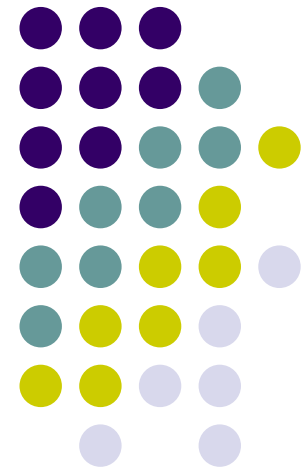


# Tool Requirements for Precise Data Integration

---

Philip A. Bernstein  
Microsoft Research





# Precise Data Integration

- Mapping data between a target schema and one or more data sources to enable:
  - queries and updates over the target schema, or
  - translation of data from the sources to the target.
- Example applications
  - Message mapping for E-commerce
  - Query mediation over heterogeneous DBs
  - Data translation
  - DB wrapper generation
  - DB-driven portals
  - Data warehouse loading

# Other Info Integration Problems

## Possibly with Precise Mappings



- Information resource management
  - Store descriptions of independently managed objects
  - Maintain relationships between them
  - Usually no data transformation semantics
  - For impact analysis, cross-tool navigation, compatible configurations

# Other Info Integration Problems

## Imprecise Mappings



- Exploratory integration (aka dataspace)
  - To integrate data, analyze it and its metadata
  - Use precise data integration tools, fuzzy integration tools, statistics, & other applied math
- Information discovery
  - Use integration tools to find information, with little or no human intervention
  - E.g., semantic web

# Component Operations for all of the above problems



- Model Management
  - Match schemas
  - Merge schemas
  - Compose mappings
  - Diff schemas
  - Invert mapping
  - Translate schema
  - Translate mapping
- Capture & discover lineage (= provenance)
  - Helps understand the result of integration
- Cleaning and correlation
  - Entity extraction, entity resolution, and data clustering

# Some Realities of Precise Data Integration

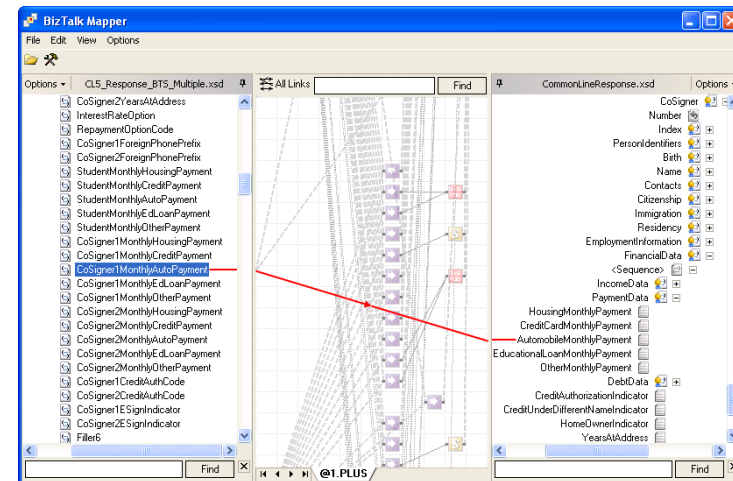


- 40-50% of the software engineering work in enterprise IT is precise data integration, data cleaning, and lineage tracing.
- A human data architect must be in the loop.
  - At best there's a weak description of data semantics
  - So there's a limit to how much tools can help
  - There's no hope of full automation anytime soon



# The Good News

- Attractive GUI's for mapping definition
  - You don't need to type SQL, XQuery, XSLT, etc.
- Useful partial-automation
  - Schema matching
  - Target schema generation
  - Mapping generation
  - Code generation from mapping
- Components for partial-automation appear to be reusable in many scenarios (see slide 2)
  - Very important, given the difficulty of the engineering
  - But reusability hasn't been validated





# Commercial Systems

- MS ADO.NET V3: ER-to-SQL mapping system
  - Supports inheritance and complex types
  - Queries and updates via eSQL
- MS Dynamics & CRM:
  - Metadata-driven generation of class wrappers
  - Most ERP apps use metadata-driven wrappers
- ETL tools, e.g., MS SQL Server Integration Services
- Many visual prog. tools to generate transformations
  - MS BizTalk Mapper: XSLT
  - BEA AquaLogic Data Services: Generates SQL, XQuery
  - IBM Rational Data Architect: Mapping discovery, and generates SQL, SQL/XML, XQuery, XSLT

