SQL SERVER 2005

Fuzzy Lookups and Groupings Provide Powerful Data Cleansing Capabilities

Jay Nathan

This article discusses:

- SQL Server Integration Services
- SQL Server 2005 Fuzzy Lookup and Grouping technology
- Data-cleansing techniques

This article uses the following technologies:
SQL Server 2005

Data cleansing is an important task for data warehouse specialists, database administrators, and developers alike. Deduplication, validation, and householding techniques can be applied whether you are populating data warehouse dimensions, integrating new data into an existing transactional system, or supporting real time dedupe efforts within a transactional system. The goal is a high level of data accuracy and reliability that translates into better customer service, lower costs, and peace of mind. Data is a valuable organizational asset that should be cultivated and refined to realize its full benefit.

To help with this, the SQL Server™ 2005 database platform offers a completely redesigned SQL Server Integration Services (SSIS) engine. Formerly known as Data Transformation Services (DTS), SSIS includes many new features to aid data architects and extract, transform, and load (ETL) developers. One noteworthy new feature is built-in support for Fuzzy Lookups and Fuzzy Groupings. This is a powerful data-cleansing solution. I will provide an overview of fuzzy searching techniques and a dissection of the underlying fuzzy search technology implemented in SQL Server 2005. I'll place a particular emphasis on real-world scenarios.

Data Cleansing

Data-cleansing techniques come in several forms including deduplication, validation, and householding. Because of limitations in the way many transactional systems gather and store data, these practices become a necessary part of providing accurate information back to the business users.

Deduplication ensures that one accurate record exists for each business entity represented in a corporate transactional or analytic database. Validation ensures that each attribute maintained for a particular record is correct. Addresses are a good candidate for validation procedures where cleanup and conformation procedures are performed. Householding is the technique of grouping individual customers by the household or organization of which they are a member. This technique has some interesting marketing implications, and can also support cost-saving measures of direct advertising.

Cleansing data before it is stored in a reporting database is necessary to provide value to consumers of business intelligence applications. The cleansing process usually includes deduping processes that prevent duplicate records from being reported by the system.

Suppose that I am a customer of a life insurance company that has a sister company from whom I also buy disability insurance. That I am an individual who buys life and disability insurance through related companies should be an interesting fact to either company's actuarial department. Since the sister companies are separate legal entities, they do not necessarily share transactional systems and most likely do not even realize that my information exists in both of their databases.

Many months after I buy my policies, a business intelligence consulting firm is hired to create a data warehouse that combines key customer and policy data from both companies' systems into one unified data store. The consultants quickly determine that customer information is commonly duplicated on the systems. They must come up with a way to combine this information to leave only one conformed customer record in the database for each customer across both companies. The resulting customer record must represent the latest and greatest information as it was captured by either system. And to make matters more confusing, each sister company's transactional system was implemented independently with varying levels of data consistency validation and enforcement.

Common data inconsistencies include misspellings, inconsistent abbreviation usage, and other free-form text input anomalies. Consider the duplicate records shown in **Figure 1**. While these records describe the same customer to human eyes, automating a custom process to recognize this fact is much more difficult.

Traditional Approaches

Any number of techniques and tools can be employed to handle these kinds of situations. Specialized structured query language constructs such as the T-SQL LIKE and CONTAINS clauses can be used for basic wildcard searches. But LIKE queries are limited in their ability to handle misspellings, and CONTAINS queries are used in conjunction with SQL Server Full Text indexing.

Fuzzy search databases can be amassed that compile common misspellings (or variants) of specific words which can then be substituted during the cleansing process. This technique works better for applications that check one word at a time, like Microsoft® Word, which employs a similar technique for making spelling corrections on the fly.

Phonetic matching algorithms, implemented in SQL Server as SOUNDEX queries, also detect similarities between single words by matching prominent phonetic characteristics that are then scored numerically for comparison. Key drawbacks to SOUNDEX are that the input string must be contiguous with no spaces, and if the first character of the input string is not correct, the probability of a match being made drops down to zero.

