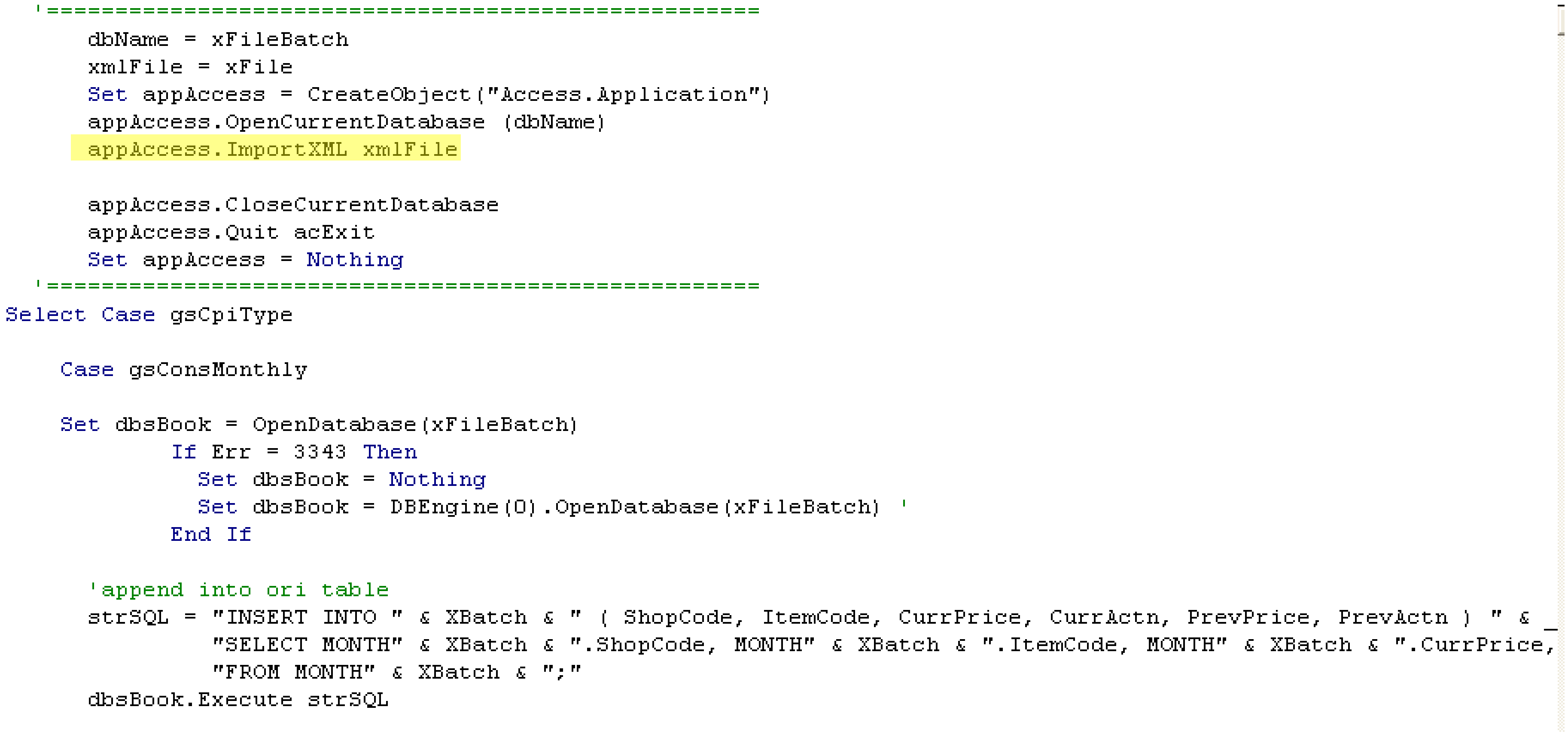
1	CenBatch	ShopCode	ItemCode	CurrPrice	CurrActn	ns1:space	PrevPrice	PrevActn	ns1:space2	Flag
2	78002	185	62301	4000		preserve	4000		preserve	0
3	78002	185	63301	14500		preserve	14500		preserve	0
4	78002	185	63302	7000		preserve	7000		preserve	0
5	78002	264	621032	0		preserve	0		preserve	0
6	78002	264	62301	0		preserve	0		preserve	0
7	78002	264	63301	0		preserve	0		preserve	0
8	78002	264	63302	0		preserve	0		preserve	0
9	78002	285	451011	7700		preserve	7700		preserve	0
10	78002	287	1254011	69635		preserve	69635		preserve	0
11	78002	287	125402	19488		preserve	19488		preserve	0
12	78002	290	951021	2790		preserve	2790		preserve	0
13	78002	290	951022	690		preserve	690		preserve	0
14	78002	291	125102	7000		preserve	7000		preserve	0
15	78002	291	125201	76620		preserve	76620		preserve	0
16	78002	291	125301	4375		preserve	4375		preserve	0
17	78002	291	125302	39800		preserve	39800		preserve	0
18	78002	292	126101	3300		preserve	3300		preserve	0
19	78002	296	73204	0		preserve	0		preserve	0
20	78002	299	1119	250		preserve	250		preserve	0
21	78002	299	1121	0	1		0	D		0
22	78002	299	11221	550		preserve	550		preserve	0
23	78002	299	11222	550		preserve	550		preserve	0
24	78002	299	1123	150		preserve	150		preserve	0
25	78002	299	1124	400		preserve	400		preserve	0