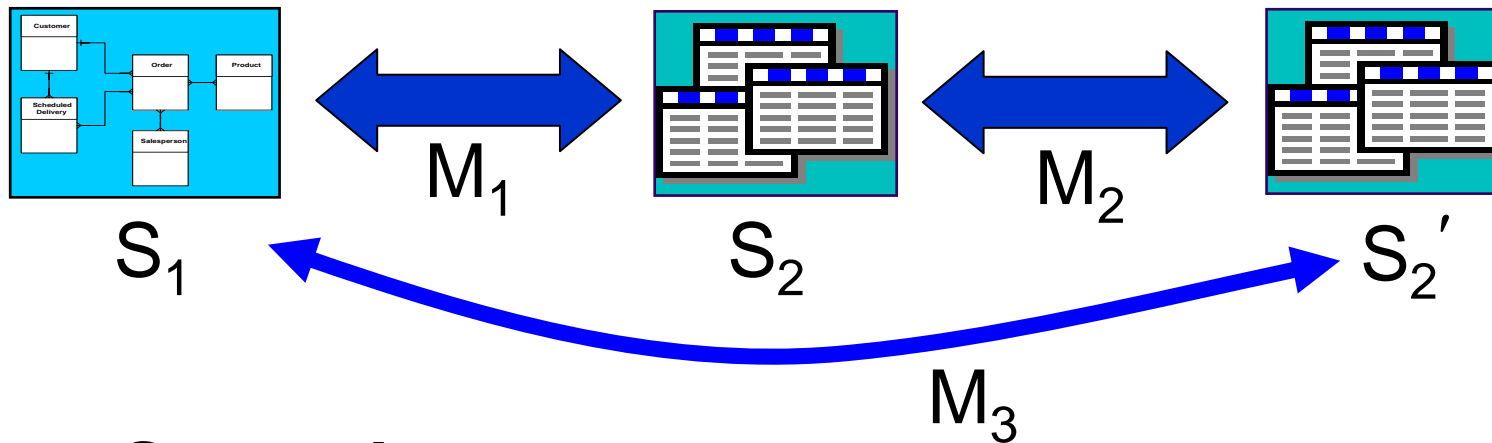


# The Bad News

- Mapping languages need to be more expressive
- Large schemas & mappings are hard to visualize
- Need help in validating mappings w.r.t. constraints
- Need help to round-trip incremental modifications
- Need help in automating schema evolution

