Microsoft Association Rules

Beep ... beep ... Good afternoon, sir. Did you find everything you need? ... beep ... beep ... bacon, eggs ... beep ... coffee, sugar ... beep ... milk, cookies ... ketchup, mustard, hot dogs ... beep ... cake mix ... Did you forget the frosting? I thought so! Service to aisle nine, could you bring over a can of frosting? ... Would you like any help out today, sir?

Every purchase a customer makes builds patterns about how products are purchased together. You can use these patterns to learn more about your customers’ shopping behaviors in order to optimize product layout and cross-promote the right products to the right customers.

You can find these patterns (called market basket analysis) by using the Microsoft Association Rules algorithm, as described in this chapter.

In this chapter, you will learn about the following:

- How to use the Microsoft Association Rules algorithm
- How to create Microsoft Association Rules models using DMX
- How to interpret Microsoft Association Rules mining models
- The principles of the Microsoft Association Rules algorithm
- How to tune the algorithm by using parameters

You can find examples, datasets, and projects for this chapter in its downloadable companion, Chapter11.zip, which is available on the book’s website (www.wiley.com/go/data_mining_SQL_2008/). The archive contains the following:

- A SQL Server 2008 database backup containing the data sets used in this chapter
- A file containing the DMX scripts for this chapter
- An Analysis Services project
The project uses data from the included database, as well as from the SQL Server AdventureWorksDW2008 sample database. To download and install the sample database, go to the CodePlex website at www.codeplex.com/MSFTDB-ProdSamples, select SQL Server 2008 Product Samples, and then locate and install the SQL2008.AdventureWorks_DW_BI_v2008.x86.msi package. Be sure to select the Restore AdventureWorksDW2008 (Data Warehouse) option during setup to get the sample database ready.

For the DMX examples for this chapter to function, you will need to open (in Visual Studio) and deploy the Analysis Services project included in the downloadable companion, Chapter11.zip.

**Introducing Microsoft Association Rules**

Put yourself in the role of a supermarket manager. One of your many responsibilities is to ensure that you sell the highest volume of product. Your goal is to sell more and be more profitable than your peers who are managing other stores in the chain. Understanding the purchasing patterns of your customers is the first step toward reaching this goal.

By using the Microsoft Association Rules algorithm to perform market basket analysis on your customers’ transactions, you can learn which products are commonly purchased together, and how likely a particular product is to be purchased along with another. For example, you might find that 5 percent of your customers have bought ketchup, pickles, and hot dogs together, and that 75 percent of those customers that bought ketchup and hot dogs also bought pickles. Now that you have this information, you can take action. You could change the product layout to increase sales. You could use the insight to manage stock levels. You could determine whether baskets containing pickles, hot dogs, and ketchup are more or less profitable than those without. If they’re more profitable, you could run a special to encourage this kind of shopping.

Additionally, you may want to learn more about the customers who shop at your store. With your courtesy cards and video club cards, you have collected several bits of information. You may learn that 15 percent of your female customers have video cards, and 75 percent of those customers rent their homes and live close to the store. Although it is possible to derive such patterns from standard SQL queries, you would have to write hundreds or thousands of queries to explore all possible combinations. This type of data exploration is made easier with the Microsoft Association Rules algorithm.

**Using the Association Rules Algorithm**

The Microsoft Association Rules algorithm is designed specifically for association analysis, a methodology typically used for shopping basket analysis. Given
the large size of sales databases, the Association Rules algorithm is optimized for fast training over large data sets, and this makes it an interesting choice for other problems as well.

The algorithm detects rules governing the layout of your data. A rule is a statement such as 'If it walks like a duck and quacks like a duck it is (probably) a duck'. More formally, this can be represented as the following logical proposition:

\[ P_1 \land \ldots \land P_n \land P_{n+1} \]

In the logical proposition, a set of one or more predicates \((P_i)\), when simultaneously satisfied, imply another predicate \((P_{n+1})\). Such a rule is detected by the algorithm after analyzing the training data and detecting that many (or all) the animals that walk like a duck and quack like a duck are actually ducks. A predicate is a simple condition (such as 'walks like a duck') that describes the value of one of the attributes of the objects being analyzed. In the product purchasing scenario presented at the beginning of this chapter, a predicate is the presence (or absence) of a product in a shopping basket. Therefore, "milk, cake mix ... beep..." can be interpreted as a collection of the following two predicates:

\[
\begin{align*}
\text{Milk} &= \text{Existing (in the shopping basket)} \\
\text{Cake Mix} &= \text{Existing (in the shopping basket)}
\end{align*}
\]

A rule may be discovered that says that, when Milk and Cake Mix are present, then Frosting is likely to be present as well. Such a rule will lead the clerk to suggest to customers that they buy frosting.

A predicate that participates in a rule is called an item. Consequently, a set of such predicates is called an itemset. Therefore, a rule can be described as a pair containing a left-hand itemset (the condition) and a right-hand itemset (the conclusion). Note that any rule is also a larger itemset that may appear on the left-hand side of another rule. For example, Milk, Cake Mix, and Frosting may be frequently associated with Soda.

**Data Exploration Models**

The algorithm works by counting frequent combinations of various model attributes’ states. To the extent that it counts correlations, the Association Rules algorithm is somewhat similar to the Naïve Bayes algorithm. However, the approach is quantitative (it is based on the raw number of occurrences of attribute states combinations) and not qualitative, as it is for Naïve Bayes (which computes all the conditional probabilities). Also, the correlation matrix is not completely computed (which it is for Naïve Bayes). As you will see in the “Association Rules Principles” section later in this chapter, only the significant correlations are retained.
Some of the frequent combinations (those that exceed certain probability thresholds) are strong enough to have predictive value, and are exposed by the algorithm as rules.

