
# Translating Web Data

Ronald Fagin, Mauricio A. Hernandez, Lucian Popa,

*IBM Almaden Research Center*

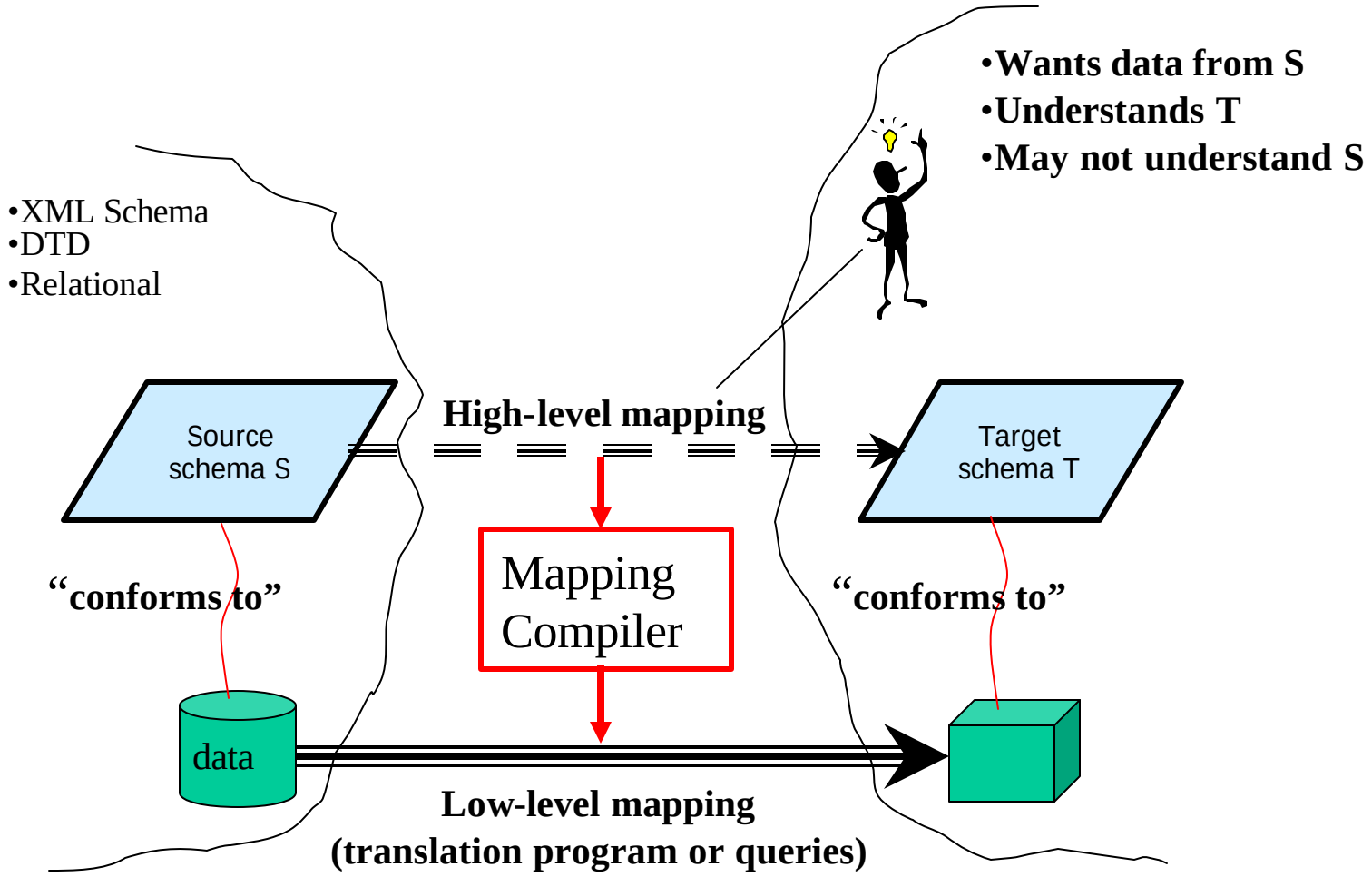
Renee. J. Miller, Yannis Velegrakis (speaker)