Algorithm: Export MS-Access (VB) to XML



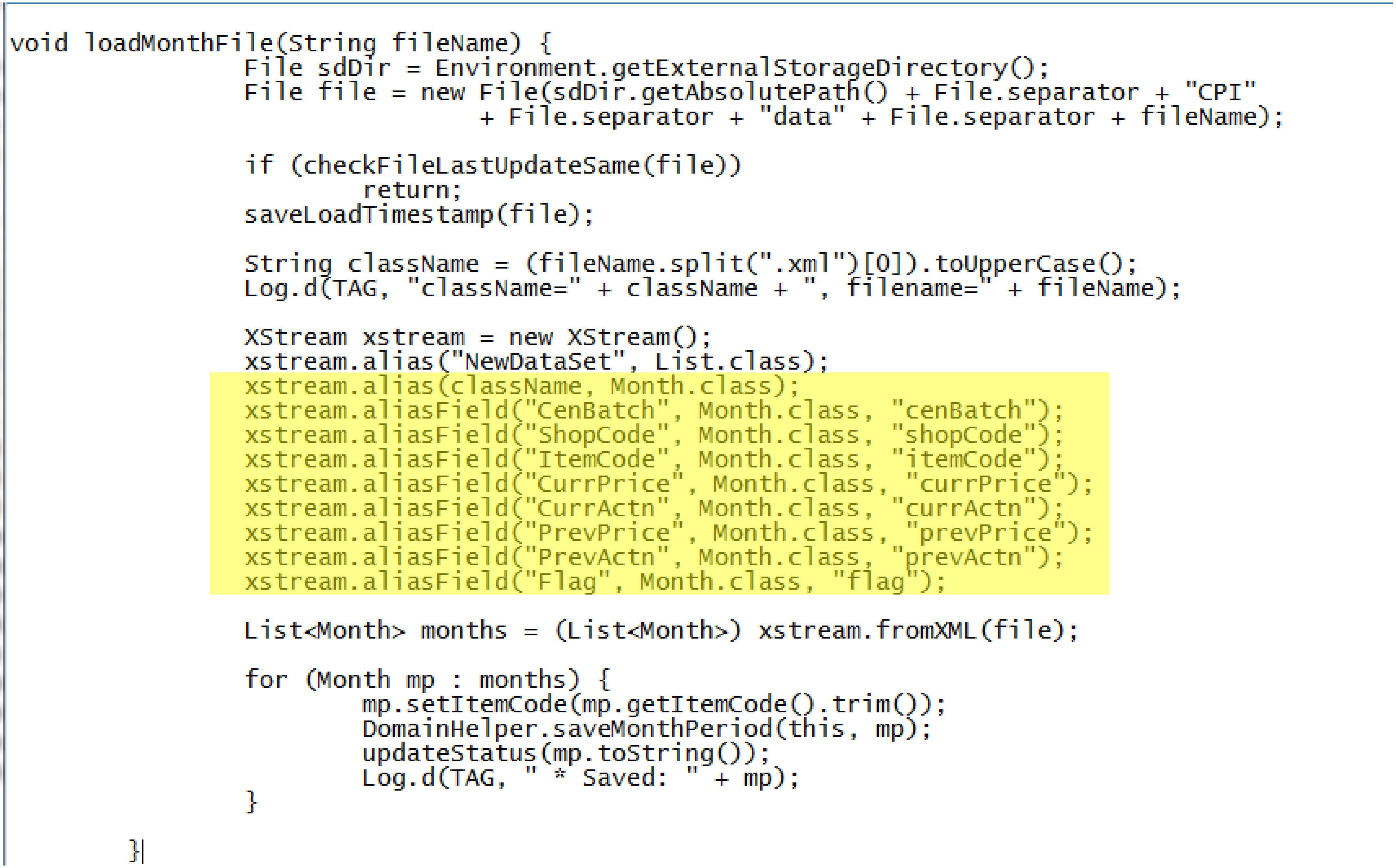
```
Case gsConsMonthly
     xFile = gsCpiPath & "\Month" & XBatch & ".xml"
     Open xFile For Output As #1
     Print #1, xHeader_Xml(0)
     Print #1, xNewDataset_Xml(0)
     '
     With rstBook
      .Index = "PrimaryKey"
       If Not .BOF Then
          .MoveFirst
       End If
       Do While Not .EOF
          If .Fields("ItemCode") <> "9999999" Then
            Print #1, "  " & Mid(xMonthly_Xml(0), 1, 6) & XBatch & ">"
            Print #1, Space(4) & xCenbatch_Xml(0) & XBatch & xCenbatch_Xml(1)
            Print #1, Space(4) & xShopCode_Xml(0) & .Fields("ShopCode") & xShopCode_Xml(1)
            Print #1, Space(4) & xItemCode_Xml(0) & .Fields("ItemCode") & xItemCode_Xml(1)
            Print #1, Space(4) & xCurrPrice_Xml(0) & .Fields("CurrPrice") & xCurrPrice_Xml(1)
            If .Fields("CurrActn") = " " Then
               Print #1, Space(4) & Replace(xCurrActn_Xml(0), ">", xSpace_Preserve_xml) & " " & xCurrActn_Xml(1)
            Else
               Print #1, Space(4) & xCurrActn_Xml(0) & .Fields("CurrActn") & xCurrActn_Xml(1)
            End If
            Print #1, Space(4) & xPrevPrice_Xml(0) & .Fields("PrevPrice") & xPrevPrice_Xml(1)
            If .Fields("PrevActn") = " " Then
               Print #1, Space(4) & Replace(xPrevActn_Xml(0), ">", xSpace_Preserve_xml) & " " & xPrevActn_Xml(1)
            Else
               Print #1, Space(4) & xPrevActn_Xml(0) & .Fields("PrevActn") & xPrevActn_Xml(1)
            End If
            Print #1, Space(4) & xFlag_Xml(0) & "0" & xFlag_Xml(1)
            Print #1, "  " & Mid(xMonthly_Xml(1), 1, 7) & XBatch & ">"
          End If
        .MoveNext
       Loop
       .Close
     End With
     Print #1, xNewDataset_Xml(1)
```