Error Tolerant Index

While all of these traditional approaches and techniques are certainly an option with SQL Server 2005, there is a key new tool in the data cleansing and ETL arsenal called the error tolerant index (ETI). The ETI is a decomposition of the field values contained within a reference table of values into smaller tokens. After an ETI has been generated (also known as a match index), the SSIS Fuzzy Lookup transformation consumes it along with input records that are being tested for similarity to the records in the reference table. Rather than comparing the fields of the reference and input records on a one-to-one or exact match basis, input records are scanned to determine whether they contain smaller subsets of reference table values known as tokens, providing a much finer comparison. For example, instead of searching for a street address that contains the value "112 Sunny Vail Ln.", smaller components of the reference value might be used, such as "sunn", "nyva", and "112".

The SSIS Fuzzy Lookup and Fuzzy Grouping transformations are based on a technology that extends complex string-matching functionality. Developers apply domain-specific logic to fuzzy searches in the form of a multi-column weighting scheme. As you can probably guess, this level of search granularity is not achieved without some performance tradeoffs, which I'll discuss later.

The indexing process of a Fuzzy Lookup transformation creates tokens by splitting a reference table's field values based on a set of various delimiters that include the following characters: ,.;:-"'&/\@!?()<>[]{}|#*^%, as well as the space, tab, line feed, and carriage return characters. These default delimiters can be adjusted by the developer if necessary. After individual tokens are extracted from the reference data, they are further decomposed into smaller tokens by the indexing process and stored to the match index. **Figure 2** demonstrates how an address reference field value may be broken down into tokens during the indexing process.

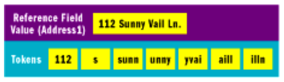

Figure 2 Generating Sample Tokens Reference Field Value

The main difference between error-tolerant match index searches and other fuzzy search techniques is that each input record is evaluated against subsets of characters that make up the strings found in the database fields. Rather than a pass or fail inner-join test, the input record is assigned a score based on the overall set of matching tokens found in the reference table for each input record. Searching at the substring level in combination with a sophisticated character-distance-searching algorithm is what makes the match index fault-tolerant. As you can see, a mechanism exists that can power through the misspellings and other nonstandard data found in transactional systems. The similarity function also uses "term frequency/inverse document frequency" (TF-IDF) measures for assessing the importance of a token. For example, if the reference table contains A. Datum Corporation, WingTip Toys Corporation, and Adventure Works Corporation, then the token "Corporation" will have less importance than other tokens for matching purposes.

SQL Server Integration Services Basics

Before going any further, let's discuss the new features of the Business Intelligence Development Studio in SQL Server 2005. Anyone who has used Visual Studio® .NET will find this development environment fairly comfortable. Inside the Business Intelligence Development Studio, you create data transformation projects as well as Analysis Services and Reporting Services projects.

Data transformation projects contain SSIS packages, which are now individual .dtsx files. Each SSIS package that is part of a project will have its own .dtsx file under the SSIS Packages node of the Solution Explorer. All of the connections needed for use within the project are created under the Data Sources node as .ds files.

In SSIS, each SSIS Package is now divided into several main areas: the Control Flow tab represents the overall control logic of a SSIS package; the Data Flow tab actually contains the logic behind individual data flows within the package; Event Handlers allow the developer to define actions to be triggered from specific SSIS engine events raised during package execution (such as OnError, OnPreExecute, and OnTaskFailed); and the Package Explorer tab provides access to variables and metadata for the SSIS package. The Data Flow tab toolbox contains a robust new set of transformations, which is where the Fuzzy Lookup and Fuzzy Grouping tasks can be found.

A Data Warehousing Scenario

In a data warehousing scenario, the customer dimension typically contains the latest and greatest information for each customer, no matter how many databases the customer's data is stored in. Revisiting the insurance policy example, where customer information is held in two mutually exclusive databases, the data warehouse ETL developer must merge sources of customer data together at the parent-company level (see **Figure 3**). This means that initially two records will exist in the staging area (partitioned by a source identifier) for individuals who hold both disability and life insurance policies with the sister insurance companies.