*University of Toronto*

# Our Motivation

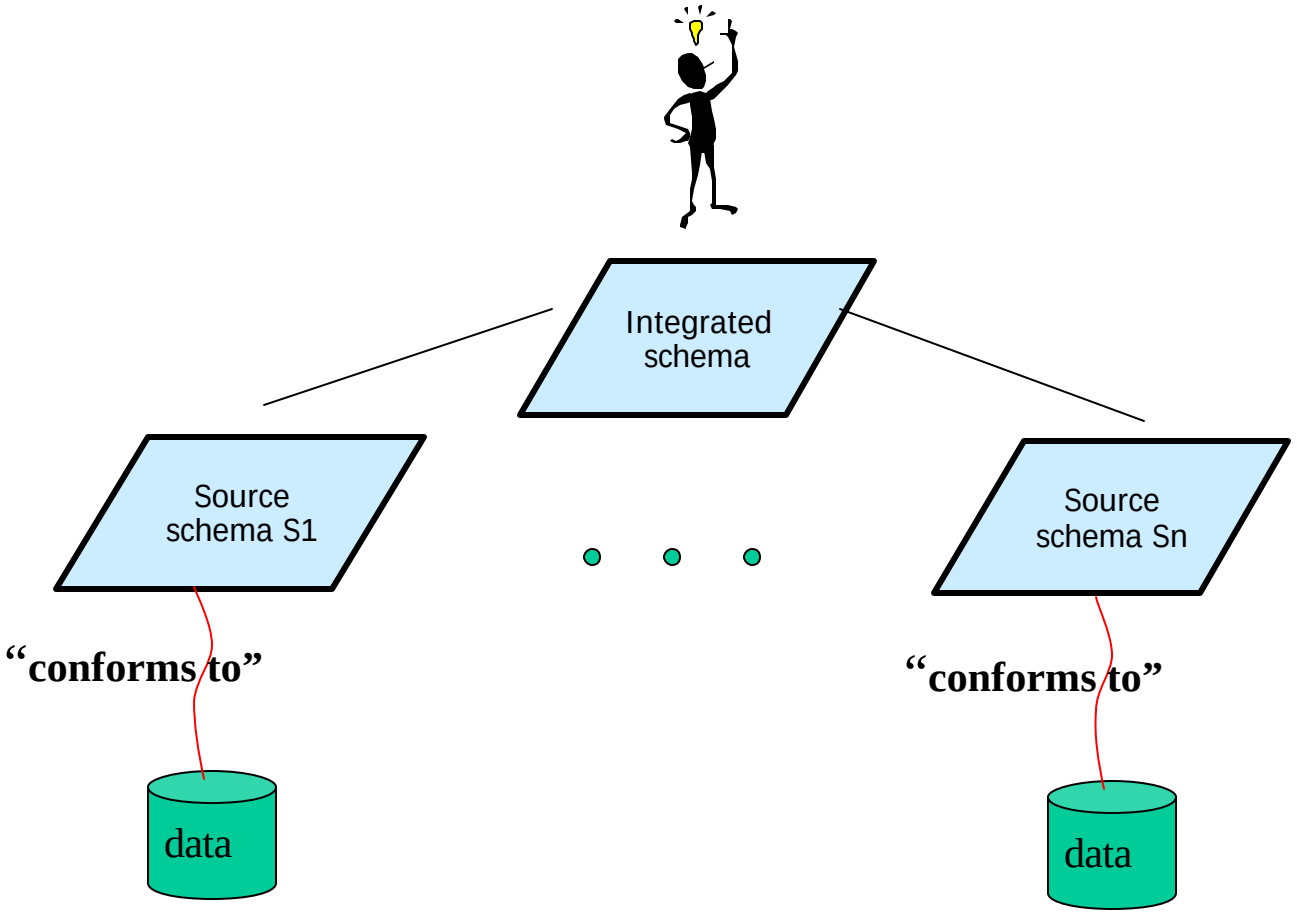
- We address the problem of data translation between schemas.
  - This is an old but recurrent problem (see the old Express project from IBM – 70's)
  - People usually write complex queries to transform data
    - ✓ Time consuming
    - ✓ Requires query experts
    - ✓ Even more when the data and schemas are XML
- Our approach emphasizes *automation* of the task of data translation:
  - Given two schemas (XML and/or relational), and a high-level specification of a mapping between schemas, we generate queries (XQuery/XSLT/SQL)
  - The user does not have to know XQuery/XSLT/SQL
- Major challenges that we address here:
  - Reason about schemas and mapping to *infer* the "right" queries
  - Guarantee that the translated data will *comply* with the target schema

# Schema Mapping & Data Translation




Our approach can be applied in both target materialization and query unfolding

# Mapping is not Schema Integration




Design Problem: integrated schema

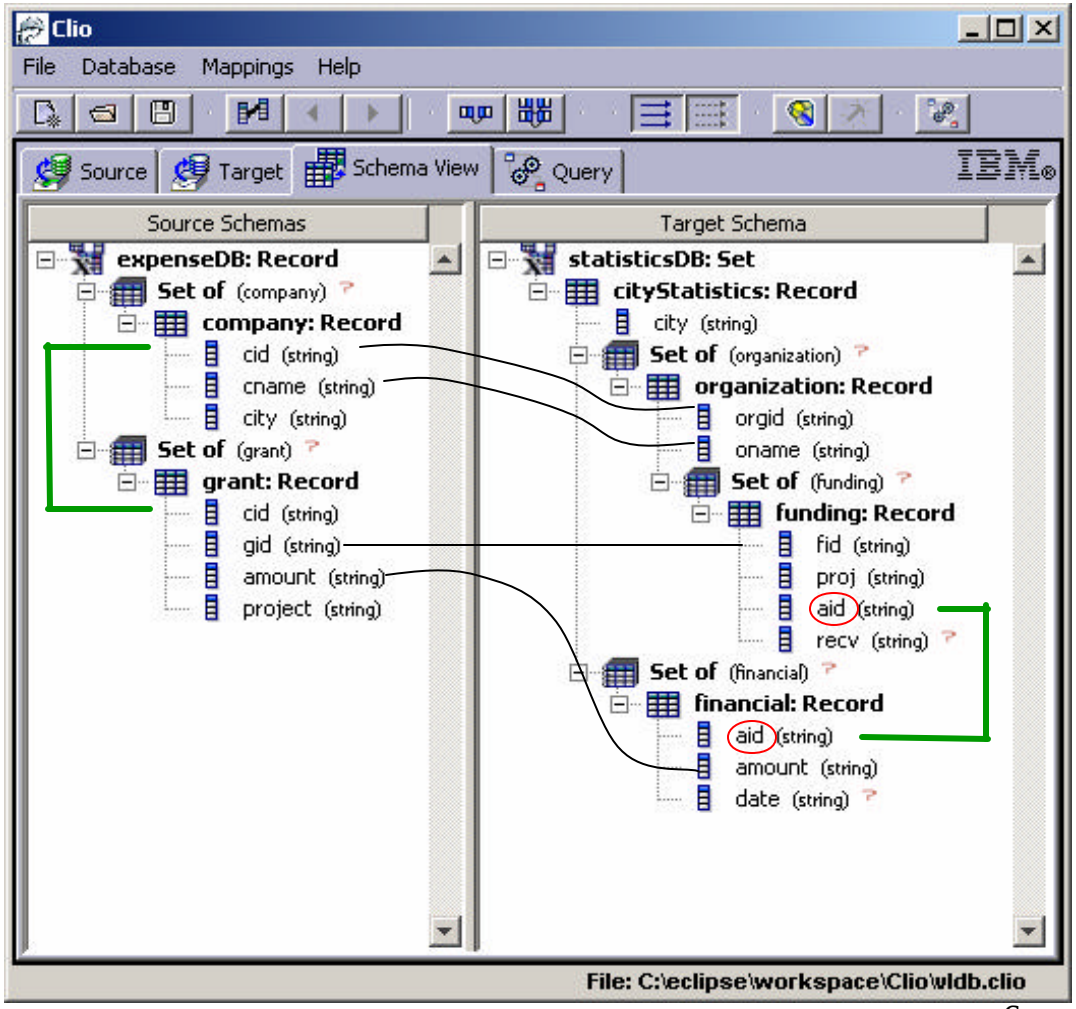
- Designed to match sources
- Has no semantics of its own