Algorithm: Import MS-Access (VB) from XML



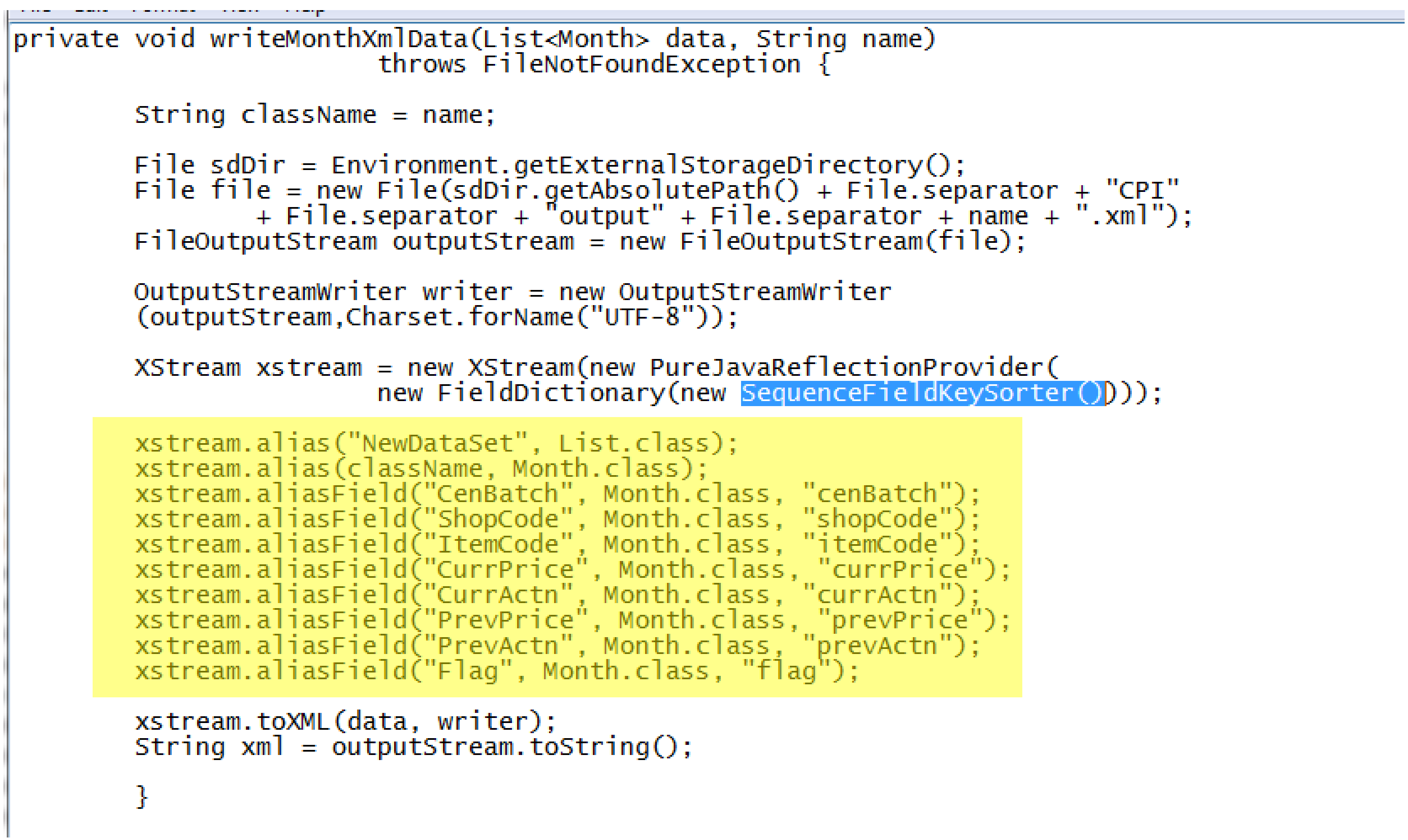
```
  '=====================================================
      dbName = xFileBatch
      xmlFile = xFile
      Set appAccess = CreateObject("Access.Application")
      appAccess.OpenCurrentDatabase (dbName)
      appAccess.ImportXML xmlFile

      appAccess.CloseCurrentDatabase
      appAccess.Quit acExit
      Set appAccess = Nothing
  '=====================================================
Select Case gsCpiType

    Case gsConsMonthly

    Set dbsBook = OpenDatabase(xFileBatch)
            If Err = 3343 Then
              Set dbsBook = Nothing
              Set dbsBook = DBEngine(0).OpenDatabase(xFileBatch) '
            End If

      'append into ori table
      strSQL = "INSERT INTO " & XBatch & " ( ShopCode, ItemCode, CurrPrice, CurrActn, PrevPrice, PrevActn ) " & _
               "SELECT MONTH" & XBatch & ".ShopCode, MONTH" & XBatch & ".ItemCode, MONTH" & XBatch & ".CurrPrice,
               "FROM MONTH" & XBatch & ";"
      dbsBook.Execute strSQL
```

