Transforming data2knowledge

D2K Developer White Paper

D2K's Integrated Structuring Platform (ISP)
Feature Summary

Extracts data from any number of data sources:

- Retrieves data from one or more data sources
- Separates needed and meaningful pieces of information from the unneeded and meaningless pieces
- Handles any kind of data source, including semi structured data sources, web pages, relational databases, applications, text files, user feedback and more.
- Data source navigation hierarchy is defined by describing a spanning tree of the data source structure; actual traversal of this spanning tree happens automatically by using breadth-first, depth-first or custom algorithms, with the maximum possible degree of parallelism
- Additional basic data source types, data formats and traversal algorithms can be created and used through the Navigator, Dataview and Traversal API's

Transforms extracted data:

- Transforms individual pieces of data into other representations (e.g. converts currency values)
- Transforms the overlying data structure into another (e.g. renames fields, breaks down complex data items into simple ones, or assembles smaller data items into larger, complex units)
- Additional transformation algorithms can be created and used through the Tagger and Transformer API's

Integrates data processing tasks:

- The extraction and transformation units can be combined in a data processing network, where data retrieval, extraction and transformation can be driven and controlled by previously extracted data
- Data flow through any point of the processing network can be directed into data sinks that automatically handle the relations between different types of data records; these data sinks can output structured data into many different formats (relational databases, XML, e-mail or SMS notifications, text files etc.) through Renderers
- The configuration of the data processing network is stored in the central metadata repository which can be accessed from external applications, so a single source for metadata is available for every application that uses D2K Core Technology™ technology
- The data processing network is scalable to any number of processing components, regardless of the physical level of integration (multiprocessing, clustering or distributed computing).
- Additional output data formats can be defined through the Renderer API
- Through its extensive set of API's, any kind of extraction/transformation or processing algorithm can easily be integrated into the integrated structuring platform (ISP).

D2K Technology Architecture

In general terms D2K's ISP is a network of interconnected Data Extraction Components, Data Transformation Components and Datasinks.

- Data Extraction Components accept records as parameters, extract data from one or more data sources based on these parameters, and output the extracted data into the network as records. These records can be used to control other Data Extraction Components or they can be transformed by;

- Data Transformation Component, which is used to group together any number of these, and transform them into more complex records, which can again be fed into the data processing network. From any point in the network, records can be dispatched to;

- Datasinks, where they will be rendered into their final physical format(s) (e.g. a row in a table of a relational database, a mail message used for notification, or a text file - any number of renderers can be configured for a single datasink).

The network remembers how more complex records are born, and this information can be made available in the final physical representation of the data. Multiple networks can be joined together using special Data Sinks and Renderers.

D2K's ISP, in its simplest configuration, is comprised of a single Data Extraction Component, a Datasink and one Renderer. In this configuration it can be used to extract data from one or more data sources, execute data-item level transformations and simple mappings and output the data into one or more formats. In a more complex, general case it also contains multiple Data Transformation Components, in addition to multiple Data Extraction Components and Datasinks; in this case the system can be used to monitor multiple information sources, process their information outflow, provide notifications or directly react to events. An example for this use may be an Audit Trail Reduction system, where multiple system and transaction log files may be monitored for complex unusual events (e.g. monitoring money flow and alerting for unusual transfers).

The two most important components of the ISP architecture are the Data Extraction Component and the Data Transformation Component.

The Data Extraction Component works in a three-stage pipeline: as soon as a pipeline stage is finished, it passes its output to the next stage and starts processing its next input. This means that as soon as the processing of a particular piece of information is finished, it is available for further processing as a new output record. In effect, records appear one-by-one, as soon as available, on the outputs (although this is not necessarily visible in end user applications because Datasinks and Renderers may buffer records for performance reasons œ obviously, buffering only happens if needed, since no notifications should be buffered for example).
The three stages of the pipeline are Data Source Navigation, Data Item Extraction and Data Item Transformation:

- Data Source Navigation: In the first stage of the pipeline, the native interface of the data source is used for retrieving information, in units that the data source supports. For example, the system may connect to a relational database through ODBC and use SQL to retrieve data.

- Data Item Extraction: the second stage of the pipeline, extracts individual data items from a retrieved data unit. A retrieved data unit may be a row from a SQL database or an HTML document, while a data item may be a field of a row or an element in the HTML syntax tree. D2K Core Technology™ applies patented Data Views to retrieved data units to expose their structure and enable location of specific data items. Multiple Data Views can be applied to the same data unit to view its contents from multiple perspectives, which enables higher accuracy content extraction and higher tolerance for structure changes.

- Data Item Transformation: this stage of the pipeline is a simple transformation stage, where a number of predefined and user-defined operations may be applied to one or more data items; as an example, multiple data items may be concatenated, or data types may be converted etc.

The description of the actual data source structure is located in the Metadata Repository. This description provides a high-level, hierarchic description of the data source, which can be as specific as a —site map“ or as general as a web crawler, or anything in-between. This description is then converted to a software pipeline made up of the above mentioned stages. Although there may be a number of interdependencies between individual pieces of

data, the system is constructed to minimize their effect and can utilize all available resources for high-performance processing. Specifically, this means that the data source structure description in the Metadata repository is a general description of the structure of the data source, most importantly taking data interdependencies and data source state requirements into consideration, not a specific algorithm for traversing the data source hierarchy; for actual data retrieval, a number of pre-made and custom traversal algorithms are available for use.

Most of the time, the system is configured through data source descriptions stored in the Metadata Repository, but some projects may need more flexibility. Applications, databases or other, unsupported data source types must be handled, or new, more efficient algorithms for handling a particular data type may be invented. D2K's ISP can be augmented to support these extensions through its comprehensive set of API's. The Data Extraction Component supports the following API's:

All of these extensions to D2K's ISP are implemented as custom objects, the API's are specified by standard IDL interface descriptions. This means, that the Transformer API is used to implement Transformers, the Dataview API to implement Data Views etc.