# Applications

- Data Migration
- Schema Evolution
- Data Exchange on the Web (P2P applications)
- Software engineering

# Challenges in Schema Mapping

- Goal: interoperability between independent data sources
- Support Nested Structures
  - Nested Relational Model
  - Nested Constraints
- Element correspondences
  - Human friendly
  - Automatic discovery
- Capture User's Intentions
- Preserve data meaning
  - Discover associations
  - Use constraints & schema
- Create New Target Values
- Produce Correct Grouping
- and …



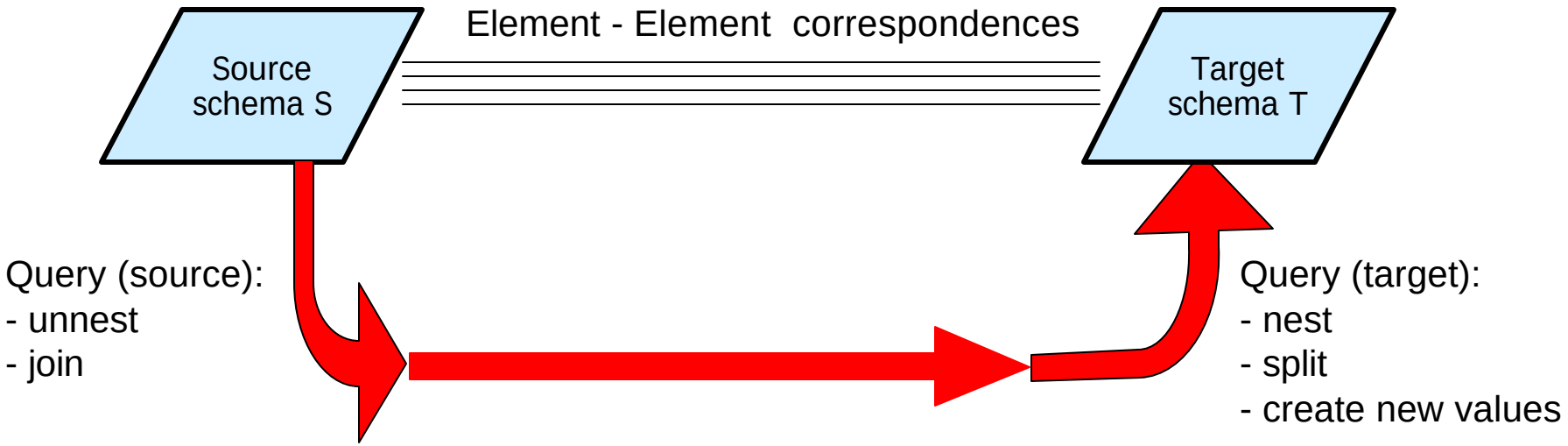
# … Generated Transformation (XQuery)

```
<?xml version="1.0" encoding="UTF-8"?>
<statisticsDB>
   <cityStatistics>
      <city/>,
      distinct (
      FOR
       $x0 IN $doc/expenseDB/grant,
       $x1 IN $doc/expenseDB/company
      WHERE
         $x1/cid/text() = $x0/cid/text()
      RETURN
         <organization>
            <orgid>   $x0/cid/text()  </orgid>,
            <oname>   $x1/cname/text()  </oname>,
            distinct (
            FOR
             $x0L1 IN $doc/expenseDB/grant,
             $x1L1 IN $doc/expenseDB/company
            WHERE
               $x1L1/cid/text() = $x0L1/cid/text() AND
               $x1/cname/text() = $x1L1/cname/text() AND
               $x0/cid/text() = $x0L1/cid/text()
            RETURN
               <funding>
                  <fid>   "Sk35(", $x0L1/amount/text(), ", ", $x1L1/cname/text(), ", ", $x0L1/cid/text(), ")"  </fid>,
                  <proj>   "Sk36(", $x0L1/amount/text(), ", ", $x1L1/cname/text(), ", ", $x0L1/cid/text(), ")"  </proj>,
                  <aid>   "Sk32(", $x0L1/amount/text(), ", ", $x1L1/cname/text(), ", ", $x0L1/cid/text(), ")"  </aid>
               </funding> )
         </organization> ),
      distinct (
      FOR
       $x0 IN $doc/expenseDB/grant,
       $x1 IN $doc/expenseDB/company
      WHERE
         $x1/cid/text() = $x0/cid/text()
      RETURN
         <financial>
            <aid>   "Sk32(", $x0/amount/text(), ", ", $x1/cname/text(), ", ", $x0/cid/text(), ")"  </aid>,
            <amount>   $x0/amount/text()  </amount>
         </financial> )
   </cityStatistics>
</statisticsDB>
```

