Building Data Mining Solutions with OLE DB for DM and XML for Analysis

Zhaohui Tang, Jamie Maclennan, Peter Pyungchul Kim

Microsoft SQL Server Data Mining
One, Microsoft Way
Redmond, WA 98007
{ZhaoTang, JamieMac, PeterKim}@microsoft.com

ABSTRACT

A data mining component is included in Microsoft SQL Server 2000 and SQL Server 2005, one of the most popular DBMSs. This gives a push for data mining technologies to move from a niche towards the mainstream. Apart from a few algorithms, the main contribution of SQL Server Data Mining is the implementation of OLE DB for Data Mining. OLE DB for Data mining is an industrial standard led by Microsoft and supported by a number of ISVs. It leverages two existing relational technologies: SQL and OLE DB. It defines a SQL language for data mining based on a relational concept. More recently, Microsoft, Hyperion, SAS and a few other BI vendors formed the XML for Analysis Council. XML for Analysis covers both OLAP and Data Mining. The goal is to allow consumer applications to query various BI packages from different platforms. This paper gives an overview of OLE DB for Data Mining and XML for Analysis. It also shows how to build data mining application using these APIs.

Categories and Subject Descriptors

Data Mining Languages and Industrial Standard

General Terms

Standardization, Languages

Keywords

Data mining, database, SQL Server, OLE DB for Data Mining, XML for Analysis, Web Service.

1. Introduction

Data Mining receives more and more attention these days. Data mining, as we use the term, is the exploration and analysis, by automatic or semi-automatic means, of large quantities of data in order to discover meaningful patterns and rules. These patterns and rules will help corporations to improve their marketing, sales, and customer support operations through better understanding of their customers. Through the years, corporations have accumulated very large databases from applications such as ERP, CRM or other operational systems. People believe there is untapped value hidden inside these data; to get these patterns out requires data mining techniques.

SQL Server 2000 has introduced data mining features for the first time. Two scalable data mining algorithms are included: Microsoft Decision Trees and Microsoft Clustering. Both algorithms are developed and patented by Microsoft Research.

The Analysis Services of SQL Server 2000 and SQL Server 2005 have two components: OLAP and data mining. Both data mining and OLAP are important techniques for data analysis, but the related technologies are different, with data mining providing automatic or semi automatic pattern discovery. Data mining combines techniques developed in artificial intelligence, database and statistics, while OLAP is mainly based on SQL plus certain aggregation techniques. The term often employed in OLAP is multi-dimensional database, or a so called data cube. For example, a sales cube can be built on top of the sales table that has a number of dimensions such as Product, Region, and Time. Each cell in the cube gives the aggregated sales value for a particular product, region and time period.

Data mining and OLAP are complementary as they are both analytical tools. For example, in the customer dimension of a sales cube, there are lots of members, so it is very difficult to find customers' buying patterns. Data mining techniques can be applied to analyze this dimension to find out what are the clusters among the customers based on customer member properties and measures. SQL Server Analysis Services has advanced features that bridge data mining and OLAP, however, we will not address details of these features in this article.

2. OLE DB for Data Mining

The data mining industry today is highly fragmented, making it difficult for application software vendors and corporate developers to integrate different knowledge-discovery tools. We can consider the current data mining market similar to the database market before SQL was introduced. Every data mining vendor has its own data mining package, which does not communicate with other products. For example, a customer is interested in a decision tree algorithm from Vendor A and has built the data mining application based on Vendor A's package. Later on, the customer finds the time series algorithm from vendor B is also very attractive for prediction tasks. He now faces a difficult situation as product A and B has no common interface and he has to restart the whole project from the beginning.

Most data mining products are horizontal packages and are difficult to integrate with user applications such as customer care, CRM, ERP. With the help of the OLE DB for DM Specification, any data mining algorithms can be accessed through OLE automation, which can be easily embedded into any consumer applications.

Another problem of most commercial data mining products is that data extraction from a relational database to an intermediate storage format is necessary. Data porting and transformation are very expensive operations. Why can't we do data mining directly on relational databases where most data are stored?