The similarity with Naive Bayes suggests the Association Rules algorithm as a good choice for data exploration mining models. The approach is similar — create a mining model and mark all the columns as predictable, as shown in Listing 11-1.

```sql
CREATE MINING MODEL VotingRecordsAnalysis
(
    [ID] LONG KEY,
    [Party] TEXT DISCRETE PREDICT,
    [Class Action Fairness Act] TEXT DISCRETE PREDICT,
    [Farm Security Act] TEXT DISCRETE PREDICT,
    [Highway Funding Restoration Act] TEXT DISCRETE PREDICT,
    [Homeland Security Act] TEXT DISCRETE,
    ...
) USING Microsoft_Association_Rules
```

**Listing 11-1** An Association Rules mining model intended for data exploration

Note that the Association Rules algorithm doesn’t accept continuous attributes because it is a counting engine that counts the correlations among discrete attribute states. You must make the continuous attributes in the mining model discrete or discretized.

The Association Rules viewer provides multiple data exploration options, including the following:

- A listing of frequent combinations of attribute states
- A listing of rules (the frequent combinations that have predictive values)
- The capability to browse a dependency network that intuitively displays relationships between attribute states

Figure 11-1 shows the Dependency Net view for the model defined in Listing 11-1. As you may notice, the relationships are not between model attributes, but instead are between attribute states.

**A Simple Recommendation Engine**

A recommendation engine should be able to make recommendation for items that are likely to be purchased by a customer based on previous purchases by the same customer. (In the simplest case, this may mean the current content of the shopping basket for the respective customer.) To make these
recommendations, the engine must learn frequent purchase patterns from existing sales transactions data.

Suppose that sales transactions are identified by order numbers and, for each transaction, the individual items are recorded. Additional information (such as category or brand) may be available for the products in the catalog (you’ll use this information in the next section). Listing 11-2 defines a mining structure to describe this problem space.

```
CREATE MINING STRUCTURE SalesData
{
    [Order Number] TEXT KEY,
    Products TABLE
    {
        Product TEXT KEY,
        Category TEXT DISCRETE
    }
}
```

**Listing 11-2** A mining structure describing a sales transaction table

To train the mining structure, you must provide data for both a case-level column and the nested columns, as shown in Listing 11-3. In general, this requires two tables. However, given that all you need for the case-level column is the list of distinct values of the key ([Order Number]), you can substitute a `SELECT DISTINCT` statement for one of the tables. For the `SHAPE` construct to work, note how the two data queries must return results sorted by the primary key and foreign key, respectively.
INSERT INTO [SalesData]
(
[Order Number],
[Products](SKIP, [Product], [Category])
)
SHAPE
{
OPENQUERY ([Adventure Works DW2008],
'SELECT DISTINCT [OrderNumber] AS [Order Number]
FROM dbo.[vDMPrep]
WHERE FiscalYear = ''2004''
ORDER BY [OrderNumber]')
}
APPEND
{
{
OPENQUERY ([Adventure Works DW2008],
'SELECT [OrderNumber] AS [Order Number],
[Model] AS [Product],
[EnglishProductCategoryName] AS [Category]
FROM dbo.[vDMPrep]
WHERE FiscalYear = ''2004''
ORDER BY [OrderNumber]')
}
RELATE [Order Number] To [Order Number]
}
AS [Products]

Listing 11-3 Training a mining structure from sales transactions data

Note how the [Order Number] column in the second data query is used only to relate items to the transaction, and is not actually used in the mining structure — hence the SKIP placeholder in the column mapping section of the INSERT INTO statement.

The next step is to create a mining model that will learn purchase patterns and be able to recommend purchases based on previous ones, as shown in Listing 11-4. The Products nested table (which describes the purchases for each transaction) must be both input and predictable. The model will detect products that sell together often, and make recommendations based on products already in the shopping basket.

ALTER MINING STRUCTURE SalesData
ADD MINING MODEL Recommendations

Listing 11-4 Creating and training a mining model for shopping basket analysis in the SalesData structure
Listing 11-4 (continued)

You can now use the Association Rules viewer to browse the patterns detected by the model. Furthermore, the model can produce recommendations. The statement in Listing 11-5 requests five recommendations based on a hypothetical shopping basket.

```
SELECT FLATTENED Predict(Products, 5) FROM Recommendations
NATURAL PREDICTION JOIN
{
    SELECT (
        SELECT 'Cycling Cap' AS Product UNION
        SELECT 'Sport-100' AS Product
    ) AS Products
} AS T
```

Listing 11-5 Using the Association Rules model for product recommendations

**Advanced Cross-Sales Analysis**

Association Rules models may be used to analyze the cross-sales driven by a subset of products. For example, you may want to perform such an analysis when planning a promotion for a certain product or brand. You may also want to do this when simply exploring the cross-sales potential of various product categories.

Conceptually, this is a matter of labeling certain products as input, and the others as exclusively predictable. However, because of the nested table nature of most association data sets, this is not as simple as in the case of top-level columns. New DMX features in SQL Server 2008 allow an intuitive partitioning of the items in a nested table for an analysis task such as this.
For example, assume you want to analyze the cross-sales driven by bicycle products. Bicycles always belong to the Bikes category. The goal is to build an Association Rules model that uses Bikes products as input, but recommends any other product. The statement in Listing 11-6 does exactly this.

```
ALTER MINING STRUCTURE [SalesData]
ADD MINING MODEL CategoryRecommendations
{
  [Order Number],
  Products AS [Bike Products]
  (Product
   ) WITH FILTER(Category='Bikes'),
  Products AS OtherProducts PREDICT_ONLY
  (Product
   ) WITH FILTER(Category<> 'Bikes'
  )
} USING Microsoft_Association_Rules(MINIMUM_SUPPORT=4,
  MINIMUM_PROBABILITY=0.2)
GO

INSERT INTO CategoryRecommendations
GO
```

**Listing 11-6** An Association Rules model to analyze cross-sales driven by Bike Products

The DMX statement in Listing 11-6 uses the table of products twice in the mining model. However, the first copy (under the [Bike Products] alias) considers only those products (nested table rows) that belong to one of the Bike categories, whereas the second copy considers only the other rows and is marked as PREDICT_ONLY. As a result, items that are not bikes will never appear on the right-hand side of a recommendation. Figure 11-2 shows the rules detected by such a model.

