Automated schema mapping

- Motivation for automated schema mapping
- Research approaches
- Syntax-based version mapping approach

Rahm, E. and Bernstein, P.A. (2001) A survey of approaches to automatic schema mapping, VLDB Journal, 10, 334-350.

Ge, C. (2002) Semi automatic version mapping for schemas, MSc thesis, University of Auckland

What need for automated mappings?

- Schemas for same domain are usually very similar
  - 80/20 rule?
- Where schemas evolve (new versions) very little of the previous structure is modified (usually)
- An automated first pass can save time overall
  - 15 min/class * 500 classes = over 3 weeks work to describe
  - Months of code writing...
  - Tedious work
  - Reduced opportunity for errors if mappings are suggested and then generated automatically

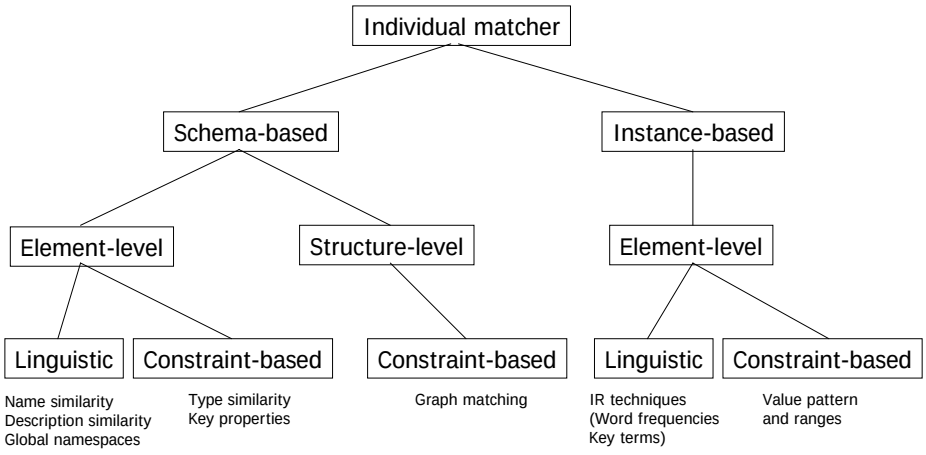
Matching is fundamental

- Typically performed manually
- But, new applications need levels of automatic matching
  - Semantic web
    - Semantic query processing
  - Web-oriented data integration
  - Electronic commerce (EDI, XML, etc)
  - Data warehousing
  - Component-based development
  - Application interoperability
- More sophisticated applications = more complex data models

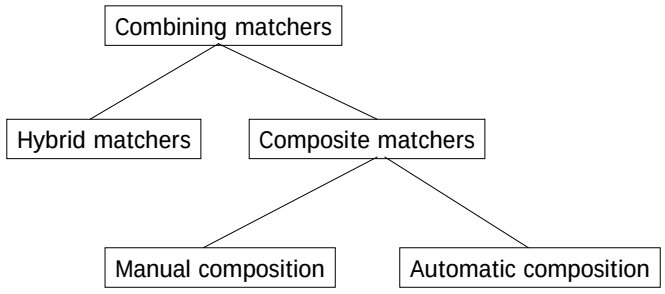
Schema matching approaches

- Instance vs. schema
  - Data level versus type specification
- Element vs. structure
  - Individual schema elements or complex structures
- Language vs. constraint
  - Based on names and textual descriptions versus keys and relationships
- Matching cardinality
  - 1:1, 1:n, n:1, n:m
- Auxiliary information
  - Dictionaries, global schemas, previous matches, user input

Schema matching approaches



Schema matching approaches



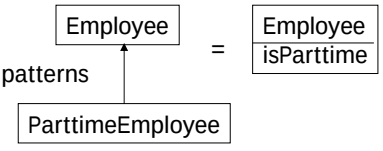
Match examples

Address Street City State ZIP	CustomerAddress Street City USState PostalCode	Full structural match possible
AccountOwner Name Address Birthdate TaxExempt	Customer CName CAddress CPhone	Partial structural match

Schema-level matchers

- Examine structure but not data
  - Name, description, data type, relationship types, constraints, schema structure, etc
  - Matching finds multiple potential candidates
    - Rank potential matches for human to make a decision
- Granularity of the match
  - Element-level
    - Atomic level (eg XML attributes, columns in a table)

      Address.ZIP = CustomerAddress.PostalCode
  - Structure-level
    - Combination of elements
    - Enhance with library of equivalence patterns



Match cardinality

- Elements can participate in 0, 1, or many mapping elements
  - Majority of work on 1:1 matching

Local match cardinalities	S1 elements	S2 elements	Matching expression
1:1 element	Price	Amount	Amount = Price
n:1 element	Price, Tax	Cost	Cost = Price * (1+Tax/100)
1:n element	Name	FName, LName	FName, LName = extract(Name, ...)
n:1 structure (n:m element)	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

- Name matching
  - Equality of names (eg XML namespaces)
  - Equality of canonical name representations after stemming, etc
    CName=CustomerName, EmpNO=EmployeeNumber, etc
  - Equality of synonyms (eg car=automobile, make-brand)
  - Equality of hypernyms (eg book=publication, article=publication)
  - Similarity of names based on common substrings, edit distance, pronunciation, soundex, etc
    RepresentedBy=Representative, ShipTo=Ship2, etc
  - User-provided name matches
    ReportsTo=Manager, Issue=Bug, etc

Name matching

- Require extensive thesauri and dictionaries
  - Multi-language dictionaries
  - Domain specific dictionaries/thesauri
- Homonyms (same words but different meaning)
- Consider structure paths as well

  Author.Name=AuthorName
- Can find multiple matches

  Phone matches HomePhone and OfficePhone

Linguistic approaches

- Description matching
  - Utilise natural language descriptions for semantics

    S1: empn // employee name

    S2: name // name of employee
  - Match on keywords from description
  - Determine semantic equivalence?

Constraint-based approaches

- Utilise data types, value ranges, uniqueness, optionality, relationship types, cardinality, etc
  - Top-down and bottom-up approaches for hierarchical schemas

Employee
  EmpNo – int, primary key
  EmpName – varchar(50)
  DeptNo – int references Department
  Salary – dec(15,2)
  Birthdate – date

Department
  DeptNo – int, primary key
  DeptName – varchar(40)

Personnel
  PNo – int, unique
  PName - string
  Dept - string
  Born - date

Reusing schema and mapping information

- Look for common schema components
  - E.g., address, customer, employee, purchase, order, invoice, etc
  - Libraries of partial schema mappings
    - Domain specific (eg salary is different for payroll and tax apps)
- Need to match new schema to partial mappings
  - The match problem...

Instance-level approaches

- Useful where schema isn't provided (eg semi-structured data) but can be partially constructed
- Can be used as a check against schema-level approach
- Many approaches like schema-level
  - Linguistic characterisation for text elements
    - E.g., keywords based on relative frequency
  - Constraint-based characterisation
    - E.g., numeric value ranges, averages, character patterns (phone numbers, addresses, ISBN, dates, etc.)
  - Works well with libraries of past matches
- Mainly works for element-level matches, not structural
- Requires similar data in both representations

Combining matchers

- Hybrid matcher
  - Combine several matching approaches into one system to determine ranked candidates
  - Performance usually better than composite matcher as fewer passes through schema and data required
- Composite matcher
  - Combines results of several independent matchers
  - Provides more flexibility than a hybrid matcher
  - Trials of machine learning to combine matchers