Microsoft Research

Data Integration Technology

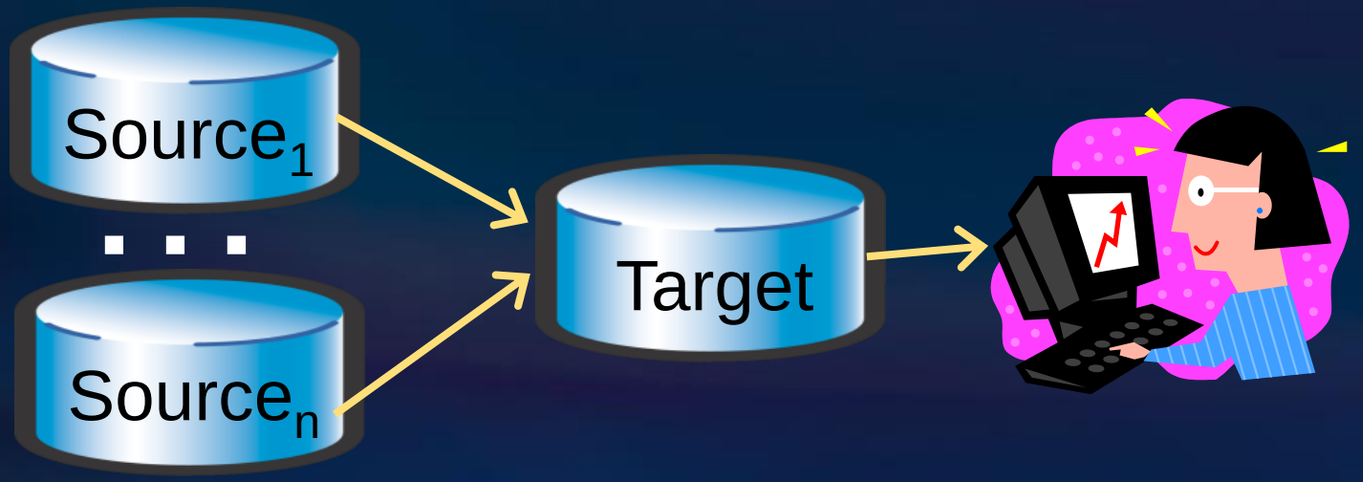
For AstroInformatics 2012, Redmond, WA

Phil Bernstein
Microsoft Research

September 10, 2012

Problem Statement

- Information integration is the task of combining information from different sources and presenting a unified view of that information.



- Input: raw data arriving, e.g. from instruments
  - Possibly heterogeneous, from different instruments and independent research groups
- What do you have to do to get to the point where you can work with the data?

Data Preparation

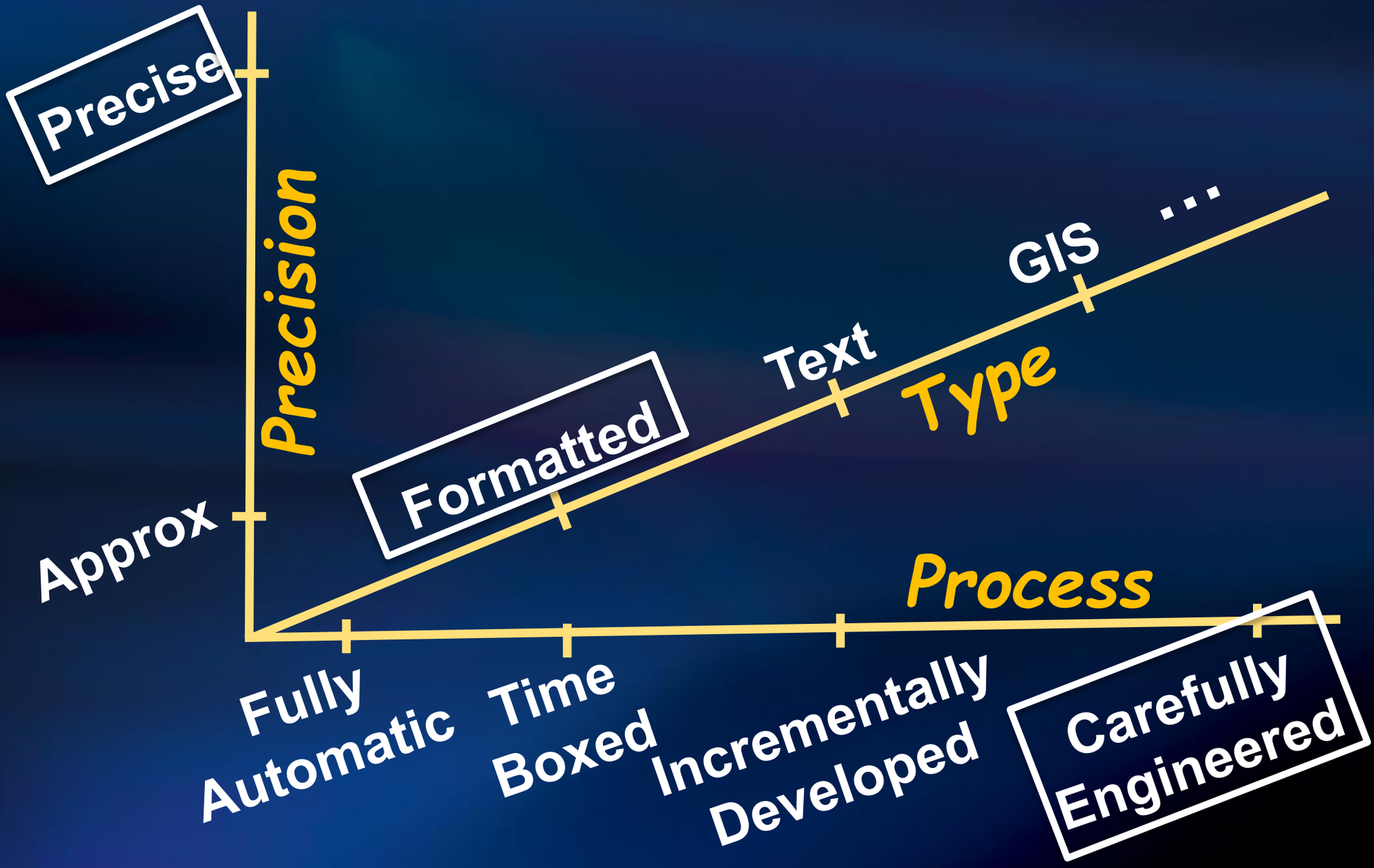
- Data preparation – everything you do to data before you start using it to solve your problem
- Data preparation is hard and expensive
- It's often as much work as writing programs to use the data