# … Generated Transformation (XSLT)

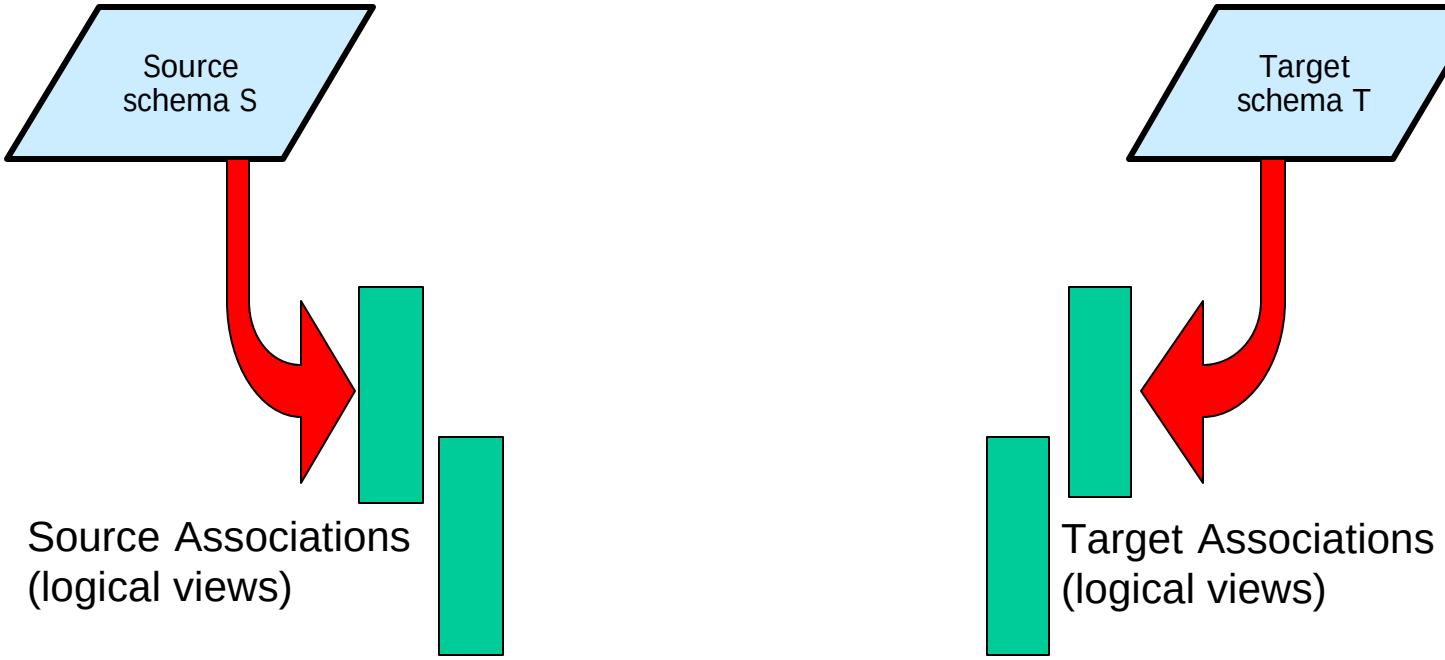
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <result>
            <xsl:call-template name="q0"/>
        </result>
    </xsl:template>
    <xsl:template name="q0">
        <xsl:element name="statisticsDB">
            <xsl:attribute name="isRoot">true</xsl:attribute>
            <xsl:element name="ClioSet">
                <xsl:attribute name="id">Sk_statisticsDB()</xsl:attribute>
            </xsl:element>
        </xsl:element>
        <xsl:for-each select="/expenseDB/grant"> <xsl:variable name="x0" select="."/>
        <xsl:for-each select="/expenseDB/company"> <xsl:variable name="x1" select="."/>
        <xsl:if test="$x1/cid=$x0/cid">
            <xsl:element name="cityStatistics">
                <xsl:attribute name="inSet">Sk_statisticsDB()</xsl:attribute>
                <xsl:element name="city"/>
                <xsl:element name="ClioSet">
                    <xsl:attribute name="id">Sk_statisticsDB_0_1(Sk_statisticsDB())</xsl:attribute>
                </xsl:element>
                <xsl:element name="ClioSet">
                    <xsl:attribute name="id">Sk_statisticsDB_0_2(Sk_statisticsDB())</xsl:attribute>
                </xsl:element>
            </xsl:element>
            <xsl:element name="organization">
                <xsl:attribute name="inSet">Sk_statisticsDB_0_1(Sk_statisticsDB())</xsl:attribute>
