# Schema and data translation

**Paolo Atzeni**
Dipartimento di Informatica e Automazione
Università Roma Tre, Italy
http://atzeni.dia.uniroma3.it/

Tutorial at ICDE 2006

# Outline

- **Introduction and motivation**
- Model management
- Schema and data translation
- Models, schemas, mappings
- Information capacity dominance and equivalence
- Schema translation
- Data exchange

# Databases in the Internet era

- Databases before the Internet
  - An internal infrastructure, a precious resource, but usually hidden, with some controlled cooperation
- Internet changes the requirements
  - More users (not only humans), more diverse cooperating systems (distributed, heterogeneous, autonomous), more types of data
- "Future" Internet changes more
  - New devices (embedded everywhere), even more users (many “per person”), real mobility, need for personalization and adaptation

# A ten-year goal for database research

- The “Asilomar report” (Bernstein et al. Sigmod Record 1999 www.acm.org/sigmod):
  - *The information utility: make it easy for everyone to store, organize, access, and analyze the majority of human information online*

# A general framework: cooperation

- "The capacity of a system to interact (effectively) with other systems, possibly managed by different organizations"

# Forms of cooperation

- Process-centered cooperation:
  - the systems offer services one another, by exchanging messages, information or documents, or by triggering activities, without making remote data explicitly visible
- Data-centered cooperation:
  - the systems offer data one another; data is distributed, heterogeneous and autonomous, and accessible from remote locations according to some co-operation agreement

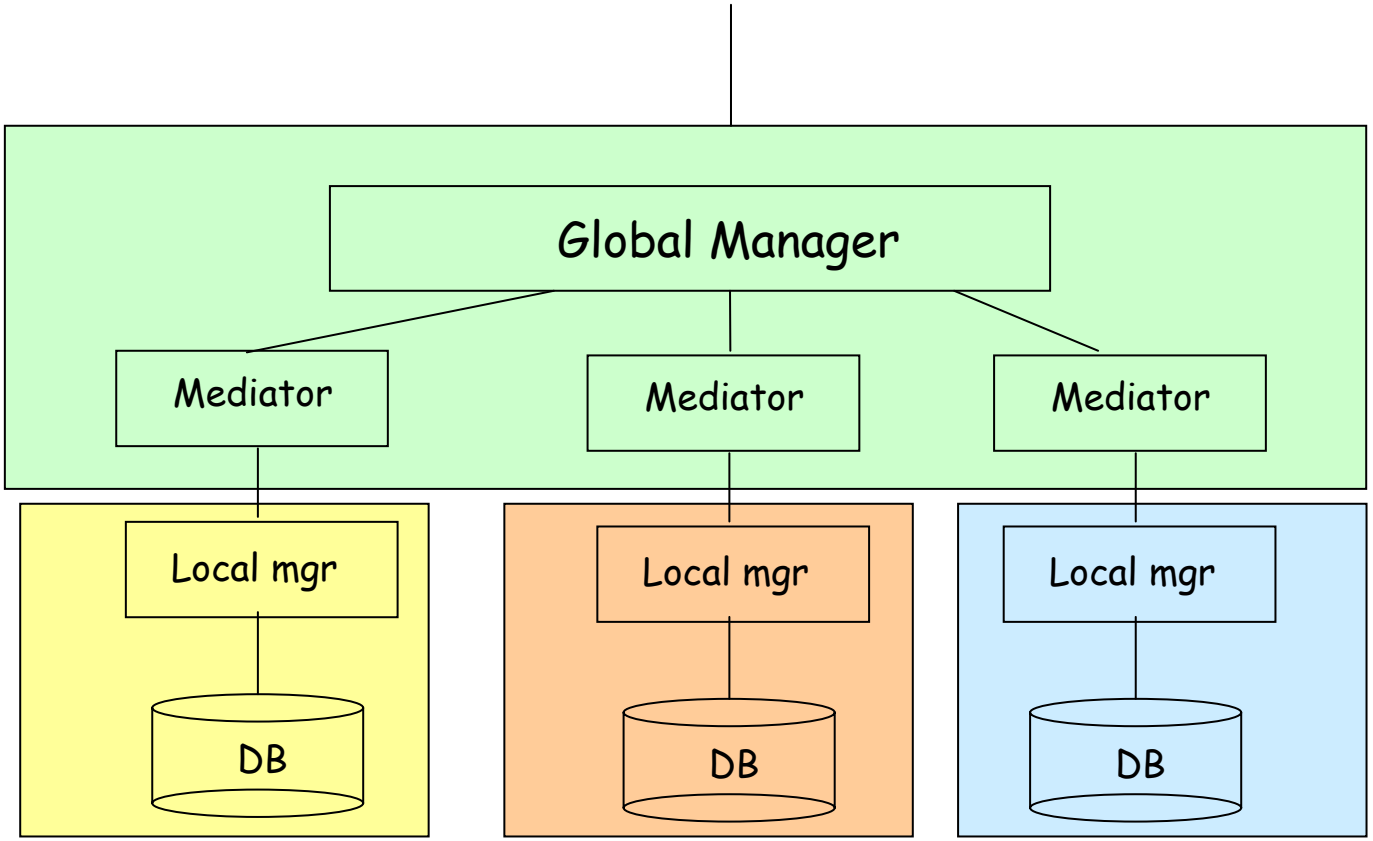
# The most studied form of data-centered cooperation: integration