Typical Data Integration Scenarios

- Data translation
- XML message mapping
- Data warehouse loading
- Query mediators
- Portal to access a DB
- Object-relational wrappers
- Report writers
- Query designers
- Forms managers
- Composing web services

- What's common about them? They all involve mapping between two representations of the data.

Mapping Space



Typical Scenarios

- Data integration scenarios involve one or more source databases, a target database, and a mapping in between

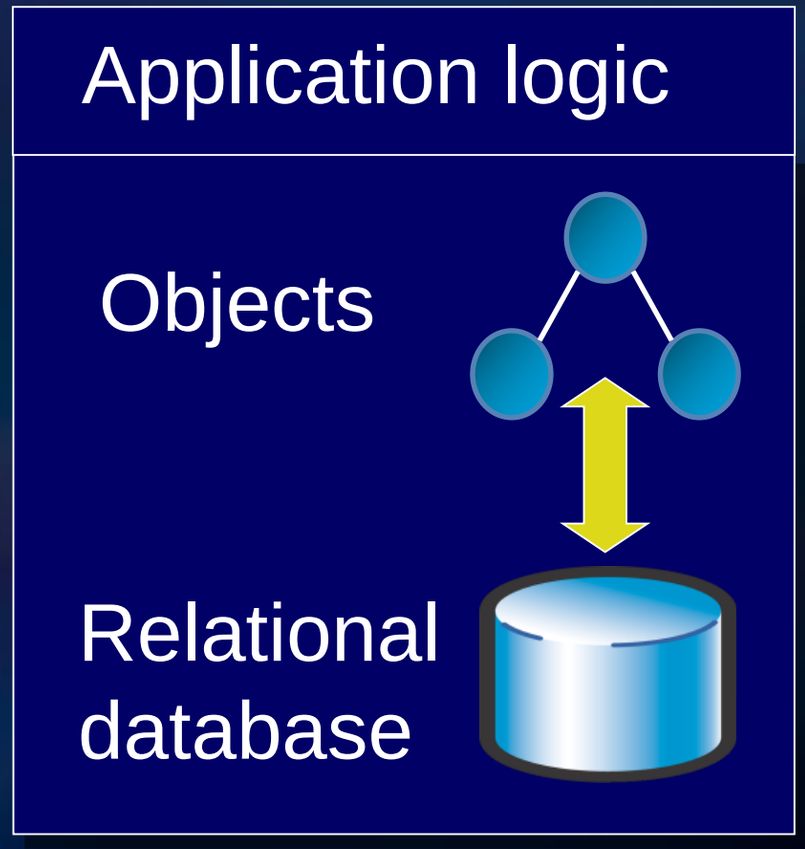
- Given sources, generate a target and mapping
- Given sources and target, generate a mapping