**PREDICTIVE ASSOCIATION RULES MODELS**

The Association Rules algorithm may be used as a predictive algorithm (for example, to perform a classification task). In general, this algorithm is not a good predictor, at least when compared against the other predictive algorithms in the SQL Server 2008 suite. If you decide to try this, however, one trick may help: increase the value of the MINIMUM_PROBABILITY algorithm parameter, which is discussed in detail later in this chapter. The default value, 0.4, is good for many associative tasks, but too low for classification. Bumping up the value of this parameter to at least 0.5 will improve the classification performance.
DMX

Suppose that you have two tables: Customer and Purchase. The Customer table contains customer demographic information. It includes attributes such as gender, age, marital status, profession, and so on. The Purchase table is a transaction table containing the list of movies each customer purchased in the store. There are two columns in the Purchase table: Customer_ID and Movie_Name. In this section, you build an association model to analyze the relationships among movies and demographics.

Listing 11-7 creates a model for associative analysis using Gender, Marital_Status, and the purchased movies.

```
CREATE MINING STRUCTURE MovieAssociation (  
    Customer_Id LONG KEY,  
    Gender TEXT DISCRETE,  
    Marital_Status TEXT DISCRETE,  
    MoviePurchase TABLE(    
        Movie_Name TEXT KEY    
    )  
)  
GO

ALTER MINING STRUCTURE MovieAssociation
ADD MINING MODEL MovieAssociation (  
    Customer_Id,  
)
```

Listing 11-7 Association Rules model analyzing relationships between movies and demographics
As you already know, a model training statement mainly depends on the model structure, not the algorithm on which the model is based. Listing 11-8 shows the training statement for the MovieAssociation model.

Listing 11-8 Training a mining structure containing demographics and movie purchases

After the model is processed, you can issue queries to retrieve itemsets (Listing 11-9) and rules (Listing 11-10) from the content. You do this by filtering the content on the node types for itemsets and rules, which are 7 and 8, respectively.
Using the Association Rules Algorithm

Listing 11-9 Retrieving all frequent itemsets

```
SELECT Node_Description FROM MovieAssociation.CONTENT
WHERE Node_Type = 7
```

Listing 11-10 Retrieving all rules

```
SELECT Node_Description FROM MovieAssociation.CONTENT
WHERE Node_Type = 8
```

If you have only customer demographic information and you want to give movie recommendations based on Gender and Marital_Status, you can use the prediction query shown in Listing 11-11.

```
SELECT T.CustomerID, Predict(MoviePurchase, 5) AS Recommendation
FROM MovieAssociation
NATURAL PREDICTION JOIN
OPENQUERY([Chapter 11],
    'SELECT CustomerID, Gender, Marital_Status FROM NewCustomer') AS T
```

Listing 11-11 Retrieving recommendations based on demographics

Predict(MoviePurchase, 5) returns the top five movies in a table column based on probability. This kind of prediction is called an associative prediction.

Sometimes, you not only know the customer demographics, but you also know a few movies a customer has already purchased. You can use the prediction query in Listing 11-12 to give more accurate recommendations.

```
SELECT T.CustomerID, Predict(MoviePurchase, 5) AS Recommendation
FROM MovieAssociation
PREDICTION JOIN
SHAPE {
    OPENQUERY ([Chapter 11],
        'SELECT CustomerID, Gender, [Marital Status] FROM Customers
         ORDER BY CustomerId')
}
```

Listing 11-12 Retrieving recommendations based on demographics and transaction history
APPEND
{
{
OPENQUERY ([Chapter 11],
'SELECT CustomerID, Movie FROM Movies
ORDER BY CustomerID')
}
RELATE CustomerID To CustomerID
}
AS Movie_Purchase AS T
ON
MovieAssociation.Gender = t.Gender
AND MovieAssociation.Marital_Status = t.[Marital Status]

Listing 11-12 (continued)

PREDICT, PREDICTASSOCIATION, AND PARAMETERS FOR RECOMMENDATION QUERIES

If the first argument is a predictable nested table, then the DMX Predict function is actually an alias for the PredictAssociation function. This may take a variable number of parameters and flags. Flags are included in the query in the same way that you include parameters. Their presence affects the query result, but their order does not. All flavors will take a mandatory first parameter — the name of the nested table. A possible second parameter, the number of desired recommendations, was previously discussed. Here are some flags that may be useful for your recommendation queries:

◆ EXCLUSIVE, INCLUSIVE, or INPUT ONLY — An invocation such as Predict(MoviesPurchase, 5, INCLUSIVE) may return the movies that appear in the input if they are recommended by other movies in the input. Such an invocation may be used in evaluating the accuracy of the recommendation engine. INPUT ONLY limits the results to the attributes present in the input. Although this is not very useful for a recommendation system, it is extremely useful when the nested table contains other predictable columns (for example, predicting the user rating for the movies in a shopping basket). The default behavior, EXCLUSIVE, guarantees that the list of recommendations does not contain any input.