- We are interested in data-centered cooperation

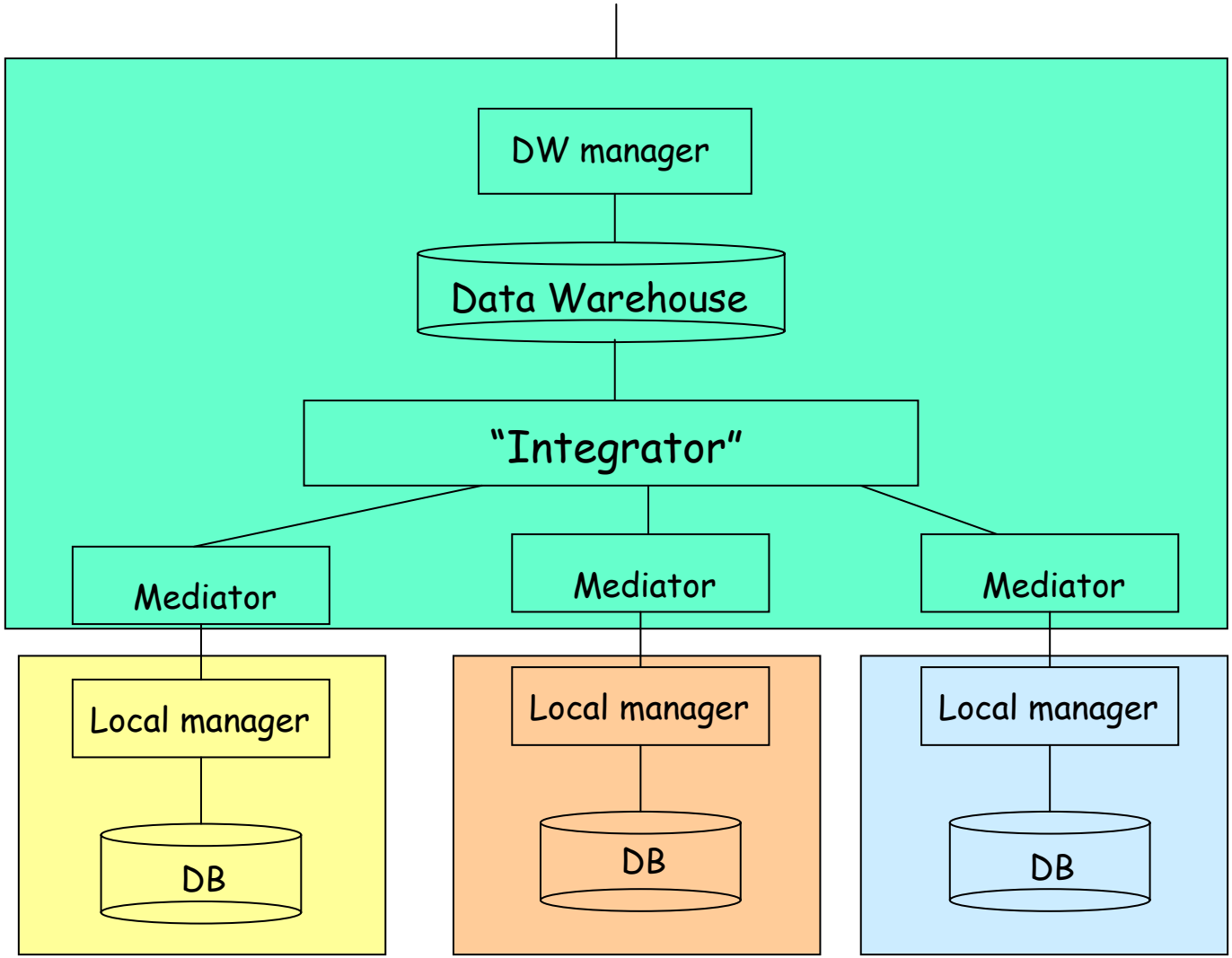
"The unification of related, heterogeneous data from disparate sources, for example, to enable collaboration" (Hammer & Stonebraker 2005)