Each time customers are loaded from the transactional source systems, a Fuzzy Lookup is used in tandem with a regular exact match Lookup transformation to ensure that data is merged for customers who already exist within the warehouse, and that only new customer records are inserted into the customer dimension. Since I consider the data warehouse dimensions a pristine, accurate, and conformed view of customers across corporate systems, this serves as the reference table for the Fuzzy Lookup operation to use. The diagram in **Figure 4** depicts the typical flow of data through the Fuzzy Lookup transformation within an SSIS package flow.

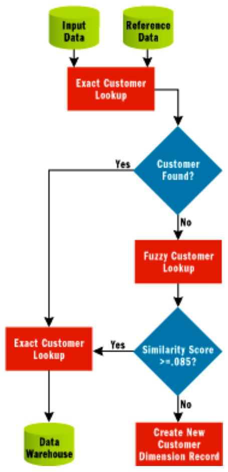

Figure 4 Data Flow via a Lookup App

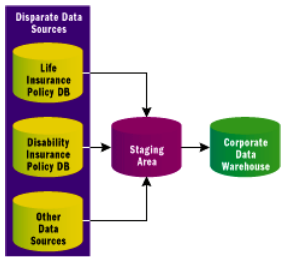

Figure 3 Multisource Data Warehouse Scenario

Fuzzy Search Architecture in SSIS

The match index is a combination of two tables created by the fuzzy transformation task. One of the tables is simply a copy of the reference table with an additional column called RID attached. The RID, or reference ID, column is an integer key added to the copy of the reference table. This field is referenced by another table that stores the individual tokens generated by the Fuzzy Lookup processing task. Each row in this other table contains a string token value, a column index corresponding to a column in the reference table, and a column filled with a set of reference IDs that contain this token in the specified column.

Fuzzy search transformations come in two types, Fuzzy Lookup and Fuzzy Grouping. As you know, the Fuzzy Lookup transformation uses a reference table to create a match index against which input records are evaluated and scored. The fields chosen for Fuzzy Lookup transformations are only allowed to contain string data. The output of a Fuzzy Lookup transformation includes those pass-through fields that the developer requires for downstream data-flow operations, as well as additional metrics that provide metadata around the fuzzy searching operation that was conducted. Two extra columns are added to the Fuzzy Lookup output stream, _similarity and _confidence. These two fields can be used together to decide whether an input record might be a duplication of an existing reference record that should perhaps be updated, or a new record that should be inserted, becoming a new reference record.

The Fuzzy Grouping task performs the same operations as the Fuzzy Lookup task but instead of evaluating input records against an outside reference table, the input set becomes the reference. Input records are therefore evaluated against other records in the input set and evaluated for similarity and assigned to a group. Later in the article, I'll outline a scenario where the Fuzzy Grouping task can be applied.

Confidence and similarity are evaluated per input record. This is an important point, since it is possible to return multiple reference records for input record processed by a Fuzzy Lookup. Similarity is a score between 0 and 1 that describes the commonality between an input record and the possible match records found by a Fuzzy Lookup in the reference table. As the similarity score approaches 1, the match record becomes closer to an exact match of the input search record. A score of 1 indicates a perfect string match. Similarity is also output by Fuzzy Lookup at a column level.

Confidence is also a score between 0 and 1; it describes the likelihood that each possible match record returned by the Fuzzy Lookup is a match to the input record. This measure indicates the probability that each returned reference record for any one input record is actually a match. A high similarity factor does not necessarily mean that a definitive match has been found, but the combination of high similarity and confidence scores usually does.

Designing Fuzzy Search Operations in SSIS

When configuring a Fuzzy Lookup in an SSIS data flow, there are several key decisions to be made regarding match index reuse, reference table sources, similarity thresholds, and the input and output columns that determine a record's uniqueness.

First, you need to determine whether your match index will be reused for other Fuzzy Lookup operations. Match indexes can be reused as long as they are based upon the same set of columns. In other words, a match index configured to use the customer last name, address 1, city, state, and ZIP code fields for a deduplication process in one SSIS package can be reused by other packages only if the same set of fields can be used to perform the search. When a match index is reused, Fuzzy Lookup transformations generally run faster because they don't have to generate their own match index each time the SSIS package is run. For this reason, match index reuse is desirable if the reference table remains static.