How to Express Mappings

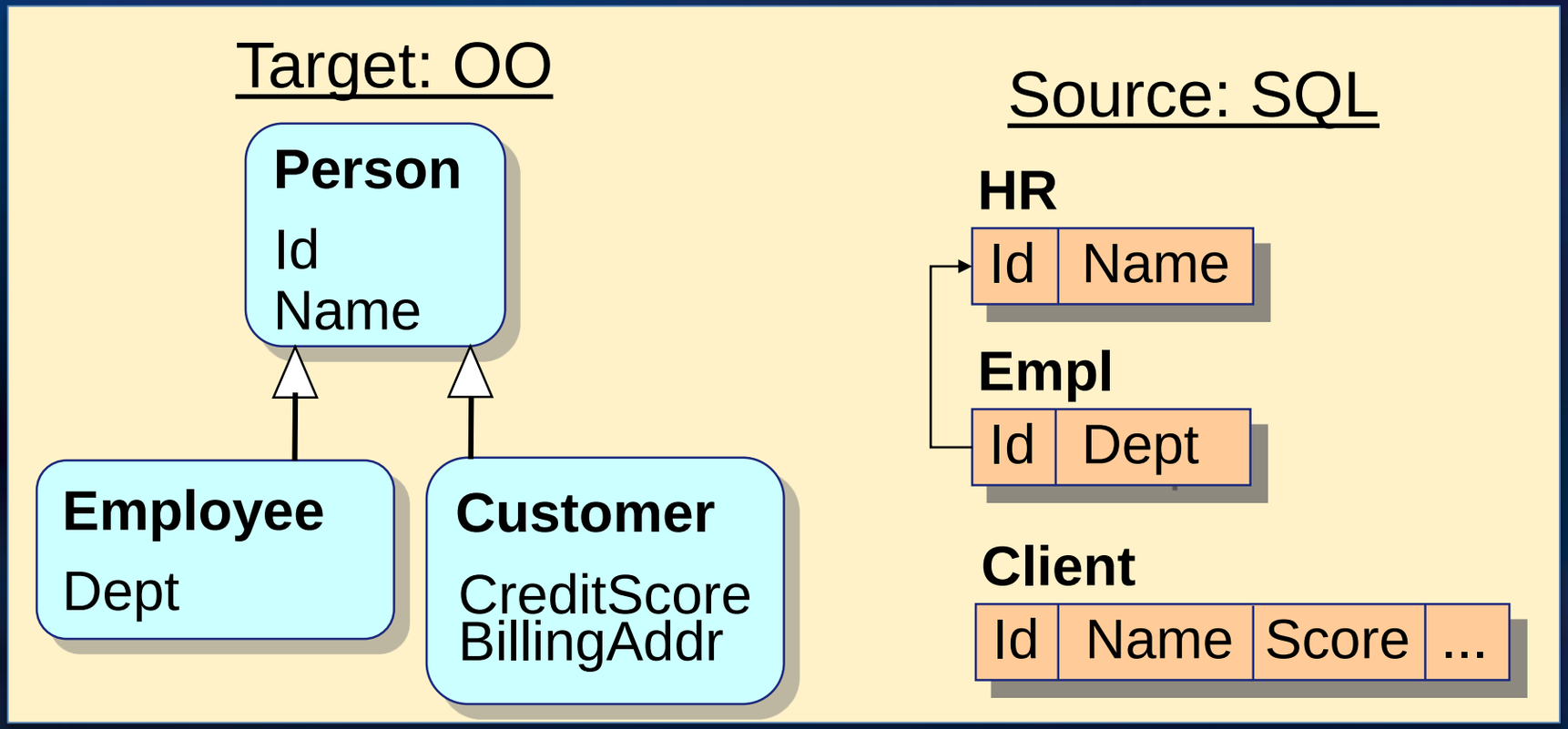
- Ideally, a mapping is
  - Abstract (i.e., short)
  - Easy to express
  - Has clear semantics
  - And compiles into an executable query / view / program to transform data
- SQL is better than Java, C#, Fortran, …
- But even higher level languages are needed

Object-Relational Wrappers

- Object-oriented apps need to access an object-oriented view of relational data
- Requires an object-to-relational mapping



An Example Mapping




```
SELECT p.Id, p.Name
FROM Person
=
SELECT Id, Name FROM HR
Union
SELECT Id, Name FROM Client
```

```
SELECT * FROM Employee = SELECT * FROM HR JOIN Empl
```

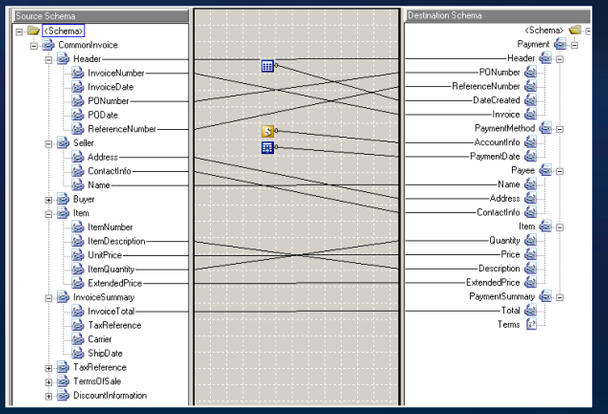
```
SELECT * FROM Customers = SELECT * FROM Client
```

Constructing Persons

```
SELECT VALUE
    CASE
      WHEN (T5._from2 AND NOT(T5._from1))   THEN Person(T5.Person_Id, T5.Person_Name)
      WHEN (T5._from1 AND T5._from2)
        THEN Employee(T5.Person_Id, T5.Person_Name,  T5.Employee_Dept)
        ELSE Customer(T5.Person_Id, T5.Person_Name,  T5.Customer_CreditScore,
             T5.Customer_BillingAddr)
    END
FROM (   (SELECT T1.Person_Id, T1.Person_Name,  T2.Employee_Dept,
                 CAST(NULL AS SqlServer.int) AS Customer_CreditScore,
                 CAST(NULL AS SqlServer.nvarchar) AS Customer_BillingAddr, False AS _from0,
                 (T2._from1 AND T2._from1 IS NOT NULL) AS _from1,  T1._from2
          FROM  ( SELECT  T.Id AS Person_Id, T.Name AS Person_Name, True AS _from2
                  FROM HR AS T) AS T1
                  LEFT OUTER JOIN (
                    SELECT  T.Id AS Person_Id,  T.Dept AS Employee_Dept, True AS _from1
                    FROM dbo.Empl AS T) AS T2
                  ON T1.Person_Id = T2.Person_Id )
          UNION ALL (
                 SELECT  T.Id AS Person_Id,  T.Name AS Person_Name,
                         CAST(NULL AS SqlServer.nvarchar) AS Employee_Dept,
                         T.Score AS Customer_CreditScore,  T.Addr AS Customer_BillingAddr,
                         True AS _from0,  False AS _from1,  False AS _from2
                 FROM Client AS T)
      ) AS T5
```

Mappings

- Element correspondences
  - First step in aligning schemas
  - For lineage & impact analysis
  - Weak or no formal semantics

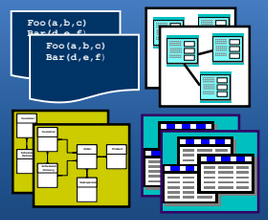


- Mapping constraints relate instances of schemas
  - E.g., equality of relational expressions