To solve these problems, Microsoft initiated the OLE DB for Data Mining (DM) Specification with more than a dozen independent software vendors (ISVs) in business intelligence. Its goal is to provide an industry standard for data mining so that different data mining algorithms from various data mining ISVs can easily plug into consumer applications. Those software packages that provide data mining algorithms are called Data Mining Providers, while those applications that use data mining features are called Data Mining Consumers. OLE DB for DM specifies the common interface between Data Mining Consumers and Data Mining Providers. Figure 1 shows the basic architecture of OLE DB for Data Mining. It allows consumer applications to communicate

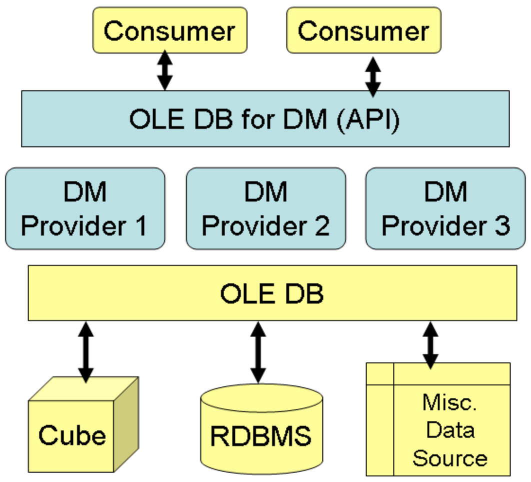

Figure 1. Architecture of OLE DB for Data Mining

with different data mining algorithm providers through the same API (SQL style).

2.1 Data Mining Model

A *data mining model* is a new concept introduced in OLE DB for DM. A data mining model can be considered as a relational table in the sense that it has a list of columns with different data types. Some of these columns are input columns while others are predictable columns. A data mining model is a container. However, a data mining model is different from a relational table as it doesn't store raw data, rather it stores the patterns that data mining algorithms have discovered in some source data. A mining model also has to specify the data mining algorithm with which it is associated and (optionally) a list of parameters. To create a data mining model, OLE DB for DM adapts the table creation syntax of SQL. The following example creates a mining model to predict credit risk level based on customer demographic information using Microsoft Decision Trees algorithm:

```
CREATE MINING MODEL CreditRisk
(
 CustomerId     long     key,
 Profession     text     discrete,
 Income          text    discrete,
 Age            long   continuous,
 RiskLevel      text   discrete  predict,
)
USING [Microsoft Decision Tree]
```

2.2 Training a Mining Model

When a data mining model is created, it is an empty container. During the training stage, the data mining algorithm analyzes the input cases and populates the patterns it has discovered to the mining model. According to OLE DB for DM Specification, training data can be from any tabular data source as long as there is a proper OLE DB driver. It does not require users to export data from a relational source to any special intermediate storage format. This largely simplifies the process of data mining. To be consistent with SQL, OLE DB for DM adopts the syntax of data insertion query. The following sample trains the CreditRisk mining model with the data stored in the customers table of a SQL Server database.

```
INSERT INTO CreditRisk
(
 CustomerId,Profession,Income, Age,RiskLevel
)
OPENROWSET('sqloledb', 'sa'; 'mypass'; '' ,
'SELECT CustomerID, Profession, Income,
             Age, Risk
 FROM Customers'
)
```

The Openrowset command can access remote data from an OLE DB data source. SQL Server 2000 ships OLE DB drivers for SQL Server, Access and Oracle. Data does not have to be loaded ahead of time. This is called *in-place mining*. The training process may take some time as during this stage, the data mining algorithm goes through all the input cases and does some complicated calculations. After training, the data mining algorithms find patterns, which are persisted inside the data mining model. Users can browse the mining model to look at the discovered patterns, or use the trained mining model for prediction tasks.