For reference tables that do not remain static, the Fuzzy Lookup transformation can be configured to maintain the match index as the underlying reference table changes. The Fuzzy Lookup Table Maintenance feature allows the error-tolerant index to be updated each time the reference table changes and a Fuzzy Lookup task is executed against it. This feature involves adding a trigger to the reference table itself, so performance implications should be weighed carefully when considering this technique. As an alternative to both of these methods, the match index can simply be recreated each time the Fuzzy Lookup transformation is executed.

The next steps are setting the similarity threshold and maximum number of results per input record. The MinSimilarity custom property tells the Fuzzy Lookup transformation how similar a reference record must be to be returned by the search. Additionally, the transformation needs to know how many matches to output for each input record. The MaxOutputMatchesPerInput custom property sets this value. The maximum output matches per input setting and performance have an inverse relationship. The more records that the Fuzzy Lookup must locate and output, the slower the search will be. Likewise, the lower the similarity threshold setting, the longer the search duration, because the set of possible reference table matches increases.

Field Selection and Weighting

It is important for developers to understand that the fuzzy search technology available in SQL Server 2005 is powerful, but it's also domain-agnostic in its implementation. Therefore, the selection of fields that provide linkage information used to measure the similarity of input and reference data is more of an art than a science and will become the primary factor influencing a fuzzy search's accuracy. Ralph Kimball, noted as one of the fathers of modern data warehousing techniques, calls these attribute sets "survivorship blocks" in his book *The Data Warehouse Toolkit* (Wiley, 2002). Depending on the business domain and purpose of the operation being performed on input records, the survivorship block you define will vary, as will the weighting that each field in the block receives toward an overall match score. To illustrate the use of field selection and weighting for different matching scenarios, let's talk about deduplication and householding in more depth.

In a customer record deduplication scenario, such as that found in the sample project, the ultimate goal is to ensure that one accurate up-to-date record exists in the data warehouse's customer dimension for each unique customer across all source systems. Given the schema of the customer dimension shown in **Figure 5**, I must first decide which fields I might combine to form a natural key for each of my customers. For the purposes of this deduping effort, choosing CompanyName, ContactName, Address, City, Region, and PostalCode as my survivorship block should provide me with a unique customer reference. After adding these attributes to the Fuzzy Lookup transformation properties, setting my minimum similarity threshold and maximum results per input record, I am ready to begin testing input records for similarity to my reference table customer. By running subtly different input records through the ETL process, you'll get the output shown in **Figure 6**.

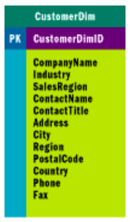

Figure 5 Customer

The fields chosen for each entity may differ significantly depending on your business rules. In addition to selecting multiple columns that represent similarity, you can also take advantage of the Minimum Similarity per Column feature, which allows match similarity and confidence scoring to be controlled at the field level.

Some columns should be valued more highly toward an overall similarity score than others. By using the advanced Fuzzy Lookup editor (right-click Fuzzy Lookup transformation task, Show Advanced Editor), match contribution values can be set higher for individual fields in the survivorship block that require a higher degree of accuracy to be considered a good match. The salutation and state fields in a customer table demonstrate why more or less emphasis should be placed on certain attributes. The salutations and state abbreviations for Alabama (AL), Alaska (AK), and Arizona (AZ) may appear relatively similar to the Fuzzy Lookup transformation, but business knowledge and understanding of these data elements suggest otherwise. Thoroughly studying the potential impact of each domain-specific attribute utilized in a Fuzzy Lookup will in essence allow you to train the operation to think in domain-specific terms. Any merging or conforming that occurs as the result of a Fuzzy Lookup's output should be logged to an audit table. Over time, analysis of audit records may indicate that the settings on the Fuzzy Lookup may need to be tweaked to be more or less inclusive in similarity evaluations.

When evaluating hierarchical data, such as a data warehouse dimension, the Minimum Similarity per Column feature can also be brought to bear. This setting instructs the Fuzzy Lookup transformation to automatically apply a heavier weighting to the attributes closer to the leaf level of the dimension hierarchy.