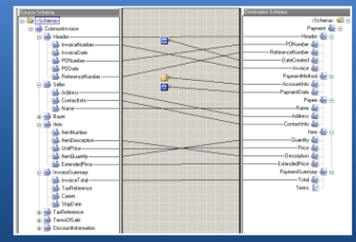
```
SELECT Id, Name, Dept  =  SELECT Id, Name, Dept
FROM   Employee           FROM HR JOIN Empl ON Id
```

- Transformation is an executable mapping constraint
  - Constructs target instances from source instances
  - E.g., SQL query, XSLT, C# program

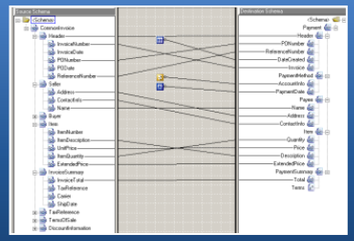
Code Generation Process



Schemas



Correspondences



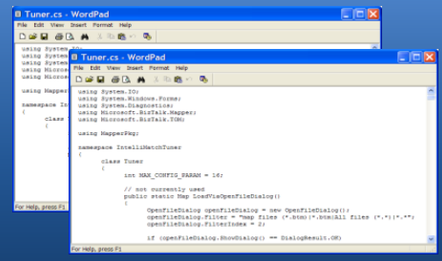
Correspondences

Select ord#, prod#, cust#
From Shipped
=
Select ord#, prod#, cust#
From Order Join Item
on ord#

Constraints

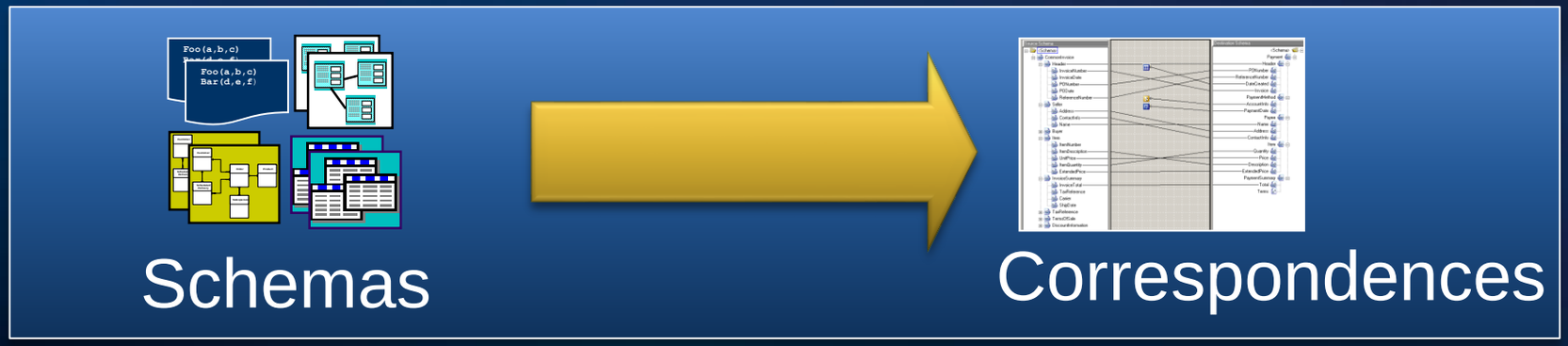
Select ord#, prod#, cust#
From Shipped
⊆
Select ord#, prod#, cust#
From Order Join Item
on ord#