◆ $ADJUSTEDPROBABILITY, $PROBABILITY, or $SUPPORT — This sorting criterion for recommendations determines the measure used in ranking recommendations before selecting the ones to be returned. The default is $ADJUSTEDPROBABILITY.
PREDICT, PREDICTASSOCIATION, AND PARAMETERS FOR RECOMMENDATION QUERIES (continued)

- **INCLUDE_STATISTICS** — When this flag is present, the query result includes support, probability, and adjusted probability for each recommendation.

- **INCLUDE_NODE_ID** — This is the identifier of the content node that leads to the recommendation. When this flag is present, the query result includes a new column, $NODEID, which contains the NODE_UNIQUE_NAME of the content node describing the left-hand itemset (for those recommendations derived from rules) or NULL (for recommendations based on frequent items popularity). This flag may be particularly useful when you want to identify recommendations that are derived from rules and not item popularity.

The statement in Listing 11-3 returns five recommendations, sorted by probability (and not the usual adjusted probability) with all additional information that can be extracted.

```sql
SELECT FLATTENED PredictAssociation(
    MoviePurchase,
    5,
    $PROBABILITY,
    INCLUDE_STATISTICS,
    INCLUDE_NODE_ID
) FROM MovieAssociation NATURAL PREDICTION JOIN
( SELECT 'Male' AS Gender,
    'Married' AS Marital_Status,
    ( SELECT 'Alien' AS Movie_Name UNION
    SELECT 'Raiders of the Lost Ark' AS Movie_Name)
    AS MoviePurchase) AS T
```

**Listing 11-13** Using Predict flags to get additional information about the recommendation results for a new customer

**Model Content**

Figure 11-3 shows the content of an association model. The top level has a single node that represents the model. The second level contains nodes that represent qualified itemsets and rules. The relationships between rules and itemsets are presented for a rule that recommends *Empire Strikes Back* when *Attack of the Clones* and *Return of the Jedi* are present.

For any itemset content node (identified by a value of 7 for the NODE_TYPE property), the Distribution rowset contains detailed information about the
itemsets, with each row representing an individual item. Other interesting columns of the itemset nodes include the following:

- **NODE_UNIQUE_NAME** — This is a unique content identifier for this itemset, used as a reference from the rule nodes.
- **NODE_SUPPORT** — This is the support for this itemset.
- **NODE_DISTRIBUTION** rows — Each row represents an attribute/value pair that is part of the itemset.

**Figure 11-3 Content of an association model**

For any rule content node (a value of 8 for the NODE_TYPE property), the Distribution rowset contains the predicted item on the right-hand side of the rule, and the node identifier for the itemset on the left-hand side of the rule. If you decide to write your own Association Rules browser, the following properties of a rule content node may also be useful:

- **NODE_PROBABILITY** — This is the probability of the rule represented by the current content node.
- **MSOLAP_MODEL_COLUMN** — This contains the NODE_UNIQUE_IDENTIFIER for the itemset that represents the left-hand side of the rule.
- **MSOLAP_NODE_SCORE** — This contains the rule’s importance.
- **NODE_SUPPORT** — This is the support for the rule.
- **NODE_DISTRIBUTION** — The first row is the attribute and state that are on the right-hand side of the rule.
- **NODE DISTRIBUTION** — The second row is the **NODE UNIQUE NAME** of the 1-itemset (itemset of length 1) that represents the right-hand side of the rule.

**Interpreting the Model**

After the association model is processed, you can browse the contents of the model using the Association Rules viewer. This viewer contains three tabs: Itemsets, Rules, and Dependency Network.

The Itemsets tab (shown in Figure 11-4) displays the frequent itemsets discovered by the Microsoft Association Rules algorithm. The main part of the screen is a grid that shows the list of frequent itemsets and their supports and sizes. If **Minimum Support** is set too low, this list can be quite long. The Itemsets view includes drop-down lists that enable you to filter these itemsets based on support and itemset size. You can also use the Filter Itemset drop-down option to filter the itemsets. For example, you could select only the itemsets that contain **Gender=Male**.

![Figure 11-4 Frequent itemsets](image)

<table>
<thead>
<tr>
<th>Support</th>
<th>Size</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>2439</td>
<td>1</td>
<td>Gender = Male</td>
</tr>
<tr>
<td>1957</td>
<td>1</td>
<td>Married Status = Married</td>
</tr>
<tr>
<td>1535</td>
<td>2</td>
<td>Married Status = Married, Gender = Male</td>
</tr>
<tr>
<td>1024</td>
<td>1</td>
<td>Age = 29 - 35</td>
</tr>
<tr>
<td>954</td>
<td>1</td>
<td>Age &lt; 29</td>
</tr>
<tr>
<td>845</td>
<td>1</td>
<td>Married Status = Never Married</td>
</tr>
<tr>
<td>601</td>
<td>2</td>
<td>Age = 29 - 35, Gender = Male</td>
</tr>
<tr>
<td>778</td>
<td>2</td>
<td>Age &lt; 29, Gender = Male</td>
</tr>
<tr>
<td>718</td>
<td>1</td>
<td>Age = 35 - 41</td>
</tr>
<tr>
<td>714</td>
<td>1</td>
<td>Gender = Female</td>
</tr>
<tr>
<td>711</td>
<td>2</td>
<td>Married Status = Never Married, Gender = Male</td>
</tr>
<tr>
<td>697</td>
<td>2</td>
<td>Age = 29 - 35, Married Status = Married</td>
</tr>
<tr>
<td>588</td>
<td>3</td>
<td>Age = 29 - 35, Married Status = Married, Gender = Male</td>
</tr>
<tr>
<td>560</td>
<td>2</td>
<td>Married Status = Never Married, Age &lt; 29</td>
</tr>
<tr>
<td>527</td>
<td>2</td>
<td>Age = 35 - 41, Married Status = Married</td>
</tr>
<tr>
<td>625</td>
<td>2</td>
<td>Age = 35 - 41, Gender = Male</td>
</tr>
<tr>
<td>523</td>
<td>1</td>
<td>Star wars = Existing</td>
</tr>
<tr>
<td>466</td>
<td>1</td>
<td>Metric, The = Existing</td>
</tr>
</tbody>
</table>
The filters are actually regular expressions—the itemsets that match the regular expression are included in the Itemsets report. The language used is the .NET Framework Regular Expression Language, which is documented on the MSDN library (available at msdn.microsoft.com). The regular expression language allows more advanced filters. An expression such as .*Godfather.* will return all itemsets that include one of the movies in the Godfather series.