                <xsl:element name="orgid"><xsl:value-of select="$x0/cid"/></xsl:element>
                <xsl:element name="oname"><xsl:value-of select="$x1/cname"/></xsl:element>
                <xsl:element name="ClioSet">
                    <xsl:attribute name="id">Sk_statisticsDB_0_1_0_2(<xsl:value-of select="$x0/cid"/>,
                                                                     <xsl:value-of select="$x1/cname"/>,
                                                                    Sk_statisticsDB_0_1(Sk_statisticsDB()))
                    </xsl:attribute>
                </xsl:element>
            </xsl:element>
            <xsl:element name="funding">
                <xsl:attribute name="inSet">Sk_statisticsDB_0_1_0_2(<xsl:value-of select="$x0/cid"/>,
                                                                    <xsl:value-of select="$x1/cname"/>,
                                                                    Sk_statisticsDB_0_1(Sk_statisticsDB()))
                 </xsl:attribute>
                <xsl:element name="fid">
                        Sk35(<xsl:value-of select="$x0/amount"/>, <xsl:value-of select="$x1/cname"/>,
                             <xsl:value-of select="$x0/cid"/>)
                </xsl:element>
              . . . . . .
```

# Mapping Algorithm




- **Step 1**. Intra-schema associations discovery
- **Step 2**. Logical mapping generation
- **Step 3**. Query generation