Constraints

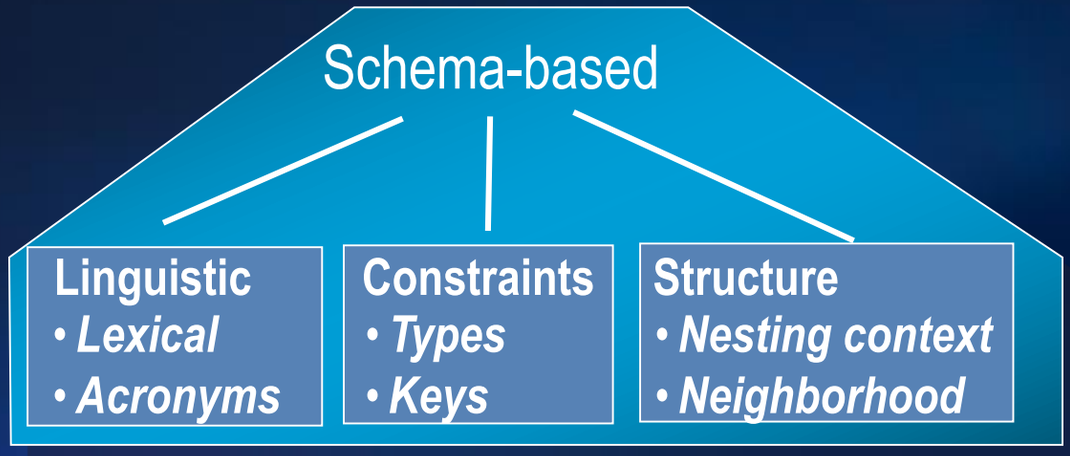


Transformations

Schema Matching



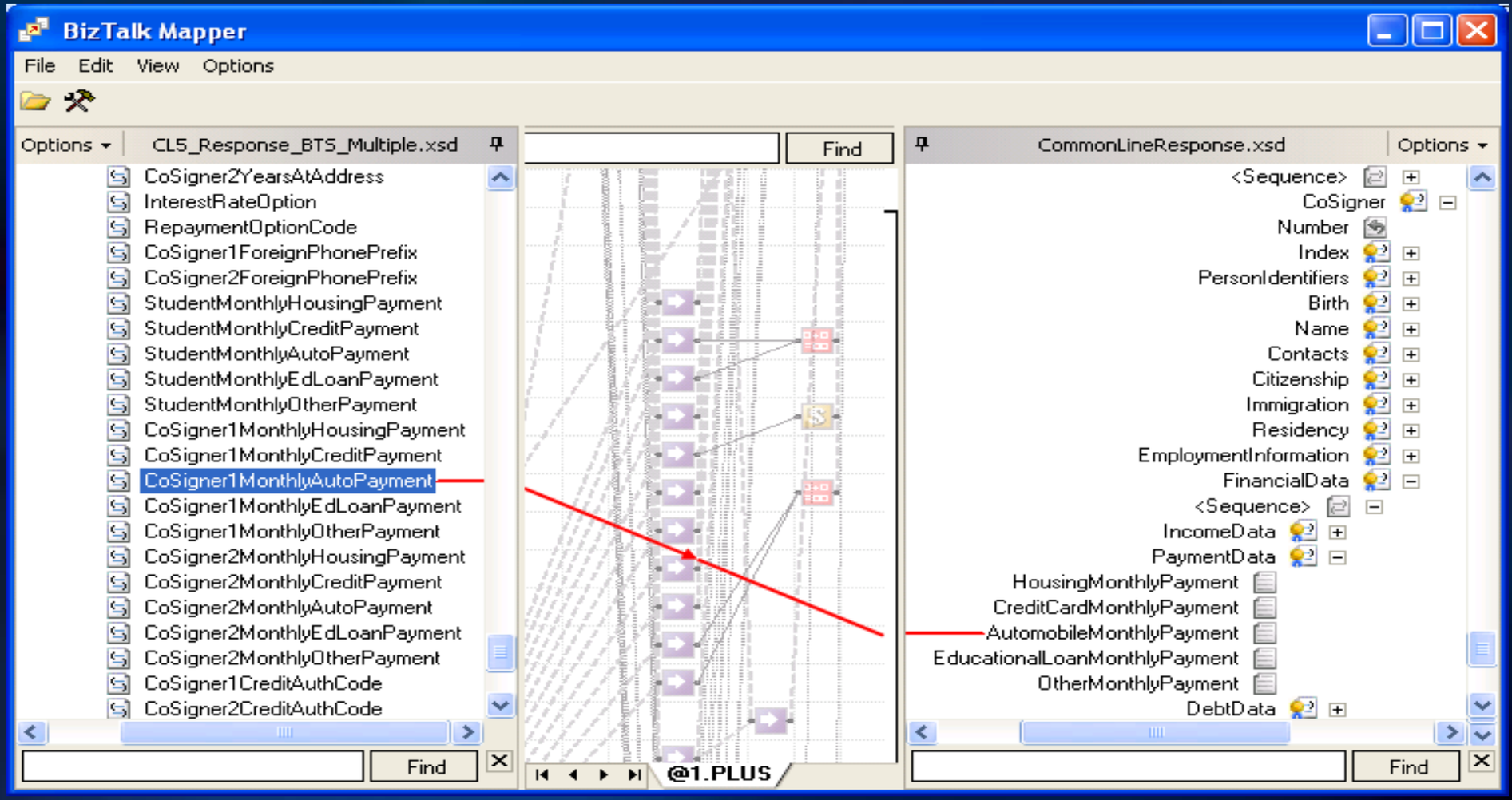
- Match operator
  - Input: two schemas, auxiliary information
  - Output: correspondences



Reuse-based
- *Thesaurus*
- *Validated matches*

Content-based
- *Values, patterns*
- *Machine learning*

Matching Tool



Correspondences → Constraints

- Constraint generation
  - Input: two schemas & correspondences
  - Output: mapping constraints

- Should be an automated compilation step
  - In effect, the correspondences are a visual programming language for generating constraints

Constraints → Transformations

- Transformation generation
  - Input: two schemas & mapping constraints
  - Output: query views and update views