The Rules tab (shown in Figure 11-5) displays the qualified association rules. The main part of the tab is the rule grid. It displays all qualified rules, their probabilities, and their importance scores. The importance score is designed to measure the usefulness of a rule. The higher the importance score, the better the quality of the rule is. Similar to the Itemsets tab, the Rules tab contains some drop-down lists and text files for filtering rules. For example, you can select all rules that contain Gender=Male on the right side.

![Figure 11-5 Association rules](image)

The third tab of the association is the Dependency Net view (shown in Figure 11-6). As discussed in the “Data Exploration” section earlier in this chapter, each node in this view represents an item (for example, StarWars = Existing or Gender = Male). Each edge represents a pairwise association rule. The slider is associated with the importance score. By default, it displays up to 60 nodes. You may add hidden nodes to the graph using the Search.
button in the toolbar. You can also filter out the weak edges using the slider. If you want to show more nodes and edges in the Dependency Net view, you can lower the value of Minimum_Probability and reprocess the model.

![Figure 11-6 Dependency Net view](image)

**Association Algorithm Principles**

An association algorithm is nothing more than a correlation counting engine. The Microsoft Association Rules algorithm belongs to the Apriori association family, which is a very popular and efficient algorithm for finding frequent itemsets (common attribute value sets). There are two steps in the Microsoft Association Rules algorithm, as illustrated in Figure 11-7. The first step of the algorithm, a calculation-intensive phase, is to find frequent itemsets. The second step is to generate association rules based on frequent itemsets. This step requires much less time than the first step does.

**Understanding Basic Association Algorithm Terms and Concepts**

The following sections define the basic terms and association-algorithm concepts you will need to understand before implementing the Microsoft Association Rules algorithm principles.
Finding frequent itemsets

<table>
<thead>
<tr>
<th>Tid</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beer, diaper, cake</td>
</tr>
<tr>
<td>2</td>
<td>beer, bread, milk</td>
</tr>
<tr>
<td>3</td>
<td>cake, pepsi, milk</td>
</tr>
<tr>
<td>4</td>
<td>cheese, ham</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Frequent Itemsets Support

<table>
<thead>
<tr>
<th>Frequent Itemsets</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>beer, diaper</td>
<td>3.0%</td>
</tr>
<tr>
<td>cake, pepsi, milk</td>
<td>2.5%</td>
</tr>
<tr>
<td>milk, bread</td>
<td>2.0%</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Finding rules

<table>
<thead>
<tr>
<th>Probability</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>75.0%</td>
<td>beer =&gt; diaper</td>
</tr>
<tr>
<td>65.8%</td>
<td>cake, pepsi =&gt; milk</td>
</tr>
<tr>
<td>63.5%</td>
<td>ham =&gt; cake</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Figure 11-7 The two-step process of the Microsoft Association Rules algorithm

**Itemset**

An *itemset* is a set of items. Each item is an attribute value. In the case of market basket analysis, an itemset would contain a set of products such as cake, Pepsi, and milk. In the case of customer demographic exploration, an itemset would contain a set of attribute values such as \(\{\text{Gender} = \text{‘Male’}, \text{Education} = \text{‘Bachelor’}\}\). Each itemset has a size, which is the number of items contained in the itemset. The size of itemset \(\{\text{Cake, Pepsi, Milk}\}\) is 3.

*Frequent itemsets* are those itemsets that are relatively popular in the data set. The popularity threshold for an itemset is defined using *support*, which is discussed in the next section.

**NOTE** To be more precise, cake, Pepsi, and milk are all attributes. Their values are binary: Existing or Missing. For simplicity, you use \(\{\text{Cake, Pepsi, Milk}\}\) to denote \(\{\text{Cake} = \text{Existing, Pepsi} = \text{Existing, and Milk} = \text{Existing}\}\).

**Support**

*Support* is used to measure the popularity of an itemset. Support of an itemset \(\{A, B\}\) is made up of the total number of transactions that contain both \(A\) and \(B\), and is defined as follows:

\[
\text{Support} (\{A, B\}) = \text{NumberofTransactions}(A, B)
\]
Understanding Basic Association Algorithm Terms and Concepts

Minimum Support is a threshold parameter you can specify before processing an association model. It means that you are interested in only the itemsets and rules that represent at least minimum support of the data set. The parameter Minimum Support is used to restrict the itemset, but not rules.

**NOTE** Minimum Support represents the number of cases for the frequency threshold of an itemset. However, many people find it handy to have a percentage value instead of actual counts for this parameter. For example, Minimum Support = 0.03 means that the threshold for support is 3 percent. In the Microsoft Association Rules algorithm, if a user specifies this parameter with an integer number, the algorithm considers the actual case count to be the threshold. If a user inputs a floating number (less than 1.0) for this parameter, the algorithm considers it the percentage threshold.