Algorithm: Import SQLite (Java) from XML



```
void loadMonthFile(String fileName) {
                File sdDir = Environment.getExternalStorageDirectory();
                File file = new File(sdDir.getAbsolutePath() + File.separator + "CPI"
                                + File.separator + "data" + File.separator + fileName);

                if (checkFileLastUpdateSame(file))
                        return;
                saveLoadTimestamp(file);

                String className = (fileName.split(".xml")[0]).toUpperCase();
                Log.d(TAG, "className=" + className + ", filename=" + fileName);

                XStream xstream = new XStream();
                xstream.alias("NewDataSet", List.class);
                xstream.alias(className, Month.class);
                xstream.aliasField("CenBatch", Month.class, "cenBatch");
                xstream.aliasField("ShopCode", Month.class, "shopCode");
                xstream.aliasField("ItemCode", Month.class, "itemCode");
                xstream.aliasField("CurrPrice", Month.class, "currPrice");
                xstream.aliasField("CurrActn", Month.class, "currActn");
                xstream.aliasField("PrevPrice", Month.class, "prevPrice");
                xstream.aliasField("PrevActn", Month.class, "prevActn");
                xstream.aliasField("Flag", Month.class, "flag");

                List<Month> months = (List<Month>) xstream.fromXML(file);

                for (Month mp : months) {
                        mp.setItemCode(mp.getItemCode().trim());
                        DomainHelper.saveMonthPeriod(this, mp);
                        updateStatus(mp.toString());
                        Log.d(TAG, " * Saved: " + mp);
                }

        }
```

Algorithm: Export SQLite (Java) to XML



```
private void writeMonthXmlData(List<Month> data, String name)
                        throws FileNotFoundException {

        String className = name;

        File sdDir = Environment.getExternalStorageDirectory();
        File file = new File(sdDir.getAbsolutePath() + File.separator + "CPI"
                + File.separator + "output" + File.separator + name + ".xml");
        FileOutputStream outputStream = new FileOutputStream(file);

        OutputStreamWriter writer = new OutputStreamWriter
        (outputStream,Charset.forName("UTF-8"));

        XStream xstream = new XStream(new PureJavaReflectionProvider(
                        new FieldDictionary(new SequenceFieldKeySorter())));

        xstream.alias("NewDataSet", List.class);
        xstream.alias(className, Month.class);
        xstream.aliasField("CenBatch", Month.class, "cenBatch");
        xstream.aliasField("ShopCode", Month.class, "shopCode");
        xstream.aliasField("ItemCode", Month.class, "itemCode");
        xstream.aliasField("CurrPrice", Month.class, "currPrice");
        xstream.aliasField("CurrActn", Month.class, "currActn");
        xstream.aliasField("PrevPrice", Month.class, "prevPrice");
        xstream.aliasField("PrevActn", Month.class, "prevActn");
        xstream.aliasField("Flag", Month.class, "flag");

        xstream.toXML(data, writer);
        String xml = outputStream.toString();

        }
```

Advantages

- Machine and human readable
- Independent
- Occupy less memory space
- Simple to Understand: Records in a flat file and are separated by delimiters.
- Easy to track the error

Data Transfer Issues

- Have to create class sorter for exporting SQLite in Java Android to XML
- Using known DTD to defines the legal building blocks of an XML document

Conclusions

- The design goals of XML emphasize simplicity, generality and usability to exchange data between two databases. This approach proposed how to converts the MS-Access and SQLite to XML flat file and vice versa. In the future this can be enhanced into usability over the network simultaneously.

THANK YOU