- This too should be an automated step

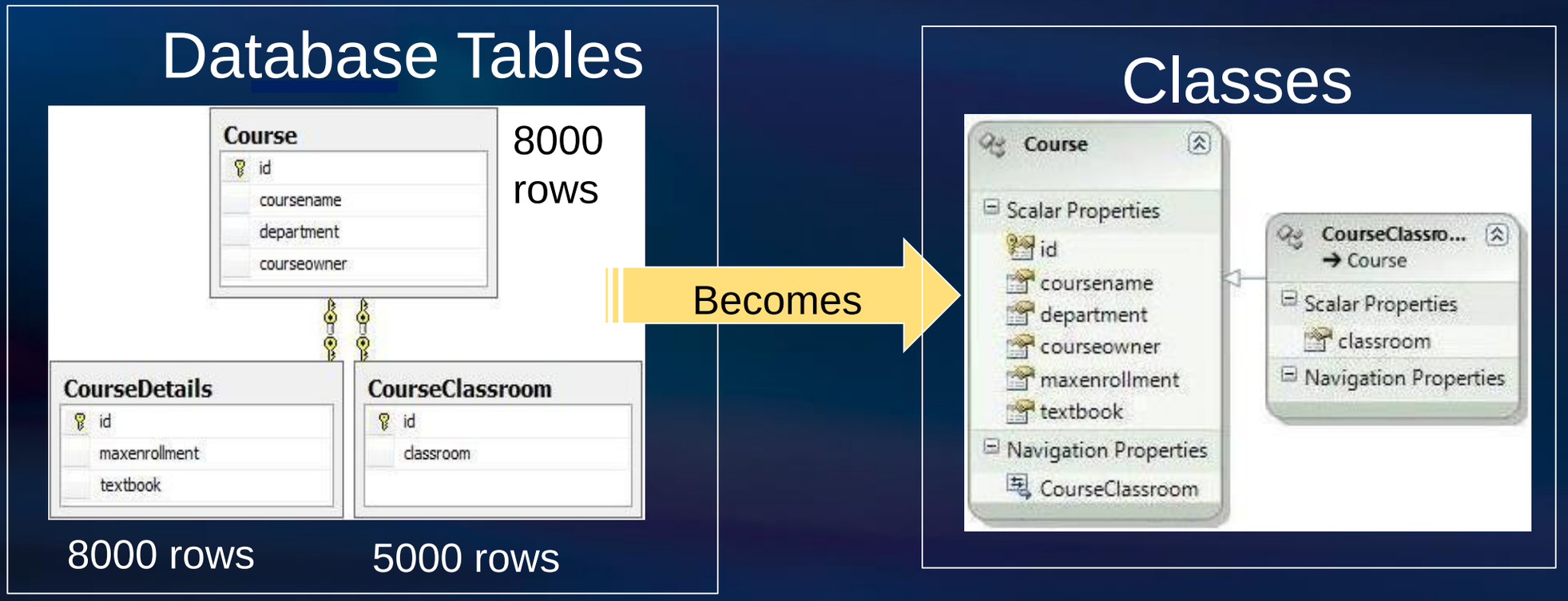
Object-to-Relational Products

- Microsoft's ADO.NET Entity Framework
- Hibernate
- Oracle TopLink
- Ruby on Rails

Schema Translation

- Translate a schema to another data model
  - Input: schema in a source data model
    name of target data model
  - Output: schema in the target data model
    mapping between the schemas
- Examples
  - Object-oriented schema → SQL schema
  - SQL schema → object-oriented schema
  - XML schema → SQL schema

Tables to Classes

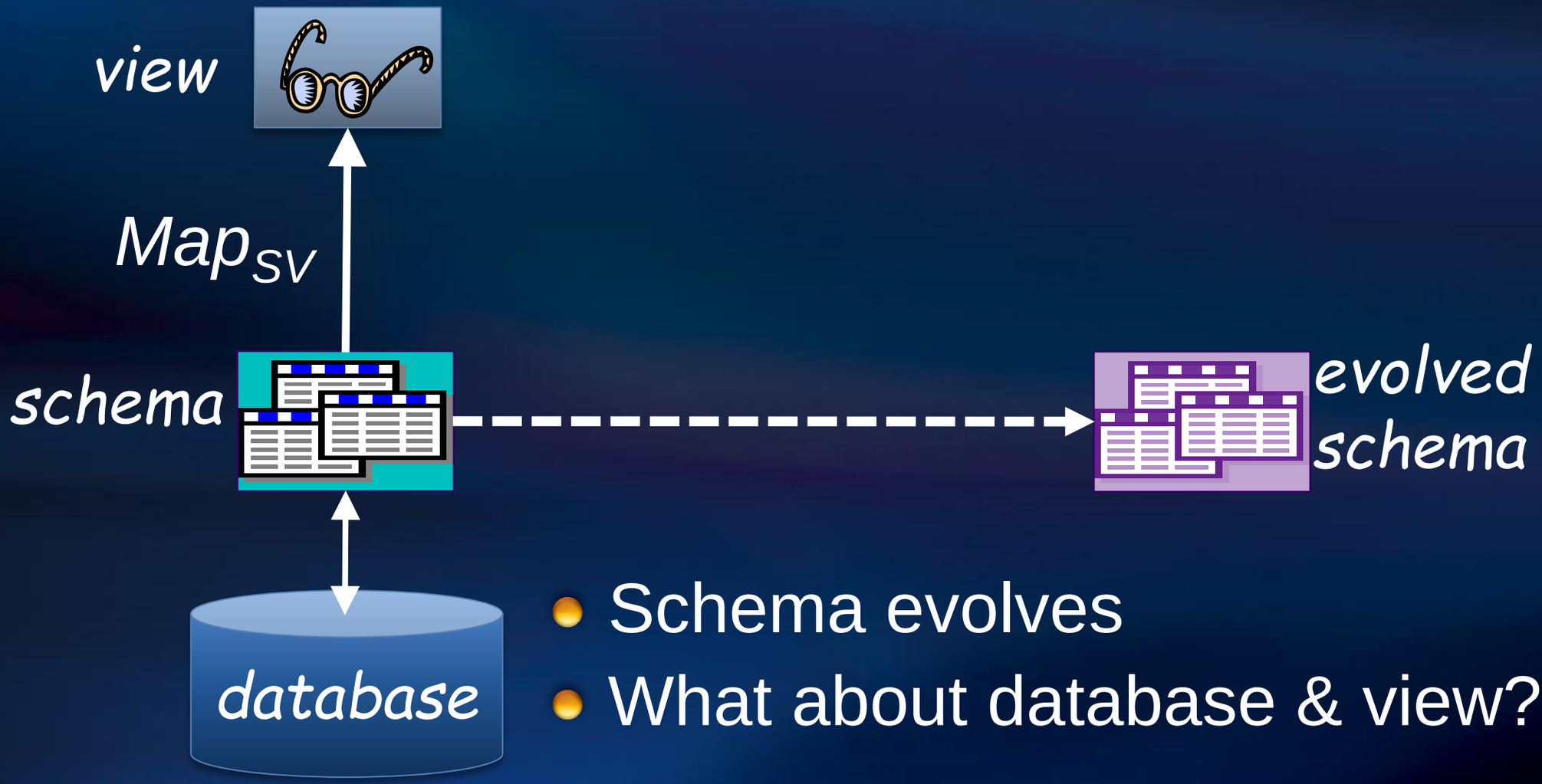


- | Course | = | CourseDetails | and 1:1 foreign key implies a partitioned class
- | Course | > | CourseClassroom | and 1:1 foreign key implies a subtype