**Probability (Confidence)**

*Probability* is a property of an association rule. The probability of a rule $A \Rightarrow B$ is calculated using the support of itemset $(A, B)$ divided by the support of $(A)$. This probability is also called *confidence* in the data mining research community.

Probability is defined as follows:

$$\text{Probability (}A \Rightarrow B\text{)} = \text{Probability (}B|A\text{)} = \text{Support (}A, B\text{)}/ \text{Support (}A\text{)}$$

Minimum Probability is a threshold parameter you can specify before running the algorithm. It means that the user is interested in only the rules that have a high probability, rather than a minimum probability. Minimum Probability has no impact on itemsets, but it does impact rules.

**NOTE** As you learned in the previous section, the popularity of itemsets is measured by their Support. However, an itemset probability can be defined as below (although it cannot be used as a threshold with the Microsoft Association Rules algorithm):

$$\text{Probability (}\{A, B\}\text{)} = \text{NumberOfTransactions (}A, B\text{)}/ \text{TotalNumberOfTransactions}$$

**Importance**

*Importance* is also called the *interesting score* (or the *lift* in some literature). Importance can be used to measure itemsets and rules.

The importance of an itemset is defined using the following formula:

$$\text{Importance (}\{A, B\}\text{)} = \text{Probability (}A, B\text{)}/(\text{Probability (}A\text{)}* \text{Probability (}B\text{)})$$
If \( \text{importance} = 1 \), A and B are independent items. It means that the purchase of product A and the purchase of product B are two independent events. If \( \text{importance} < 1 \), A and B are negatively correlated, which means that if a customer buys A, it is unlikely he or she will also buy B. If \( \text{importance} > 1 \), A and B are positively correlated, which means that if a customer buys A, it is very likely he or she also buys B.

For rules, the importance is calculated using the following formula:

\[
\text{Importance} (A \Rightarrow B) = \log \left( \frac{p(B|A)}{p(B|\text{not } A)} \right)
\]

An importance of 0 means that there is no association between A and B. A positive importance score means that the probability of B goes up when A is true. A negative importance score means that the probability of B goes down when A is true.

Table 11-1 gives the correlation counts of donut and muffin derived from a purchase database. Each cell value represents the number of transactions. For example, 15 out of 100 transactions include a customer purchasing both donuts and muffins.

<table>
<thead>
<tr>
<th>DONUT</th>
<th>NOT DONUT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muffin</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Not muffin</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>90</td>
<td>10</td>
</tr>
</tbody>
</table>

In the following calculations, the previous definitions are used to determine the support, probability, and importance of related itemsets and rules for Donut and Muffin:

- \( \text{Support} \{(\text{Donut})\} = 90 \)
- \( \text{Support} \{(\text{Muffin})\} = 20 \)
- \( \text{Support} \{(\text{Donut}, \text{Muffin})\} = 15 \)
- \( \text{Probability} \{(\text{Donut})\} = \frac{90}{100} = 0.9 \)
- \( \text{Probability} \{(\text{Muffin})\} = \frac{20}{100} = 0.2 \)
- \( \text{Probability} \{(\text{Donut}, \text{Muffin})\} = \frac{15}{100} = 0.15 \)
- \( \text{Probability} \{(\text{Donut}|\text{Muffin})\} = \frac{15}{20} = 0.75 \)
- \( \text{Probability} \{(\text{Muffin}|\text{Donut})\} = \frac{15}{90} = 0.167 \)
- \( \text{Importance} \{(\text{Donut}, \text{Muffin})\} = \frac{0.15}{(0.2 \times 0.9)} = 0.833 \)

The rule importance formula used here may lead to calculation errors if either of the conditional probabilities is 0, which is likely to happen if two items are perfectly correlated. To avoid this issue, all the counts used in computing
conditional probabilities are incremented with 1. This alteration has no impact on the relative importance of the rules, particularly for rules supported by many training cases. It has the advantage of providing a uniform treatment for all rules detected by the system. The altered correlation numbers used in computing the rules’ importance are presented in Table 11-2.

### Table 11-2 Altered Correlation Count for Donut and Muffin, Used in Computing the Rules’ Importance

<table>
<thead>
<tr>
<th></th>
<th>DONUT</th>
<th>NOT DONUT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muffin</td>
<td>15 + 1 = 16</td>
<td>5 + 1 = 6</td>
<td>22</td>
</tr>
<tr>
<td>Not muffin</td>
<td>75 + 1 = 76</td>
<td>5 + 1 = 6</td>
<td>82</td>
</tr>
<tr>
<td>Total</td>
<td>92</td>
<td>12</td>
<td>104</td>
</tr>
</tbody>
</table>

Using the altered counts, the rules’ importance is computed as shown here:

\[
\text{Importance (Muffin=>Donut)} = \log_{10}(\frac{\text{Probability}(\text{Donut|Muffin})}{\text{Probability}(\text{Donut|Not Muffin})}) = \log_{10}(\frac{16}{22} / \frac{76}{82}) = -0.105302438
\]

\[
\text{Importance(Donut=>Muffin)} = \log_{10}(\frac{\text{Probability}(\text{Muffin|Donut})}{\text{Probability}(\text{Muffin|Not Donut})}) = \log_{10}(\frac{16}{92} / \frac{6}{12}) = -0.45864
\]

From the importance of the itemset \{Donut, Muffin\}, you can see that Donut and Muffin are negatively correlated — it is rather unlikely for someone who buys a donut to also buy a muffin.

### Finding Frequent Itemsets

Finding frequent itemsets is the core part of using the Microsoft Association algorithm. First, you must specify the frequency threshold using the Minimum Support parameter (for example, Minimum Support = 2%). This means that you are interested in analyzing only the items that appear in at least 2 percent of all shopping baskets.