# Schema Evolution

## Evolve Mapping

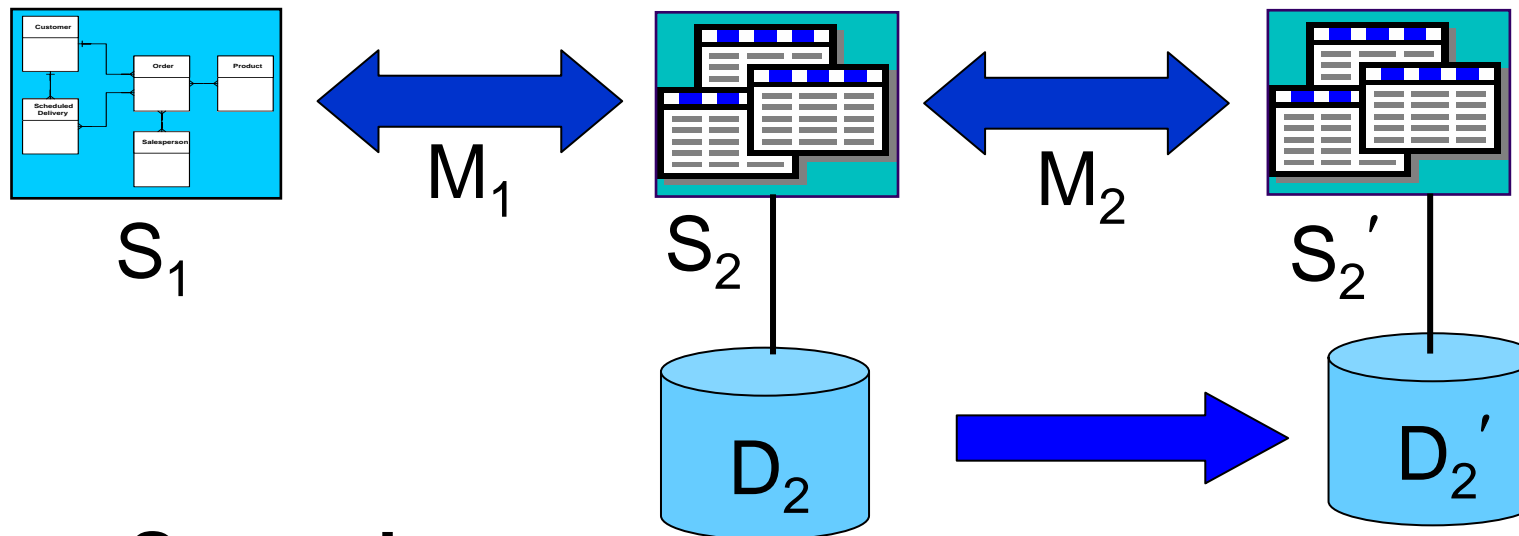


### User Scenarios

- Evolve mapping:  $M_1$  to  $M_3$ :  $S_1 \rightarrow S_2'$

# Schema Evolution

## Migrate Data

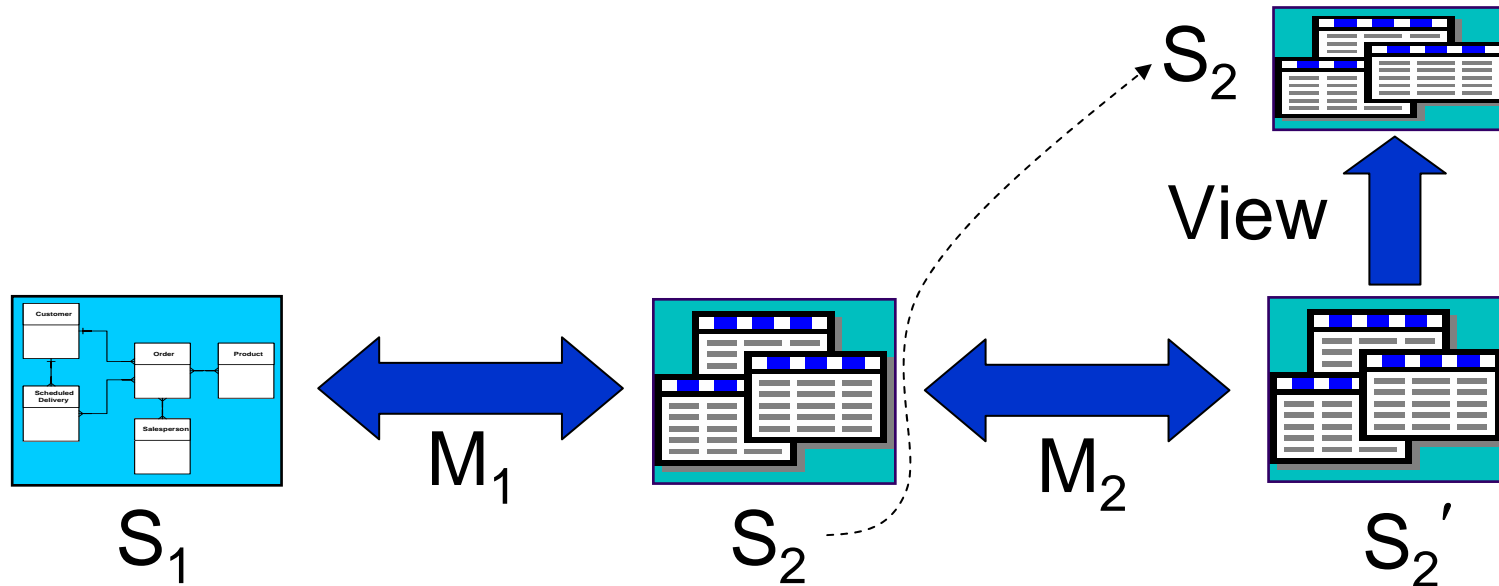


### User Scenarios

- Evolve mapping:  $M_1$  to  $M_3$ :  $S_1 \rightarrow S_2'$
- Migrate data:  $S_2$ 's database to  $S_2'$