Model Management

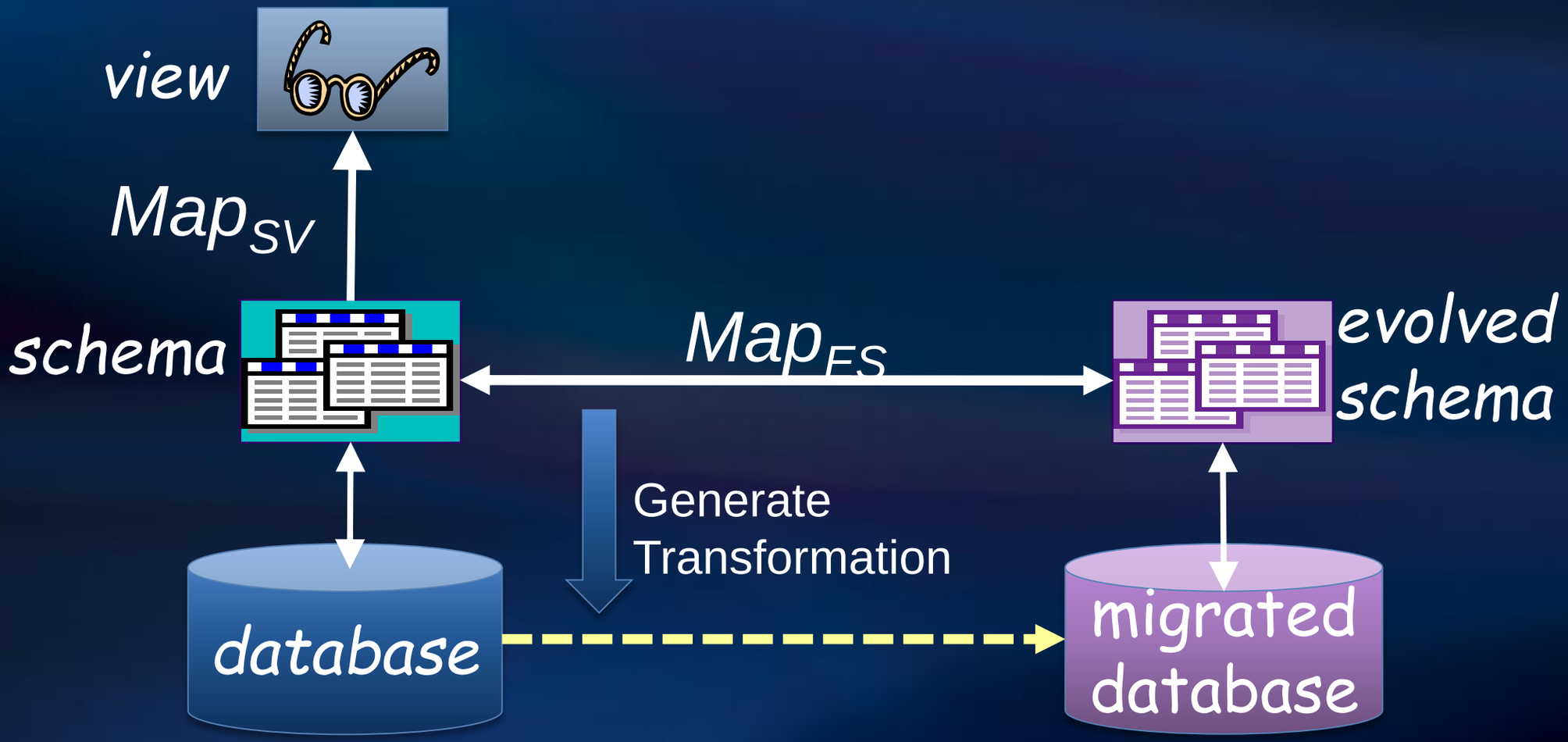
- A set of operators that manipulate schemas and mappings as bulk objects
- You just saw operators to
  - match schemas
  - generate constraints and transformation
  - translate schemas between data models
- Other operators
  - Merge two schemas
  - Diff two schemas
  - Compose two mappings
  - Invert a mapping