When designing a fuzzy search operation to aid in householding activity, the same concepts still apply, but the survivorship block will differ. Unlike in the deduping scenario, it is desirable within a householding process to search for matches only down to the individual customer household level, not the individual customer. The Address, City, Region, and PostalCode fields are utilized, leaving the name, social security, birth date, and other identifying attributes out of the survivorship block. In the end, customers who share the same address should be grouped by and assigned to a household identifier.

To achieve this with SSIS, use a two-step process. Initially, a grouping operation should be performed on the clean set of data representing the total customer population. For this task, use the

Fuzzy Grouping transformation. The Fuzzy Grouping transformation accepts a set of input records and utilizes Fuzzy Lookup functionality under the covers to create a match index and assign a grouping key to sets of input records that meet the minimum similarity threshold.

The Fuzzy Grouping transformation uses a slightly different technique. It generates similarity and confidence scores based solely on the other records in the input set. Essentially, this transformation uses an input set against itself as the reference table. The Fuzzy Grouping generates a temporary match index (with the help of Fuzzy Lookup behind the scenes), which is used during the grouping operation. During the transformation process, three key fields are added to the output stream of the task: _key_in, _key_out and _score, as shown in **Figure 7**. The _key_in column is the reference key (RID) that is assigned to an input record when the underlying match index is generated. The _key_out column points to the reference key (_key_in) of another record in the input set considered by the fuzzy grouping to be a match according to the minimum similarity threshold setting that has been chosen. The _score field represents the similarity of the _key_in and _key_out records.

Based on a matched group of records, address information can be standardized, conformed, and stored as a household identifier. After the initial grouping utilizing the Fuzzy Grouping transformation has parsed out individual households, a Fuzzy Lookup transformation can be used to create household assignments as new customer records enter the customer dimension.

Conclusion

The fuzzy search capabilities provided by SQL Server 2005 Integration Services serve as a framework for building robust data cleansing solutions. The underlying technology utilizes state-of-the-art techniques for identifying similarities in error-prone data. This level of configurability comes with performance and storage tradeoffs, but the ever increasing processing power and near limitless storage capabilities of today's business servers allow robust processes like the Fuzzy Lookup and Grouping transformations to be executed with minimal effort.

Jay Nathan is a Business Intelligence Architect with Blackbaud, Inc., a global provider of software and related services in Charleston, SC. He has delivered solutions to the utilities, telecommunications, insurance and healthcare industries using Microsoft technologies.

From the September 2005 issue of MSDN Magazine.

Figure 1 Duplicate Customer Records

First Name	Last Name	Address1	Address2	City	State	ZIP Code
John	Doe	112 Sunny Vale Ln.	Apt. #23	Anytown	NC	28227
John	Do	112 Sunny Vail Lane	Apt 23	Anytowne	NC	28227-5410

Figure 6 Sample Fuzzy Lookup Output

CompanyName	ContactName	Address1	City	State	ZIP Code	_similarity	_confidence
Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	OR	97403	.80674612	.93456379
Great lakes Food Mart	Howard Synder	2732 Baker Blvd.	Eugene	OR	97403-1176	.74123439	.67434212

Figure 7 Additional Output Fields Generated by a Fuzzy Grouping Transformation

_key_in	_key_out	_score
3	3	1
4	3	0.93498355
2	3	0.89746004
1	3	0.82209551

Related Articles from MSDN Magazine:

- **SQL Server:** Uncover Hidden Data to Optimize Application Performance by Ian Stirk
- **Data Points:** Data Binding in WPF by John Papa
- **Data Security:** Stop SQL Injection Attacks Before They Stop You by Paul Litwin
- **Data Points:** Accessing Data from a Mobile Application by John Papa
- **SQL Server 2005:** Regular Expressions Make Pattern Matching And Data Extraction Easier by David Banister
- **Data Points:** Exploring SQL Server Triggers by John Papa
- **Data Points:** Common Table Expressions by John Papa
- **VISUAL BASIC:** Unleash The Power Of Query In Visual Studio "Orcas" by Ting Liang and Kit George