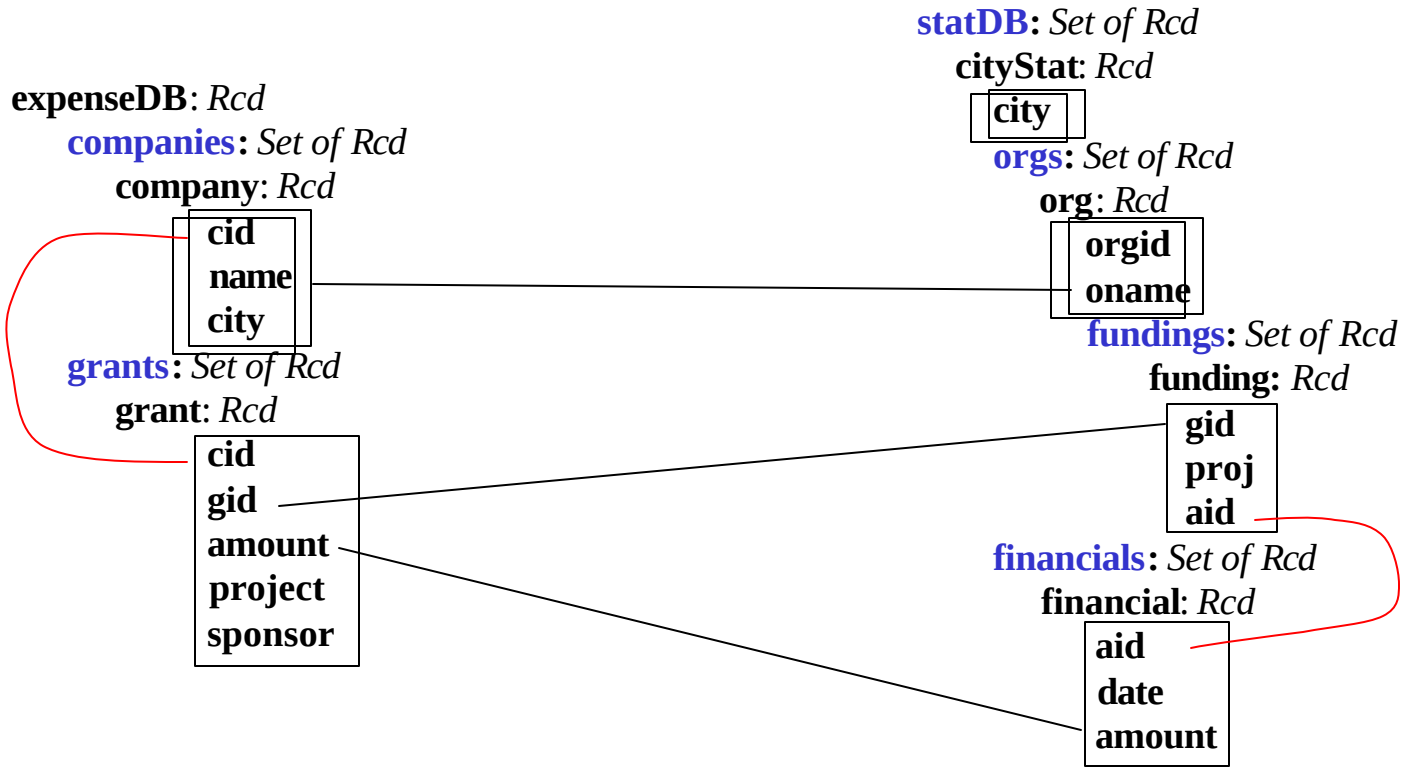
# Association Discovery




- **Step 1**. Discover intra-schema associations between schema elements
  - *relational* views that contain maximal groups of related elements
  - Each represents a different category of data that may exist in the database

# Associations

- Groups of elements that are semantically associated
- *Chase* with intra-schema constraints



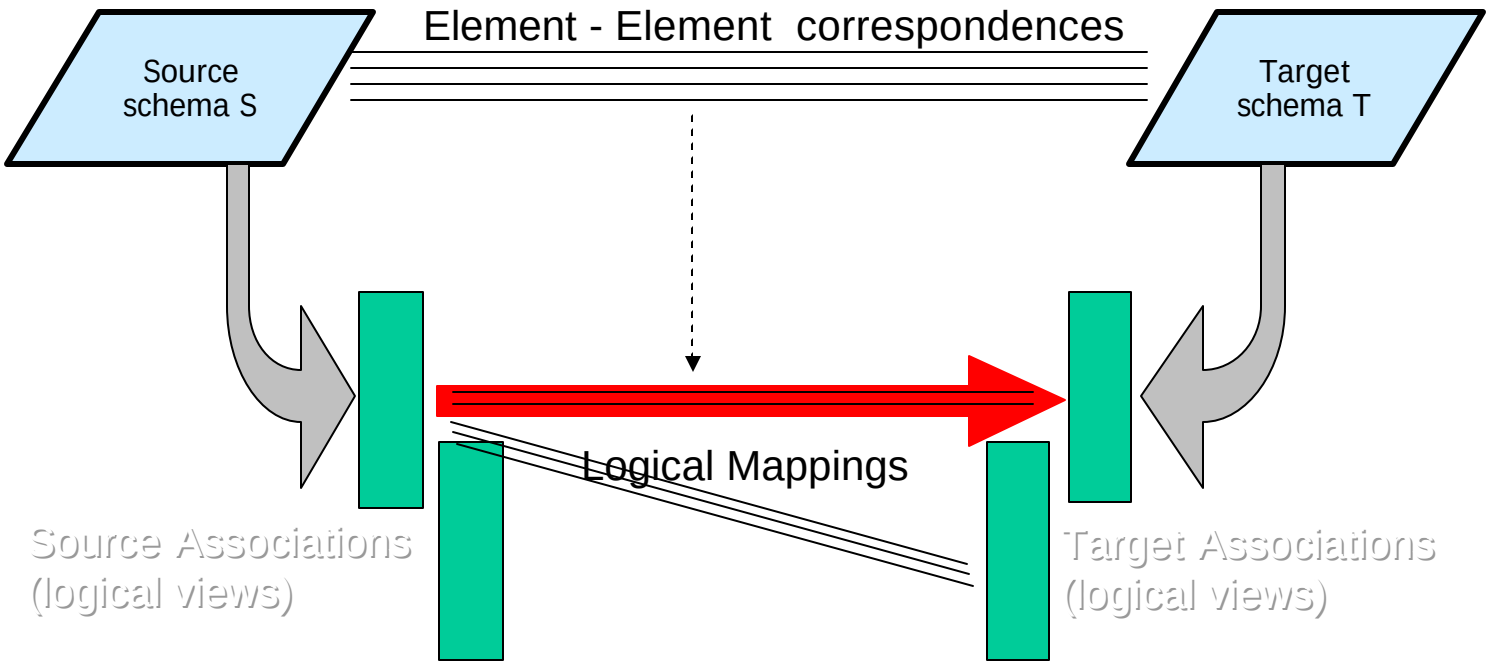

expenseDB.companies

expenseDB.grants
**?** expenseDB.companies

statDB.cityStat.orgs

statDB [ cityStat.orgs.org.fundings
**?** cityStat.financials ]

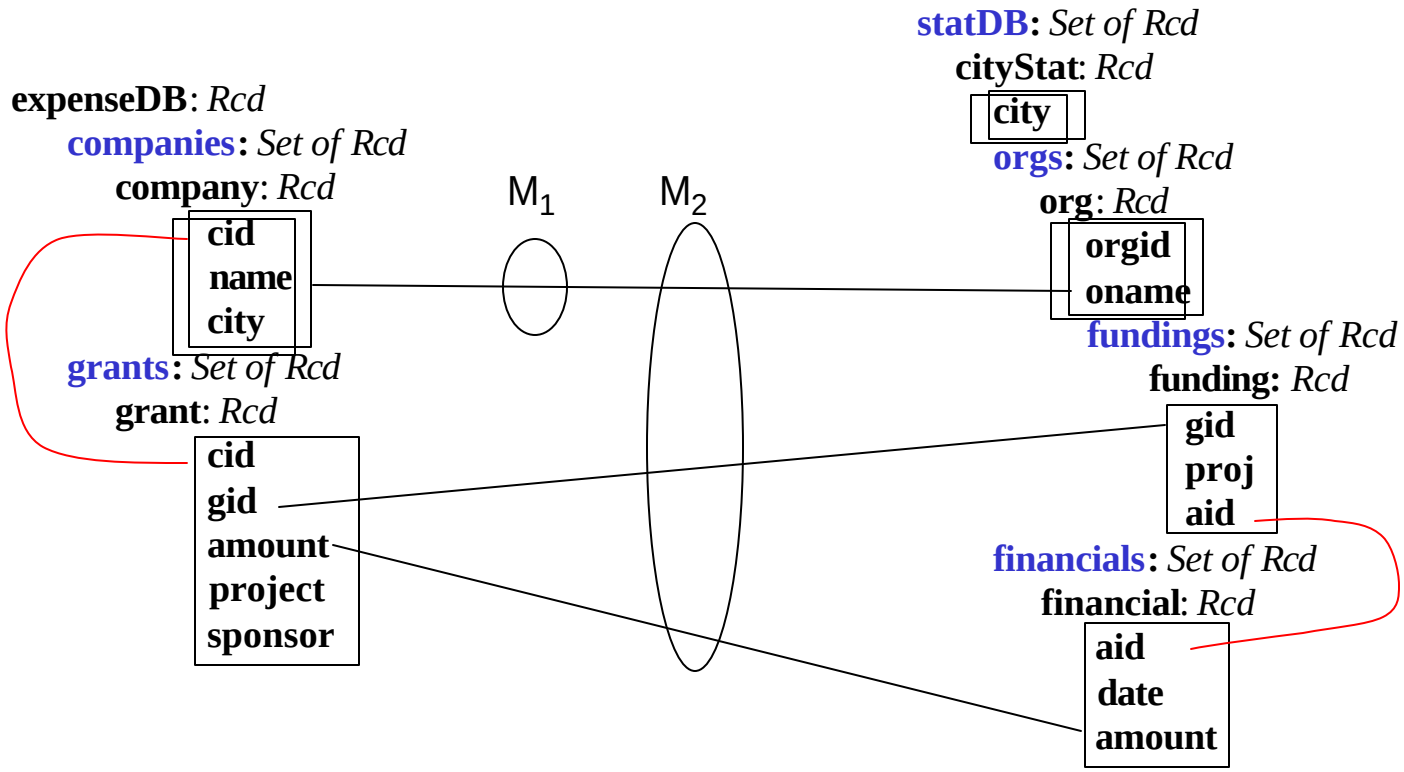
# Logical Mapping Generation




- **Step 2**. Logical mapping generation:
  - each source association -> each target association
    - ✓ based on all correspondences that are relevant
  - By construction, logical mappings preserve associations between elements

# Logical Mappings

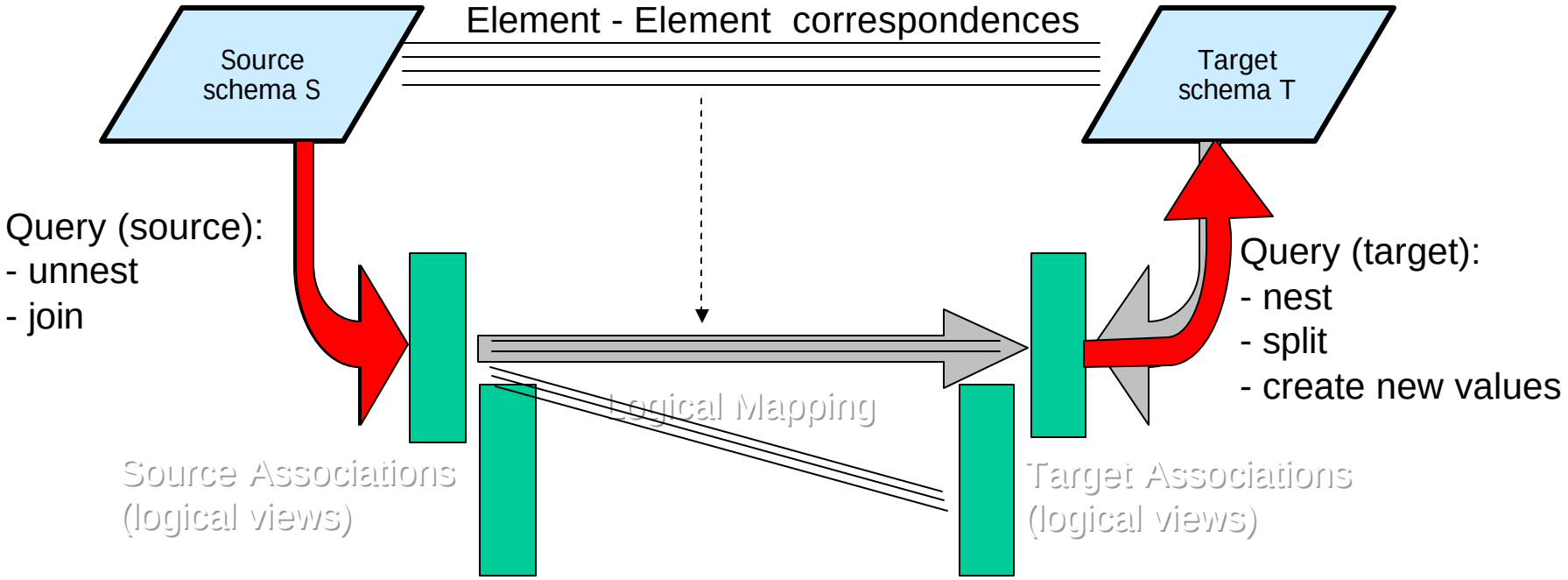
- Inter-schema constraints




$\prod_{\text{name}}$ expenseDB.companies $\subseteq$ $\prod_{\text{name}}$ statDB.cityStat.orgs

$\prod_{\substack{\text{name}\\ \text{gid}\\ \text{amount}}}$ expenseDB.grants ? expenseDB.companies $\subseteq$ $\prod_{\substack{\text{name}\\ \text{gid}\\ \text{amount}}}$ statDB [ cityStat.orgs.org.fundings ? cityStat.financials ]

# Query Generation

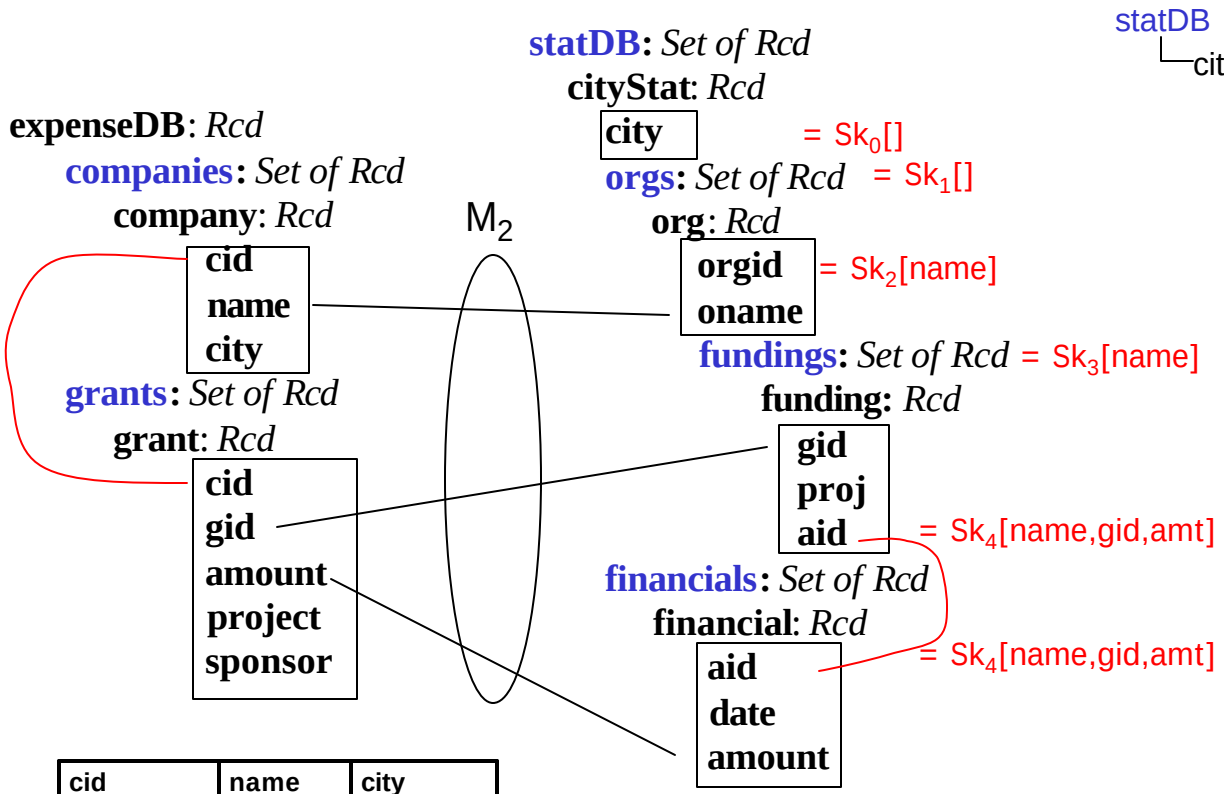



- **Step 3.** Generate queries. Each query:
  - Performs the nest and split operation into multiple target elements
  - May need to create new values (unmapped) in the target

# Query Generation Issues

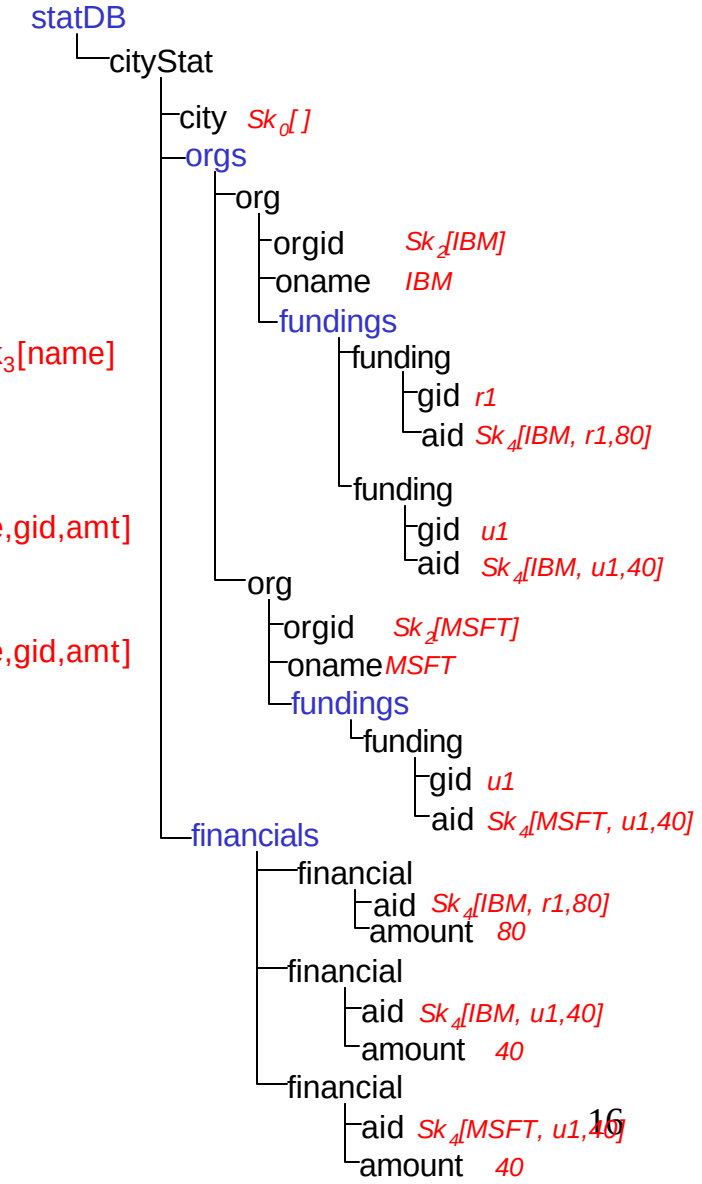
- Translation of the logical mappings into queries
  - flat representation of how schemas correspond
  - not all target attributes are determined by the source
  - we need to materialize the nested target
- **Skolemization** algorithm: the heart of query generation
  - Achieves a good nesting (grouping)
  - Generates new values (ids)
- Skolem functions control the creation of the unknown elements:
  - atomic values (this enforces the integrity of the target) , and
  - sets (this controls how we group elements in the target)
- Skolem functions are automatically generated.

# Skolemization Algorithm




| cid | name | city |
|---|---|---|
| ibm1 | IBM | Almaden |
| ibm2 | IBM | Yorktown |
| ms1 | Microsoft | Redmond |

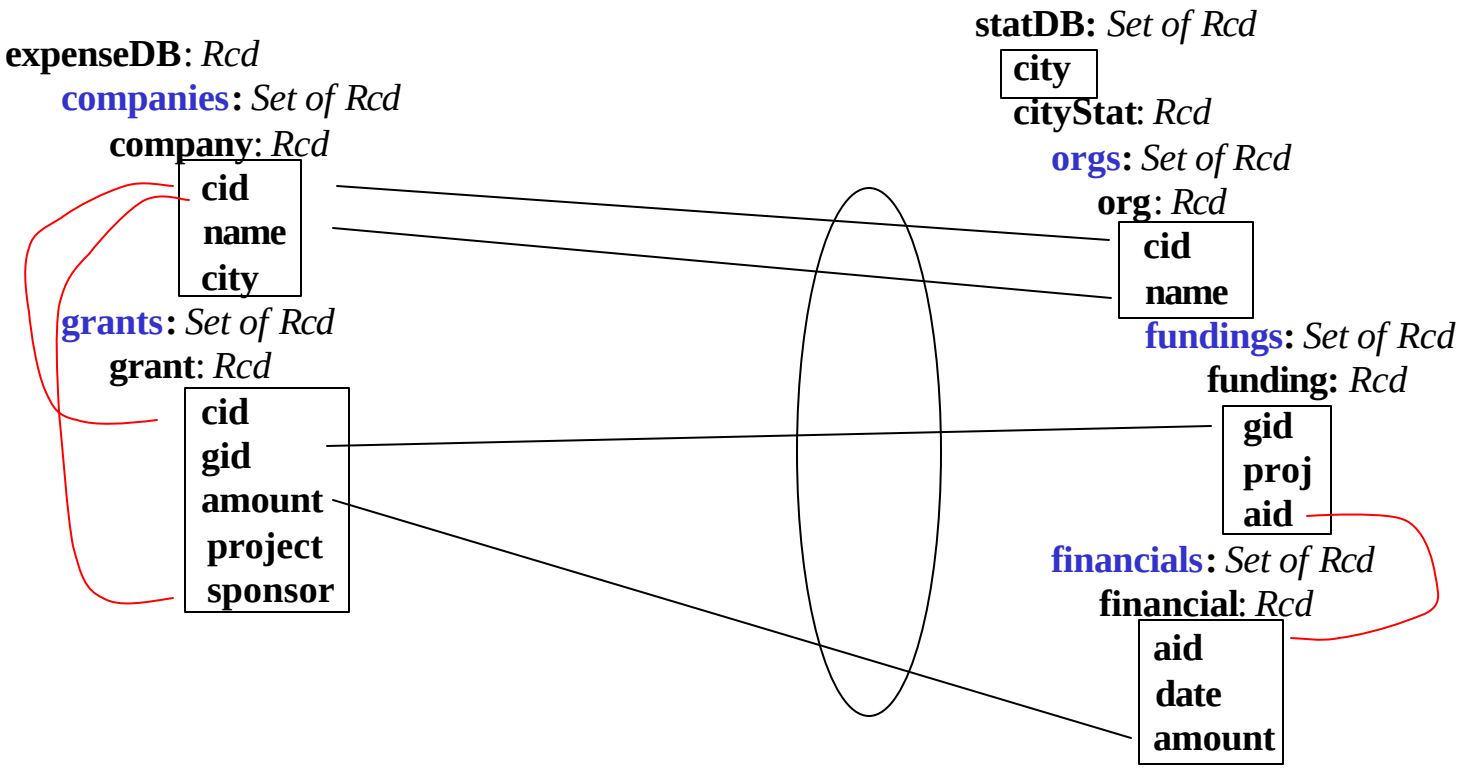
| cid | gid | amount | project |
|---|---|---|---|
| Ibm1 | r1 | 80K | DB2 |
| Ibm1 | u1 | 40K | XQuery |
| ms1 | u1 | 40K | XSL |



# Features

- Produce all the semantically meaningful queries.
  - Finds **all** the associations that exist in the schemas
  - Each one maps from a source association to a target association
  - Allows user to select a subset of them
- Target schema constraints are taken into consideration
  - To infer the user intention