Schema Evolution



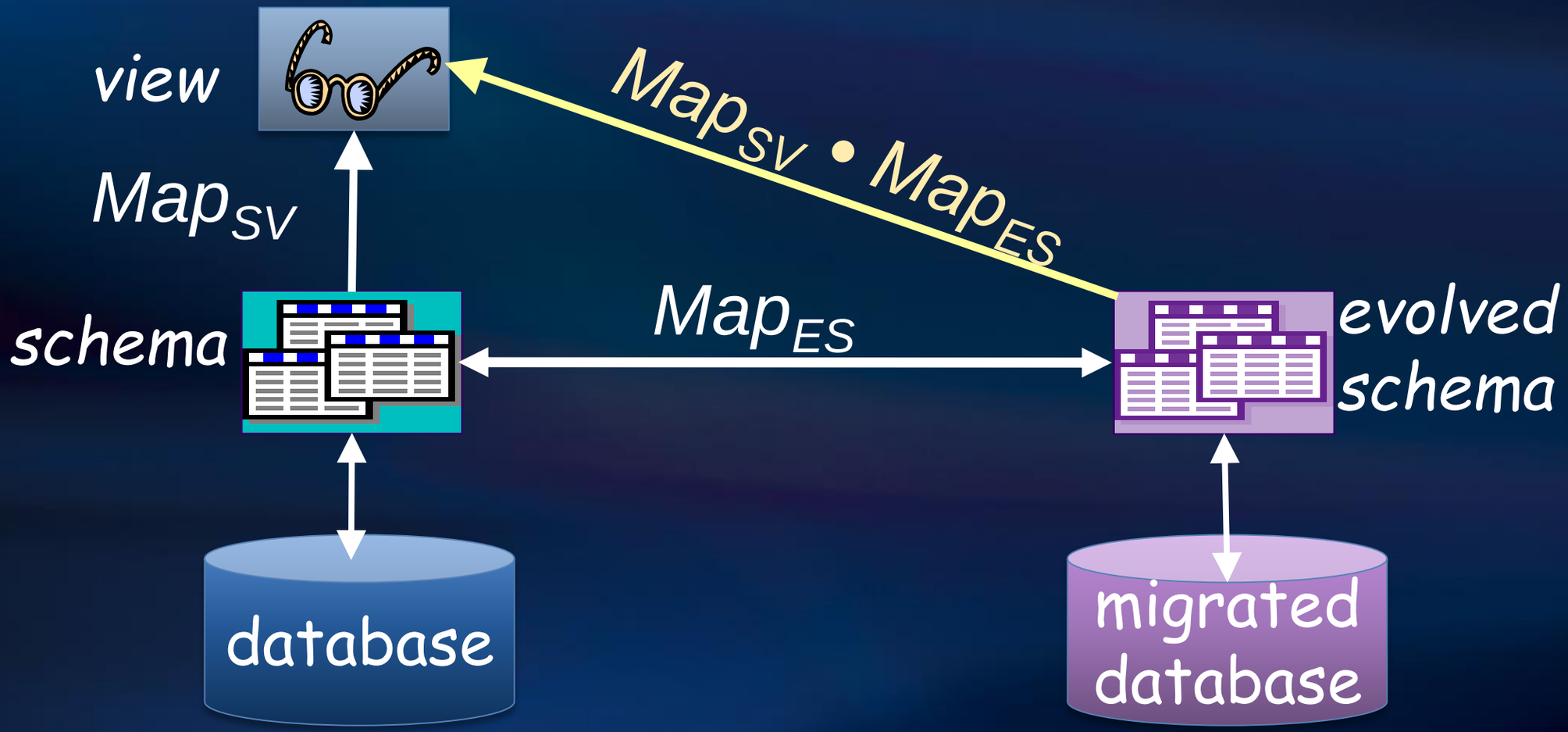
- Schema evolves
- What about database & view?

Data Migration



1. Create mapping: *schema* ⇔ *evolved schema*
2. Generate a transformation

View Migration



- Compose Map_{SV} and Map_{ES} to connect *view* to *evolved schema*

Instance-level Tools

- Integrating data also involves cleaning and transforming instances
- Extract, Transform, and Load (ETL) tools for data warehousing include algorithms to help
- Examples
  - Data profiling
  - Fuzzy lookup
  - Parsing text into formatted records
  - Fuzzy group-by (de-duplication)

Data Profiling

- Derive summaries of data
  - Keys – single-column or multi-column
  - Foreign Keys
  - Value ranges
- Can be exact or approximate
  - Based on exhaustive analysis or sampling
- A challenge is how to do this efficiently

Fuzzy Lookup

Microsft Win XP Home

Company	Product	ID
Microsoft	Windows 2003 Server	1000
Microsoft	Windows XP Home	2000
…	…	…

Parsing

- Segment a string into a given target schema
- Leverages ref tables and regular expressions

Customer
John Smith One Microsoft Way Redmond WA 98052
Bob Jones 8599 188th Ave NE Redmond 98052
Wendy Smith 10258 NE 124th St Bellevue WA

Name	Street Address	City	State	Zip
John Smith	One Microsoft Way	Redmond	WA	98052
Bob Jones	8599 188th Ave NE	Redmond		98052
Wendy Smith	10258 NE 124th St	Bellevue	WA	

Fuzzy Group By

RID	Organization Name	Address
1	Msft Inc	One Microsoft Way
2	Microsoft Corporation	One Msft Way
3	Microsoft Corp.	1 Microsoft Way
4	Boeing Corporation	1525 Pacific Ave
5	Beoing Corporation	1525 Pacific Avenue

- This is the basis of de-duplication
- Often uses domain-specific information
  - Abbreviations, formatting conventions
  - E.g. address de-duplication

Summary

- To integrate data, you need to define mappings
- You need
  - high level mapping language
  - powerful operators to generate and manipulate models and mappings
- After developing mappings you need to maintain them as data and programs evolve
- You also need tools to clean your data

Microsoft Research

Microsoft Research

Further Reading

- http://research.microsoft.com/~philbe
- S. Melnik, A. Adya, P.A. Bernstein: “Compiling mappings to bridge applications and databases.” ACM Transactions on Database Systems 33(4): (2008).
- P.A. Bernstein, L.M. Haas: “Information integration in the enterprise.” Commun. ACM 51(9): 72-79 (2008)
- P.A. Bernstein, S. Melnik: “Model management 2.0: manipulating richer mappings.” ACM SIGMOD Conf. 2007.
- L.M. Haas, M.A. Hernández, H. Ho, L. Popa, M. Roth: “Clio grows up: from research prototype to industrial tool.” ACM SIGMOD Conf. 2005

Microsoft®

Your potential. Our passion.™