# Schema Evolution

## Enable Backward Compatibility

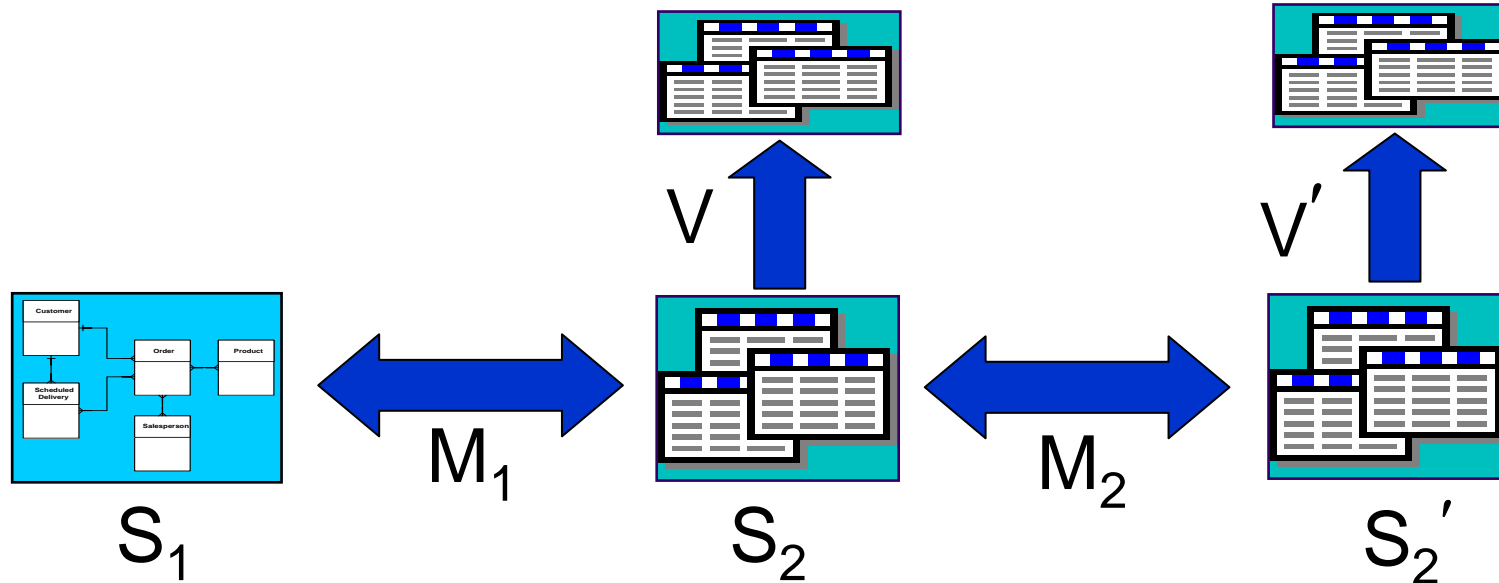


### User Scenarios

- Evolve mapping:  $M_1$  to  $M_3$ :  $S_1 \rightarrow S_2'$
- Migrate data:  $S_2$ 's database to  $S_2'$
- Enable backward compatibility: create view  $S_2$  over  $S_2'$

# Schema Evolution

## Evolve Views

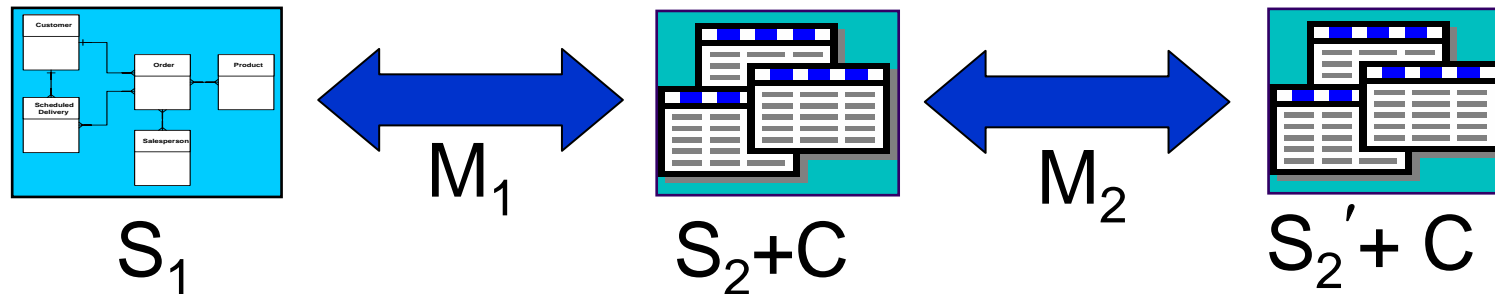


### User Scenarios

- Evolve mapping:  $M_1$  to  $M_3$ :  $S_1 \rightarrow S_2'$
- Migrate data:  $S_2'$ 's database to  $S_2$
- Enable backward compatibility: create view  $S_2$  over  $S_2'$
- Evolve a view:  $V$  to  $V'$ :  $S_2' \rightarrow S_2$

# Schema Evolution

## Don't break customizations



### User Scenarios

- Evolve mapping:  $M_1$  to  $M_3$ :  $S_1 \rightarrow S_2'$
- Migrate data:  $S_2$ 's database to  $S_2'$
- Enable backward compatibility: create view  $S_2$  over  $S_2'$
- Evolve a view:  $V$  to  $V'$ :  $S_2' \rightarrow S_2$
- Do all of the above for  $S_2'$  but don't break customizations  $C$  by ISVs, VARs and users



# Other Factors

- Sharing very large data sets
- Mixing domain and data integration expertise
- Security and privacy
- Culture of data sharing
- Recognition and rewards
- Standards
- Policy