The algorithm finds all frequent itemsets with size = 1 in the first iteration (the popular products with support greater than Minimum Support). The algorithm does this by scanning the data set and counting the support of each item. The second iteration finds the frequent itemsets of size = 2. Before starting the second iteration, the algorithm generates a set of candidate itemsets of size 2 based on the result of first iteration (frequent itemsets of size = 1). Again, the algorithm scans the data set and counts the supports for each generated candidate itemset. At the end of the iteration, it selects the candidates with support greater than or equal to Minimum Support to get the list of frequent itemsets with sizes equal to 2.
The algorithm repeats the same procedure to find frequent itemsets with sizes 3, 4, 5, and so on, until no more itemsets meet the Minimum_Support criteria.

Figure 11-8 illustrates the process of identifying frequent itemsets. The Minimum_Support is set to 250/1000. At the first iteration, cheese and cake are filtered out. At the second iteration, the candidate \( \{ \text{diaper, milk} \} \) is disqualified. At the third iteration, the candidate \( \{ \text{beer, diaper, bread} \} \) has enough support, whereas the candidate \( \{ \text{beer, milk, bread} \} \) is filtered out.

The following pseudocode is the main procedure for generating frequent itemsets:

```java
SetOfItems generateFrequentItemsets(Integer minimumSupport) {
    F[1] = {frequent items};
    for (k =1, F[k] >0; k++) {
        C[k+1] = generateCandidates(k, F[k]);
        for each transaction t in databases {
            For each candidate c in C[k+1] {
                if t contains c then c.count++
            }
        } //Scan the dataset.
        for each candidate c in C[k+1] {
```
//Select the qualified candidates
if c.count >= Minimum_Support F[k+1] = F[k+1] U {c}
}

//Union all frequent itemsets of different size
while k>=1 do {
  F = F U F[k];
  k--;
}
return F;

After you have your frequent itemsets, the generateCandidates function returns all of the candidate itemsets with size = k + 1. Every subset of a frequent itemset must itself be a frequent itemset as well. For example, if \{beer, diaper, bread\} is a frequent itemset, then \{beer\}, \{diaper\}, \{bread\}, \{beer, diaper\}, \{beer, bread\}, and \{diaper, bread\} must also be frequent itemsets.

To generate candidate itemsets C_{k+1} from frequent itemsets F_k, you use the following SQL join statement:

```
Insert into C_{k+1}
Select x1.a1, x1.a2, ..., x1.ak, x2.ak
From F_k as x1, F_k as x2
Where
  //match the itemset prefixes of size k-1
  x1.a1 = x2.a1 And
  x1.a2 = x2.a2 And
  ...
  x1.ak-1 = x2.ak-1 And
  //avoid duplicates
  x1.ak <> x2.ak
```

This SQL statement generates candidate itemsets of size k having prefixes of itemsets size k-1. However, it doesn’t guarantee that all the subsets of candidate itemsets are frequent itemsets. So, you must prune the candidates containing infrequent subsets by using the following procedure:

```
Boolean hasInfrequentSubset(Itemset c, SetofItemsets F) {
  For each (k-1) subset s of c {
    If s not in F then return true;
  }
  return false;
}
```

The generation of candidate itemsets and the counting of their correlation are time-consuming tasks. In some cases, this can generate a huge number of candidate sets. For example, suppose that there are 10,000 products (a medium-sized supermarket). If the minimum support is low enough, the
algorithm will generate up to $10^8$ candidate 2 itemsets. Many optimization techniques are available in this phase. For example, the Microsoft Association Rules algorithm stores the itemsets in a tree data structure to save space.

Some association algorithms generate frequent itemsets without any candidate generation.

### FACTORS AFFECTING EFFICIENT PROCESSING

Association algorithm processing is very sensitive to the Minimum Support parameter. When its value is set too low (less than 1 percent), the processing time and required memory become exponential. This is because of the large number of qualified frequent itemsets and frequent itemset candidates.

For large data sets with lots of distinct items, you should avoid setting this parameter too small.

The number of items is also critical to the performance of the processing. When there are too many unique items, consider grouping them into categories. For example, your store may have a dozen different jelly beans. You could group them all into a single JellyBeans category, which will not only reduce the total number of items, but also the model processing time.

### Generating Association Rules

The next step in the association algorithm process is to generate association rules. You’re looking for rules of the form cake $\rightarrow$ milk, or milk $\rightarrow$ cake and you’re interested in rules that have a high correlation. To generate these rules, you need the count for the \{cake, milk\} itemset, as well as the counts for cake and milk (the 1-itemsets). In general, you need the itemsets to the left of the arrow (the left-hand side), along with the itemset that includes all items in the rule.

As rules are generated from the itemset, each item in the rule automatically satisfies the minimum support condition. The following procedure generates all of the qualified association rules:

For each frequent itemset $f$, generate all the subset $x$ and its complementary set $y = f - x$

If $\text{Support}(f) / \text{Support}(x) > \text{Minimum Probability}$, then $x => y$ is a qualified association rule with probability $= \text{Support}(f) / \text{Support}(x)$

### NOTE

The Microsoft Association Rules algorithm doesn’t generate multiple items on the right side of the rule. However, if you want to have multiple recommendations, you can use a prediction query against an association model, which can return multiple items.
Prediction

In an association model, if a column is used for input, its values can be used only in frequent itemsets and on the left side of association rules. If a column is used to make predictions, the column’s states can be used in frequent itemsets and on the left and right sides of the association rules. If a column is predict_only, its states can appear in frequent itemsets and on the right side of rules.

Many association algorithms in commercial data mining packages stop at finding itemsets and rules. The Microsoft Association Rules algorithm can perform predictions using these rules. The results of the predictions are usually a set of items to recommend.

