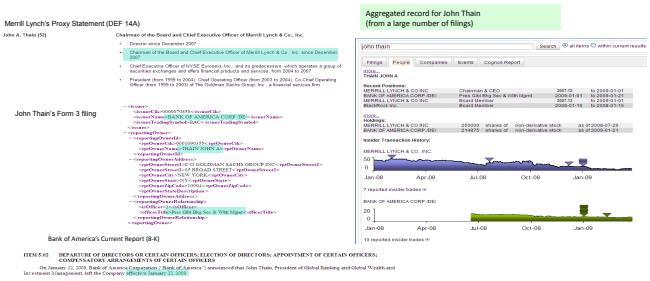
HIL : A Scripting Language for Entity Integration

Ryan Wisnesky, Lucian Popa, Mauricio A. Hernandez, Howard Ho

Entity Integration

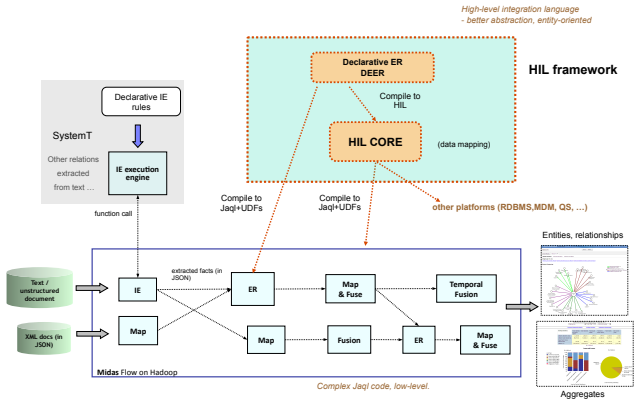
- Renewed interest in non-traditional data integration
- Large amounts of data (structured or unstructured) on the web (SEC filings, federal and state government data, healthcare records, census data)
- Entity Integration: Assemble an entity view of the domain, where each entity aggregates data from thousands of different documents
- Requires complex, end-to-end data processing: exploration, information extraction, entity resolution, mapping, entity fusion
- Relevant research projects:
  - Web of concepts (Yahoo! Research)
  - DBLife (U Wisconsin)
- **IBM's Midas (finance, linked open data, government):**



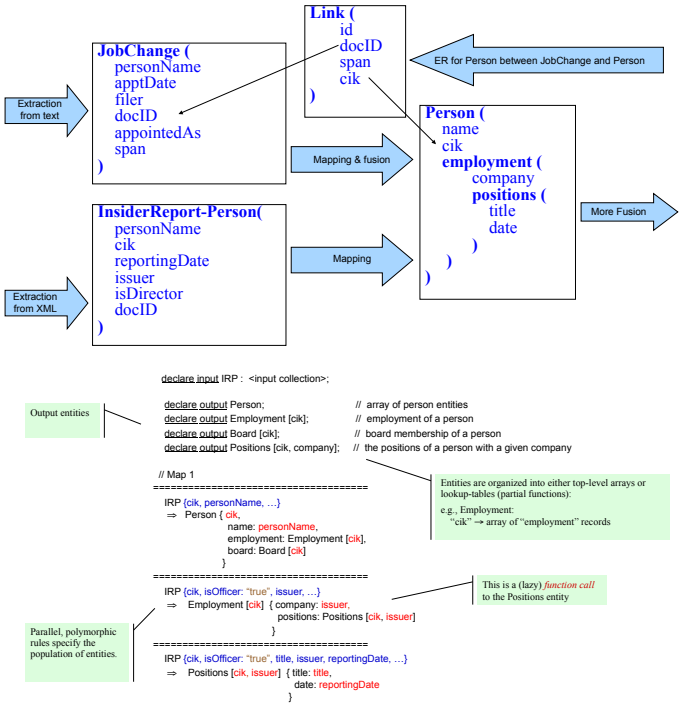
100,000's of SEC documents in many formats → 1000's of people, companies & relationships

Midas and HIL

- HIL captures multiple stages together in one framework:
  - Mapping and cleansing
  - Entity resolution (ER, or linkage)
  - Fusion & aggregation
- Rich data model: define and handle any types of entities and relationships
  - Relational, XML, semi-structured (JSON)
  - Entities to be linked (by ER) do not need to have the same schema
- HIL targets multiple platforms (Jaql Map/Reduce, RDBMS, MDM, Quality Stage)



A Midas Program with HIL



A Midas Program without HIL

```
// Mapping Step 1    (first source)
$Person =
    $IRP → group by $c = $.cik into
              { name: $[0].personName,
                cik: $c,
                employment:
                    ($ → filter $.isOfficer == "true"
                       → transform
                              { company: fn-normalizeCompany($.issuer),
                                title: $.title,
                                date: $.reportingDate
                              }
                       → group by $comp = $.company into
                                { company: $.comp,
                                  positions:
                                      ($ → transform { $.title,
                                                       $.date
                                                     })
                                  // More normalization and grouping by "title"
                                  // is needed. We stop here for now.
                                })
              };

// ER Step 2 (code to produce $Link from $JobChange and $Person)

// Mapping Step 3   (fusing the second source)
$Person =
    join $Person, $Link, $JobChange
    where $Person.cik == $Link.cik
              and $Link.id.docID == $JobChange.docID
              and $Link.id.span == $JobChange.span
              and $fn-isOfficer ($JobChange.appointedAs) == "true"

 into { name: $Person.name,
             cik: $Person.cik,
             employment: $Person.employment,
             delta_employment:
                        { company: fn-normalizeCompany ($JobChange.filer),
                          title: $JobChange.appointedAs,
                          date: $JobChange.apptDate
                        } }
    → group by $c = $.cik into
          {  // Copy person name, cik, and previous employment
             name: $[0].name,
             cik: $c,
             employment: $[0].employment,
              // Process new employment items, as we've done earlier
             delta_employment:
                    ( $[*].delta_employment
                       → group by $comp = $.company into
                                { company: $.comp,
                                  positions:
                                      ($ → transform { $.title,
                                                       $.date })
                                })   // More processing needed for "title"
          }
    → transform
          {  name: $.name,
             cik: $cik,
             employment:
                    ( [$.employment, $.delta_employment]
                       → expand          // union of the two employment arrays
                       → group by $comp = $.company into
                            { company: $comp,
                              positions: ( [$[0].positions, $[1].positions]
                                           → expand  // Another deep union
                                         )  // More processing needed for "title"
                            })
          };
```

IBM Almaden Research Center
Contact: Howard Ho/Almaden/IBM (ho@almaden.ibm.com)