2.3 Prediction

Prediction, also known as scoring, is an important data mining task. It requires two elements: a trained data mining model and a set of new cases. The result of prediction is a new recordset that contains values for predictable columns as well as other input columns. The overall process is very similar to relational join. Instead of joining two tables, prediction joins a data mining model with an input table. Thus we introduce a new concept called a *prediction join*. The following example shows the syntax of a prediction join:

```
SELECT
  Customers.ID,
  CreditRisk.RiskLevel,
  PredictProbability(CreditRisk.RiskLevel)
FROM CreditRisk PREDICTION JOIN
       Customers
ON   CreditRisk.Profession =
     Customers.Profession AND
     CreditRisk.Income =
     Customers.Income AND
     CreditRisk.Age =
     Customers.Age
```

The query returns a record set with three columns: Customer ID, RiskLevel and Probability. OLE DB for DM predefines a set of prediction functions; most of them return additional statistics for the prediction.

Usually prediction can be done on the fly. It is also possible to do prediction on single case instead of a set of new cases. We call these prediction queries singleton queries.

OLE DB for DM has also defined a list of prediction functions that can be included in the select clause of the prediction statement. These functions will return the probability of the predicted value, histogram information about other possible values and related probabilities, top counts, cluster id, etc.

2.4 Schema Rowsets

The schema information specified in OLE DB is based on the assumption that providers support the concepts of a catalog and a schema. Schema information can be retrieved in predefined schema rowsets. Schema rowsets are global tables for meta data. In OLE DB for Data Mining, we have predefined a list of schema rowsets. These schema rowsets help applications to dynamically discover the available data mining algorithms and their parameters, existing mining models and their contents. For example, after training a mining model using decision tree algorithms, the content schema rowset contains the tree nodes information. Consumer applications can display the tree graphically based on the content schema rowset.

3. Data Mining in SQL Server

In this section, we look at the data mining component of SQL Server 2000.

3.1 Components Architecture

Microsoft has implemented an OLE DB provider for Data Mining based on the OLE DB for DM Specification. The provider includes two data mining algorithms developed by Microsoft Research. The SQL Server data mining provider is part of Analysis Services 2000 (previously called OLAP Services in SQL Server 7.0). Similar to Microsoft OLAP Services, the data mining component in SQL Server 2000 is mainly targetted to DBAs. There is no sophisticated GUI component for data mining, and Microsoft is working closely with ISV partners to build these general consumer tools. However, there are a few data mining GUI components for data mining in the Analysis Services. These components include a model creation wizard, model editor, model content browser, and DTS task for prediction. Figure 2 shows the major components in Analysis Services 2000.

3.2 Data Mining Algorithms

We now describe the data mining algorithms included with the SQL Server data mining provider: Microsoft Decision Trees and Microsoft Clustering. Decision trees are widely used for classification tasks. Unlike other classification algorithms such as nearest neighbor, neural networks, regression-based statistical techniques, decision trees can handle high dimensional data and the rules found can be more easily understood.

In order to achieve highly scalable classification over a large database, we implemented the execution module proposed in [2]. The execution module batches execution of multiple queries for the classification client in a single scan of data to compute statistics, and appropriately stages data from server to client file system and to client main memory. It also has an optimization facility to tradeoff the cost of scanning data at the server versus use of in-memory operations depending on the available client memory size.

The Microsoft clustering algorithm is a scalable implementation of the Expectation-Maximization (EM) algorithm[1]. Unlike distance-based algorithms such as K-Means, EM constructs proper statistical models of the underlying data source and naturally generalizes to cluster databases containing both discrete-valued and continuous-valued data. The scalable method is based on a decomposition of the basic statistics the algorithm needs: identifying regions of the data that are compressible and regions that must be maintained in memory. The approach operates within the confines of a limited main memory buffer and requires at most a single database scan. Data resolution