You can build an association model not only based on shopping baskets, but also based on customer demographics. For example, you can include gender, marital status, and home ownership as case-level attributes in the mining structure, and include the shopping basket as a nested table in the same structure. In this case, you analyze the shopping patterns not only based on the relationship of itemsets, but also based on the demographics. For example, you may find a rule that predicts that 65 percent of male customers who purchase beer also purchase diapers in the same transaction, and that 20 percent of male customers who purchase diapers also purchase wine.

These rules can be applied for prediction. For a male customer, you may recommend a list of wines. If a male customer has already bought beer in the shopping cart, you may recommend both wine and diapers.

However, not every itemset is associated with a rule. For example, there is no rule that has the itemset \{beer, diaper, bread, milk\} on the left side. What would the recommendation list be for a customer who bought beer, diapers, bread, and milk? Here is the method the Microsoft Association Rules algorithm uses to execute associative prediction:

1. Given a list of items, find all rules with the left side matching the given items, or any subsets of the given items. Apply those rules to get the list of recommendations.
2. If there is no appropriate rule, or there are too few recommended items, apply marginal statistics to predict and return the \(n\) most popular items.
3. Sort the items from steps 1 and 2 based on probability.

**Note** The number of qualified association rules is based on the parameter Minimum_Probability. (Of course, each item in a rule must be a frequent item.) For example, when Minimum_Probability is set to 30 percent, this means 30 percent of customers who purchase A also purchase B \((A \rightarrow B)\). This is a qualified rule. Rule generation is a relatively fast process, and you may lower the
probability to have more rules. In a sparse data set like the shopping transaction table, you may set Minimum_Probability to 5–10 percent and get reasonable rules. In a dense data set like a customer demographic table, you need to raise this parameter to 40–50 percent; otherwise, you may get contradictory rules (for example, High IQ => Gender = Male and High IQ => Gender = Female).

Algorithm Parameters

As indicated throughout this chapter, the association algorithm is very sensitive to the algorithm parameter settings. This section outlines parameters for the Microsoft Association Rules algorithm.

**MINIMUM_SUPPORT**

Minimum_Support is a threshold parameter. It defines the minimum support requirement that items must meet to qualify as a frequent itemset. Its value is within the range of 0 to 1. If this value is set too low (for example, 0.001), the algorithm may take much longer to process and require much more memory. The default value is 0, and the algorithm uses a heuristic to determine a good minimum support threshold.

If Minimum_Support is set to more than 1, it is considered to be the threshold for the number of cases instead of a percentage.

**MAXIMUM_SUPPORT**

Maximum_Support is a threshold parameter. It defines the maximum support threshold of a frequent itemset. Its value is within the range of 0 to 1. The default value is 0.03. This parameter can be used to filter out those items that are too frequent.

If Maximum_Support is set to more than 1, it is considered to be the threshold for the number of cases instead of a percentage.

**MINIMUM_PROBABILITY**

Minimum_Probability is a threshold parameter. It defines the minimum probability for an association rule. Its value is within the range of 0 to 1. The default value is 0.4.

**MINIMUM_IMPORTANCE**

Minimum_Importance is a threshold parameter for association rules. Rules with importance less than Minimum_Importance are filtered out.
MAXIMUM_ITEMSET_SIZE

Maximum_Itemset_Size specifies the maximum size of an itemset. The default value is 0, which means that there is no size limit on the itemset. Reducing the maximum itemset size reduces the processing time because the algorithm can save further iterations over the data set when the candidate itemset size reaches this limit.

MINIMUM_ITEMSET_SIZE

Minimum_Itemset_Size specifies the minimum size of the itemset. The default value is 0. However, you may not always care about the smaller itemsets. For example, you may be interested only in itemsets with sizes greater than 4. Reducing Minimum_Itemset_Size will not reduce the processing time because the algorithm has to start with itemset size 1 and increase the size step by step.

MAXIMUM_ITEMSET_COUNT

Maximum_Itemset_Count defines the maximum number of itemsets. If this is not specified, the algorithm generates all itemsets based on Minimum_Support. The Maximum_Itemset_Count parameter avoids generating a large number of itemsets. When there are too many itemsets, the algorithm will keep only the top n itemsets based on their importance scores.

OPTIMIZED_PREDICTION_COUNT

Optimized_Prediction_Count defines the number of items to be cached to optimized predictions. The default value of this parameter is 0, meaning that the algorithm will produce as many predictions as requested in the query. Models having this parameter set to a value larger than 0 will expose better prediction performance. However, for those models, queries will return at most the number of predictions specified by the parameter value.

AUTODETECT_MINIMUM_SUPPORT

This parameter represents the sensitivity of the algorithm used to autodetect minimum support. Setting this value to 1.0 will cause the algorithm to automatically detect the smallest appropriate value of minimum support. Setting this value to 0 turns off autodetection, and the algorithm operates on the actual value of minimum support. This parameter is used only when MINIMUM_SUPPORT is set to 0.0.
Summary

This chapter provided you with an overview of the Microsoft Association Rules algorithm and its main usages. You learned the key terms of association algorithms, including itemset, rule, support, probability, and importance. The chapter also taught you the principles of association algorithm processing. There are two steps in this algorithm: identifying frequent itemsets and generating rules. Rules can be used for prediction.

You also learned the DMX queries to use with the association model. These queries generate recommendations based on probabilities or adjusted probabilities. The results of these queries can be used in cross-selling applications.

By now, you should be able to do market basket analysis and advanced data exploration using the Microsoft Association Rules algorithm.

In Chapter 12, you will learn about the Microsoft Neural Network algorithm and its close relative, Microsoft Logistic Regression — two of the most powerful predictive algorithms included in SQL Server Data Mining.