- Navigators (Navigator API): Navigators are used to navigate the data source, i.e. address and retrieve one or more data units and effect changes in the data source state that enables the addressing and retrieval of these data units (for example, logging into a site is effectively changing the state of the data source to enable data unit œ in most cases, HTML document œ retrieval).
- Dataviews (Dataview API): These objects are used to map a structure of a data unit, and provide an interface for the structure descriptions to locate and name individual pieces of data (data items) in data units. Dataviews and Navigators have a very close, symbiotic relationship: most Navigators supply their result in a predefined Dataview (for example, a SQL navigator supplies a Recordset dataview), although dataviews may be converted into each other through standard (XML) interfaces. An extracted data item can be remapped with another data view (for example, if a relational database stores XML documents, first a Recordset data view, then an XML Syntax Tree data view is used.) Also, a single data unit or data item can be mapped by multiple data views, to increase accuracy for example.
- Transformers (Transformer API): A Transformer transforms one or more data items into a new data item. Transformers act like simple multi-parameter functions: data items (and other, static parameters) go in, and a new data item comes out. They can be used to perform simple data item transformations, such as string concatenation, mathematical expression evaluation etc. and they can be embedded into each other.
- Traversals (Traversal API): As mentioned, data source descriptions do not provide the algorithm to traverse the data source hierarchy, they just describe the structure. Whether a depth-first traversal, a breadth-first traversal or a custom traversal algorithm is necessary must be decided by the data source description creator. Implementing custom traversal algorithms is supported by Traversal objects. Traversals support launching multiple navigators asynchronously to increase system throughput.

Using the above described four API's, any data extraction task can be solved quickly and efficiently, along with most simple data transformation tasks. However, sometimes bigger changes may be needed in the high-level structure of the retrieved data. Let's take a very simple, fictitious example: We are collecting soccer results from multiple websites. Most of the web sites return the data in —normal" structure, which means, that we extract records of the following structure: —HomeTeam", —HomeScore", —AwayTeam", —AwayScore", —Date". A simple data table in this structure looks like this:

HomeTeam	HomeScore	AwayTeam	AwayScore	Date
Smalltown	2	Bigtown	1	1-Jan
Bluesky	2	Greengrass	2	1-Jan
Greengrass	3	Smalltown	3	1-Feb
Bigtown	2	Bluesky	2	1-Feb
Bigtown	1	Greengrass	2	1-Mar
Smalltown	3	Bluesky	2	1-Mar

However, let‘s suppose we have a rogue web site that for some reason supplies the data in an unconventional format: —TeamName“, —Score“, —HomeTeam“, Date“

TeamName	Score	HomeTeam	Date
Smalltown	2	Smalltown	1-Jan
Smalltown	3	Greengrass	1-Feb
Smalltown	3	Smalltown	1-Mar
Bigtown	1	Smalltown	1-Jan
Bigtown	2	Bigtown	1-Feb
Bigtown	1	Bigtown	1-Mar
Greengrass	2	Bluesky	1-Jan
Greengrass	3	Greengrass	1-Feb
Greengrass	2	Bigtown	1-Mar
Bluesky	2	Bluesky	1-Jan
Bluesky	2	Bigtown	1-Feb
Bluesky	2	Smalltown	1-Mar

This table contains the exact same information as the previous one, only in a different format. Simple —data item transformations“ don‘t help here œ the number of rows in the two tables are not equal, for one thing, which precludes the use of this mechanism œ this is why we call this the —schema transformation“ issue. Specific simple problems like this can be solved easily by special applications, but D2K’s ISP provides a general solution for this type of problem, in the form of Data Transformation Components. Another problem that Data Transformation Components were designed to deal with is audit trail analysis and reduction. Log files contain records of individual events, which generally form a part of a complex chain of events, which may be considered an event in itself. These records for these events may be scattered across multiple log files, and they may not even be close to each other (either in time or in the log file itself). D2K Core Technology™ Data Transformation Components can be programmed to recognize temporal and information patterns in information streams and convert the data from these patterns into higher level data structures. For example, entries from a web server log file can be grouped into sessions to facilitate clickstream analysis.

A Data Transformation Component receives records as input and produces records as output. In the first step, records are grouped based on the data they contain by Record Taggers; in the second step, they are further grouped based on previously received records (Data Pattern Processing). If a pattern is recognized, or a part of a pattern, a new record is dispatched (Data Record Dispatch).

A Data Transformation Component may correspond to a real-world question, a fact, an entity or a relation between entities. As a simple example, a network of Data Transformation Components can be used for structuring complex data graphs (for example, finding a —logical“ spanning tree of a web site that can be used for clickstream analysis). Since they work in a parallel architecture, they can easily scale with more processors, and can give results in real time.

Data Transformation Components are described by data transformation configuration descriptions, located in the Metadata Repository, the same that the Data Extraction Components use. As with the Data Extraction Components, their basic functionality can be augmented through a number of API‘s.

All of these extensions to D2K’s ISP are implemented as custom objects, the API‘s are specified by standard IDL interface descriptions. This means, that the StateObject API is used to implement StateObjects, the Tagger API to implement Taggers etc.

- StateObjects (StateObject API): they are used to store/represent complex object states.
- Taggers (Tagger API): Taggers are used to group records into different record groups, based on their content.

Data Transformation Components and Data Extraction Components are combined in a data processing network.

Extracts data from any number of data sources:

Records from any point within this network can be redirected (copied) to so-called Datasinks that distribute them to Renderers, which output them into their final format. Naturally, renderers also have their own API so that D2K's ISP can output data in any data format.