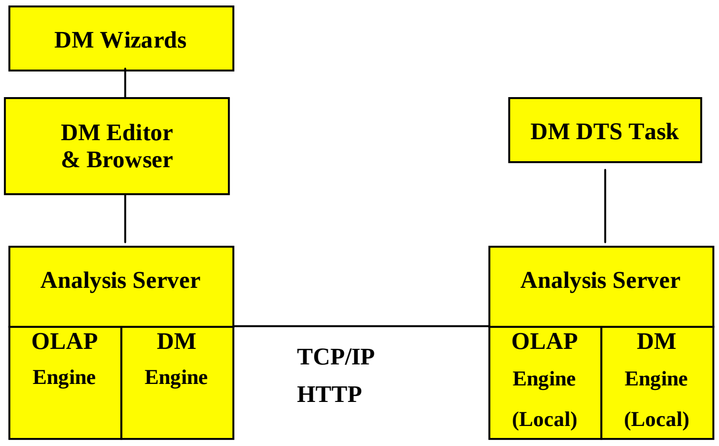

Figure 2 – SQL Server Analysis Services 2000

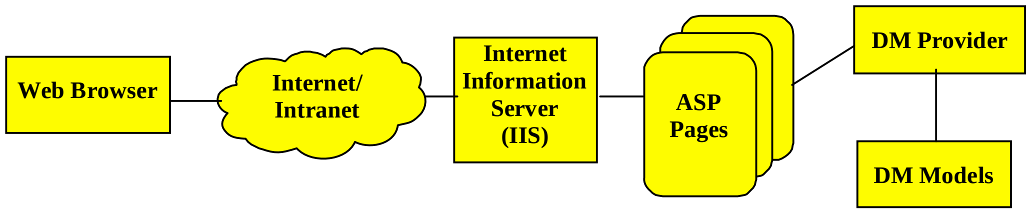

Figure 3 – Building data mining applications

is preserved to the extent possible based upon the size of the main memory buffer and the fit of the current clustering model to the data.

When there are too many attributes involved in training (e.g., more than 255 attributes by default), we apply a feature selection method to filter out less interesting attributes. The interestingness of an attribute is calculated based on the entropy of the attribute [3].

In SQL Server 2005, several more data mining algorithms are added, including time series, association, naïve bayes, sequence clustering, and neural network. It also provides an algorithm plug-in API, which allows 3rd parties to integrate their algorithms into SQL Server.

4. Building Data Mining Application using OLE DB for Data Mining

One of the biggest advantages of Microsoft Data Mining is that it is based on OLE DB for DM specification. It is very easy to use; any database developer can develop applications using data mining features. The data mining language is very similar to SQL. Microsoft data mining provider is open as it is a OLE DB component. Algorithms from other ISVs can be plugged into the same platform. Data mining services can be invoked from any consumer application through a DSO (Decision Support Object) or ADO object.

For example, a bank is developing its loan application software. It would like to add data mining to evaluate the loan risk for each applicant. The application is an n-tier intranet application. A loan assistant inputs customer information through a web form and then by clicking the submit button, the associated loan risk indication will be displayed through an Active Server Page. The architecture of the application is displayed in Figure 3:

Before building the application, the first thing to do is to create a data mining model and train the model. There are a few different ways to do this task. The easiest way is to use the mining model creation wizard of Analysis Manager. The wizard will generate data mining creation and training queries and send these queries to the Microsoft data mining provider through the OLE DB for DM interface. The other way is to write some VB or C++ code to connect to the data mining provider through the Active Data Objects (ADO) or Decision Support Objects (DSO) APIs, and then issue the text queries to the provider like a database developer does with database queries. The following example shows how to connect to the data mining provider through an ADO object in the ASP page.

```
Set con = server.CreateObject("adodb.connection")
con.open "provider=msolap; Data
Source=myserver;initial catalog=Data Mining
Database"
```

The msolap provider will then initialize the Microsoft data mining provider.

To do prediction, it should first create a prediction query in the ASP page. In our example, the loan assistant only inputs information about one loan candidate at a time, so the prediction join is on singleton cases. The query is shown as the following:

```
QueryText = “Select t.CustomerId,
CreditRisk.RiskLevel ” _
        + “From CreditRisk natural prediction join (“ _
        + “Select 100 as CustomerId, ‘Engineer’ as -
Profession, 50000 as Income, 30 as Age) as t”
```

To execute the prediction query, it is the same as doing a database query through ADO:

```
Set rs = con.Execute (QueryText)
```

The prediction will return a record set, which has two columns: CustomerId and predicted credit risk level.

