ETL

Data Warehousing

29/11/2010

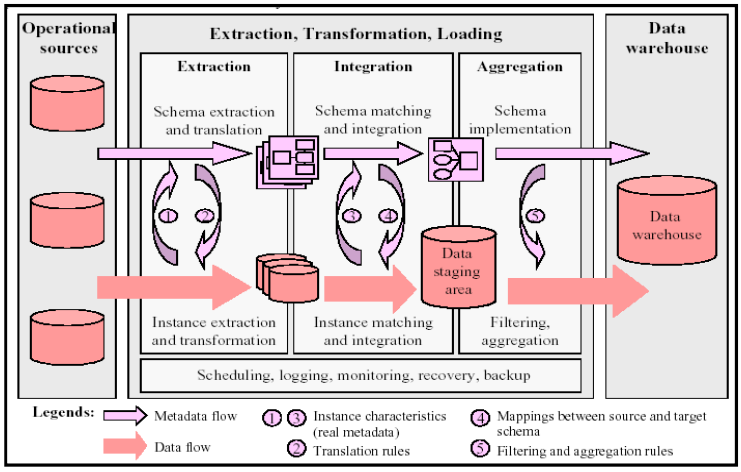
Part 1

- The process of ETL
- Data Transformation
- Schema Matching and Integration
- Some Formal Definitions
- Schema Matching Approaches
- Schema-Level Approaches
  - Granularity of match (element-level vs. structure-level)
  - Match cardinality
  - Linguistic approaches
  - Constraint-based approaches
- Combing Matchers

Part [1] Based on

- Rahm, E., and P. A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," VLDB Journal 10, 4 (Dec. 2001), pp. 334-350
- Erhard Rahm and Hong Hai Do, "Data Cleaning: Problems and Current Approaches"

The Process of Extract – Transform – Load



Data Transformation

- ETL is essentially a process of acquiring data from OLTP.
- Technically, ETL is a Data Transformation process.
- Where Data Transformation is required?
  - Migrate legacy systems to modern applications
  - Optimize queries
  - Translate from one data model to another
  - Integrate heterogeneous systems into federated databases or warehouses
  - Perform data cleansing or scrubbing
  - Evolve a schema and its associated database as driven by changing user requirements
  - Construct user-customized web sites
  - Achieve enterprise-wide integration

The Process of Extract – Transform – Load

- Solutions in ETL
  - Schema Integration and Matching
  - Data Cleansing
  - Data Loading
- A number of strategies for each of these solution.
- Let's start with Schema Integration & Matching

Schema Integration & Matching

- Fundamental Schema Matching Operator - *Match*
  - Input: Multiple, Heterogeneous Schemas
  - Output: Mappings
- Application domain
  - Schema Integration: Structures and Terminological relationships
  - Data warehouses: Source-to-warehouse Transformation
  - E-commerce: Message Translation
  - Semantic query processing: A Run-time Scenario

Some Formal Definitions

- A schema is a set of elements connected by some structure represented by a particular physical model.
- A mapping is a set of *mapping elements*, each of which indicates that certain elements of a schema, say S1, are mapped to certain elements in the other schema, say S2.
- Each mapping element can have a *mapping expression* which specifies how the S1 and S2 elements are related.

Some Formal Definitions (contd.)

S1 Elements	S2 Elements
Table: Cust	**Table: Customer**
C#	**CustID**
CName	**Company**
First Name	**Contact**
Last Name	**Phone**

- Mapping Example
  - Mapping element relating Cust.C# to Customer.CustID
  - Mapping expression Cust.C# = Customer.CustID
- Match operation is a function that takes two schemas S1 and S2 as input and returns a mapping between those two schemas, called the *match result*.

Schema Matching Approaches

- Schema Level Approaches
  - Consider schema-level information only.
  - Information includes the usual properties of schema elements, such as name, description, data type, relationship types (part-of, is-a, etc.), constraints, and schema structure
  - Heavy Metadata usage
- Instance Level Approaches
  - Matching approaches that consider instance data (i.e., data contents).
  - Especially useful when schema information is limited, as is often the case for semi structured data.

Schema-level Approaches

- Granularity of match (element-level vs. structure-level)
- Match cardinality
- Linguistic approaches
- Constraint-based approaches

Granularity of Match

- Element-level matching
  - Determines the matching elements in the second input schema.
  - In the simplest case, only elements at the finest level of granularity are considered, such as attributes in an XML schema or columns in a relational schema.
  - e.g. Address.ZIP = CustomerAddress.PostalCode
- Structure-level Matching
  - Matching combinations of elements that appear together in a structure.
  - In the ideal case, all components of the structures in the two schemas fully match.
  - Alternatively, only some of the components may be required to match (i.e., a partial structural match).

Granularity of Match

S1 elements	S2 elements	
Address Street City State Zip	CustomerAddress Street City USState PostalCode	Full structure match of Address and CustomerAddress
AccountOwner Name Address Birthdate TaxExempt	Customer Cname CAddress Cphone	Partial structural match of AccountOwner and Customer

Match Cardinality

Local match cardinalities	S1 element(s)	S2 element(s)	Matching expression
1:1, element level	Price	Amount	Amount = Price
n:1, element-level	Price, Tax	Cost	Cost = Price * (1 + Tax/100)
1:n, element-level	Name	FirstName, LastName	FirstName, LastName = Extract(Name, …)
n:1, structure-level (n:m element-level)	B.Title, B.PuNo, P.PuNo, P.Name	A.Book, A.Publisher	A.Book, A.Publisher = select B.Title, P.Name from B, P where B.PuNo = P.PuNo

Linguistic Approaches

- Use names and text (i.e., words or sentences) to find semantically similar schema elements.
- Name Matching
  - Equality of names (be aware of Homonyms)
  - Equality of canonical name representations (e.g., CName → customer name, and EmpNO → employee number)
  - Equality of synonyms
  - Equality of hypernyms (E.g., book *is-a* publication and article *is-a* publication imply book=publication, article=publication, and book = article)
  - Similarity of names based on common substrings, edit distance, pronunciation, and soundex
  - User provided name matches

Linguistic Approaches

- Use names and text (i.e., words or sentences) to find semantically similar schema elements.
- Description Matching
  - Schemas contain comments in natural language to express the intended semantics of schema elements.
  - Ex. S1: empn //**employee name**
  - Ex. S2: name //**name of employee**
  - These comments can also be evaluated linguistically to determine the similarity between schema elements.
  - Analysis could be as simple as extracting keywords from the description
  - Or it could be as sophisticated as using natural language understanding technology to look for semantically equivalent expressions.

Constraint-based Approaches

- Schemas often contain constraints to define data types and value ranges, uniqueness, optionality, relationship types and cardinalities, etc.
- A constraint based matcher use such information to determine the similarity of schema elements.
- In the example below, type and key information suggest that Born matches Birthdate and Pno matches either EmpNo or DeptNo.

S1 elements	S2 elements
Employee	Personnel
EmpNo – int, primary key	Pno - int, unique
EmpName – varchar (50)	Pname – string
DeptNo – int, references Department	Dept - string
Salary - dec (15,2)	Born - date
Birthdate – date	

Combining Matchers

- A matcher that uses just one approach is unlikely to achieve as many good match candidates as one that combines several approaches.
- Two ways of achieving this
  - A hybrid matcher that integrates multiple matching criteria
  - A composite matchers that combine the results of independently executed matchers.
- Hybrid matchers provide better match candidates and better performance.
  - Poor match candidates matching only one of several criteria can be filtered out early.
  - Complex matches requiring the joint consideration of multiple criteria can be solved
  - Reduce the number of passes over the schema.