  - To guarantee that the **generated data satisfies the structure and the constraints of the target schema**
- Target Instance is guaranteed to be in Partition Normal Form (PNF)

# Multiple Associations




- Grants may be *associated* with companies in multiple ways
- Association 1: grants **?** companies join on **cid = cid**
- Association 2: grants **?** companies join on **sponsor = cid**
- We do not make the *one flavor assumption (URA)*

# Mapping Algorithm Implementation

- Clio System
  - IBM Almaden Research Center / University of Toronto
    - ✓ http://www.cs.toronto.edu/db/clio
  - Advanced GUI
    - ✓ Constraints
    - ✓ Mappings
    - ✓ Generated queries
- Experiments with real-world schemas

| Schema | Nesting Depth | Attributes | Constraints | Queries w/o constraints | Queries w/ constraints |
|---|---|---|---|---|---|
| DBLP | 1 / 4 | 52 / 12 | 0 / 1 | 11 | 13 |
| TPC-H | 1 / 3 | 34 / 10 | 9 / 1 | 7 | 14 |
| GeneX | 1 / 3 | 65 / 63 | 9 / 3 | 4 | 4 |
| Mondial | 1 / 4 | 102 / 90 | 15 / 21 | 4 | 5 |
| Amalgam | 1 / 1 | 53 / 101 | 26 / 14 | 9 | 10 |

# Conclusion

- Schema mapping framework
  - For data with schemas
  - Covers relational/XML schemas with nested constraints
  - Output: XQuery, XSLT, SQL
  - Build *"data transformations"* (Queries with Skolem functions)
- Future extensions
  - Semantics of ordering in doing mapping
  - Key constraints
  - Recursion of more than 1 level