Analysis Services 2000 has extended its DSO model to support data mining. DSO is the only way to create and manage models that will persist on the server. Models created through an ADO command object will exist only for the duration of a session – and are called “session mining models.” Additionally, using DSO objects allows the creation and setting of security roles and integration with the repository. However, DSO requires more coding than ADO, plus access is restricted to server administrators.

Sometimes consumer applications need to discover the capabilities of a server, the existing models on the server, or even the learned content of a mining model. Model content represents the patterns the data mining algorithm found in the dataset. The mining content for a tree model represents the tree graph. Developers can get this information through schema rowsets that describe the state of the server. The following code shows how to connect to the content schema rowset. The schema rowsets are defined in the OLE DB for DM specification.

```
const DMSCHEMA_MINING_MODEL_CONTENT =
"{3add8a76-d8b9-11d2-8d2a-00e029154fde}"
set rs = con.OpenSchema(adSchemaProviderSpecific, ,
DMSCHEMA_MINING_MODEL_CONTENT)
```

5. XML for Analysis

The programming methods described above provide an easy way to access data mining functionality from inside applications. However, these methods require certain components to exist on the client machine in order to function, namely the OLEDB client for ADO programming, or the DSO libraries for DSO programming. In circumstances where it is not convenient or possible to install the client components, developers can use XML for Analysis to communicate with their mining models.

XML for Analysis (XML/A) is a SOAP-based XML API standardizing the interaction between clients and analytical data providers. It specifies how to construct SOAP packets that can be sent to an XML/A server to discover metadata and to execute queries. The format of the result is a SOAP packet containing a rowset encoded in XML. This allows connection and interaction from any client platform without any specific client components to communicate to the server, which simplifies application deployment and permits cross-platform development.

XML/A specifies only two commands to interact with the server: *Discover* and *Execute*. The former is used to pull metadata describing the capabilities and the state of the server, and the latter is used to execute queries against the server objects.

The Discover command has the following syntax:

```
Discover (
    [in] RequestType As EnumString,
    [in] Restrictions As Restrictions,
    [in] Properties As Properties,
    [out] Result As Rowset)
```

The RequestType parameter indicates the schema that is being requested – the schemas that are supported by the provider can be accessed by first using the DISCOVER_SCHEMA_ROWSETS request type. Restrictions is an array of OLEDB-type restrictions used to limit the data returned from the server – the list of acceptable restrictions is also available from the same DISCOVER_SCHEMA_ROWSET request. Properties is a collection of properties used to control various aspects of the discover method, such as the return type. The list of supported properties can be accessed through the DISCOVER_PROPERTIES request type. Required request types and properties are specified in the XML for Analysis 1.1 specification. Finally, the Result is the tabular result of the call returned in XML format.

The following example specifies a discover call to a XML/A server requesting a list of mining models in a data base. Note that the response is restricted to those models in the "Foodmart 2000" database and the return format is specified as "Tabular."

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
">
 <SOAP-ENV:Body>
  <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle
="http://schemas.xmlsoap.org/soap/encoding/">
   <RequestType>DMSCHEMA_MINING_MODELS
</RequestType>
   <Restrictions>
    <RestrictionList>
     <CATALOG_NAME>
      FoodMart 2000
     </CATALOG_NAME>
    </RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=MSOLAP;Data Source=local;
     </DataSourceInfo>
     <Catalog>
      Foodmart 2000
     </Catalog>
     <Format>
      Tabular
     </Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following is a truncated sample response from the discover call.

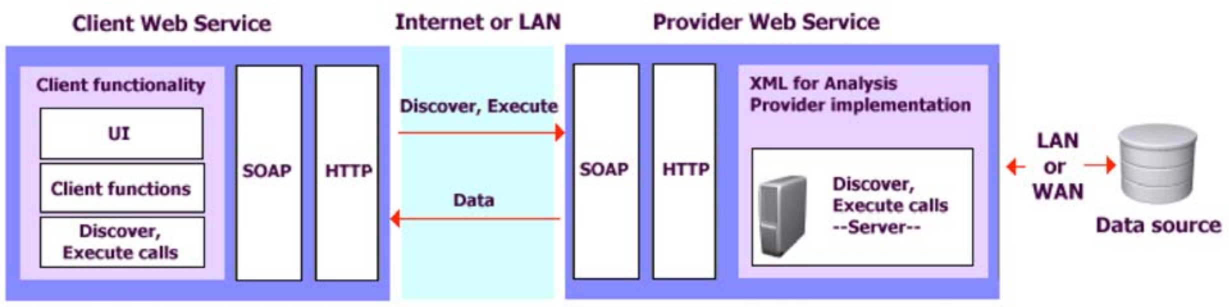

Figure 4: Architecture of XML for Analysis

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle
="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <DiscoverResponse     xmlns="urn:schemas-microsoft-com:xml-
analysis">
   <return>
    <root>
     <xsd:schema xmlns:xsd=
"http://www.w3.org/2001/XMLSchema">
      <!-- The XML schema definition of the result comes here -->
      ...
     </xsd:schema>
     <row>
      <CATALOG_NAME>FoodMart2000
</CATALOG_NAME>
      <MODEL_NAME>Sales</MODEL_NAME>
      ...
     </row>
     <row>
      <CATALOG_NAME>FoodMart2000
</CATALOG_NAME>
      <MODEL_NAME>Warehouse</MODEL_NAME>
      ...
     </row>
     ...
    </root>
   </return>
  </DiscoverResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The result of an XML/A discover request is a simple XML structure that can be interpreted on any platform using off-the-shelf XML tools.

The Execute method is very similar to the Discover method with this syntax:

```
Execute (
   [in] Command As Command,
   [in] Properties As Properties,
   [out] Result As Resultset)
```

The Command parameter specifies the query to be executed, along with any query parameters. The syntax of the query is that described in the OLEDB for Data Mining specification. The Properties parameter is identical to that of the Discover method. And, of course, the Result is the result as specified in the properties parameter.

The following example shows an Execute call of the same query we described previously using ADO.

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
"> <SOAP-ENV:Body>
 <Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle
="http://schemas.xmlsoap.org/soap/encoding/">
   <Command>
    <Statement>
Select t.CustomerId, CreditRisk.RiskLevel From CreditRisk
natural prediction join (Select 100 as CustomerId, 'Engineer' as
Profession, 50000 as Income, 30 as Age) as t
    </Statement>
   <Command>
   <Properties>
    <PropertyList>     <DataSourceInfo>Provider=MSOLAP;Data
Source=local;
     </DataSourceInfo>
     <Catalog>Foodmart 2000</Catalog>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Execute>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The resultant rowset will be formatted as in the discover call. With XML/A, the effort to develop data mining web services, especially data mining prediction, is minimal.
XML/A becomes the native communication protocol in SQL Server Analysis Services 2005.

6. Conclusion

With Analysis Services of SQL Server 2000, OLE DB for Data Mining, and XML for Analysis, ordinary database developers can create and train data mining models and embed these advanced features into consumer applications. By adding an easy and flexible programming model, data mining can now become a mass market analytical technique.

7. References

[1] Paul Bradley, Usama Fayyad, Cory Reina, Scaling EM (Expectation Maximization) Clustering to Large Databases, Microsoft Tech. Report MSR-TR-98-35, Microsoft, 1998.

[2] Surajit Chaudhuri, Usama Fayyad, Jeff Bernhardt, Scalable Classification over SQL Databases. ICDE 1999, pp. 470-479.

[3] Huan Liu, Hiroshi Motoda (ed.), Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic Publishers, 1998.

[4] Microsoft, OLE DB for Data Mining Specifications, July 2000, www.microsoft. com/data/oledb/dm.

[5] XML/A.org, XML for Analysis Specification v1.1, November 2002, http://xmla.org/docs_pub.asp