- Some "paradigms" …

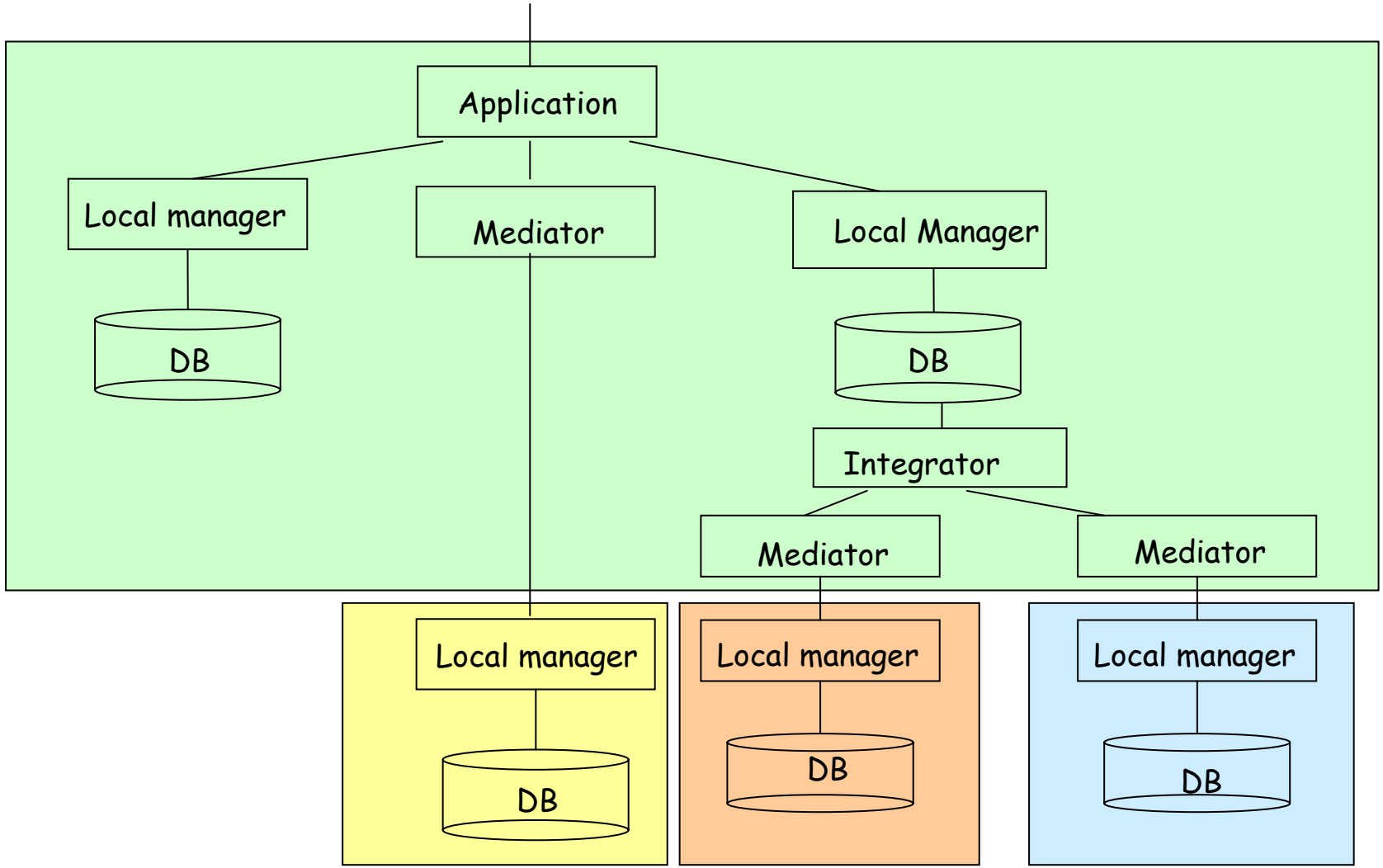
# Multidatabase



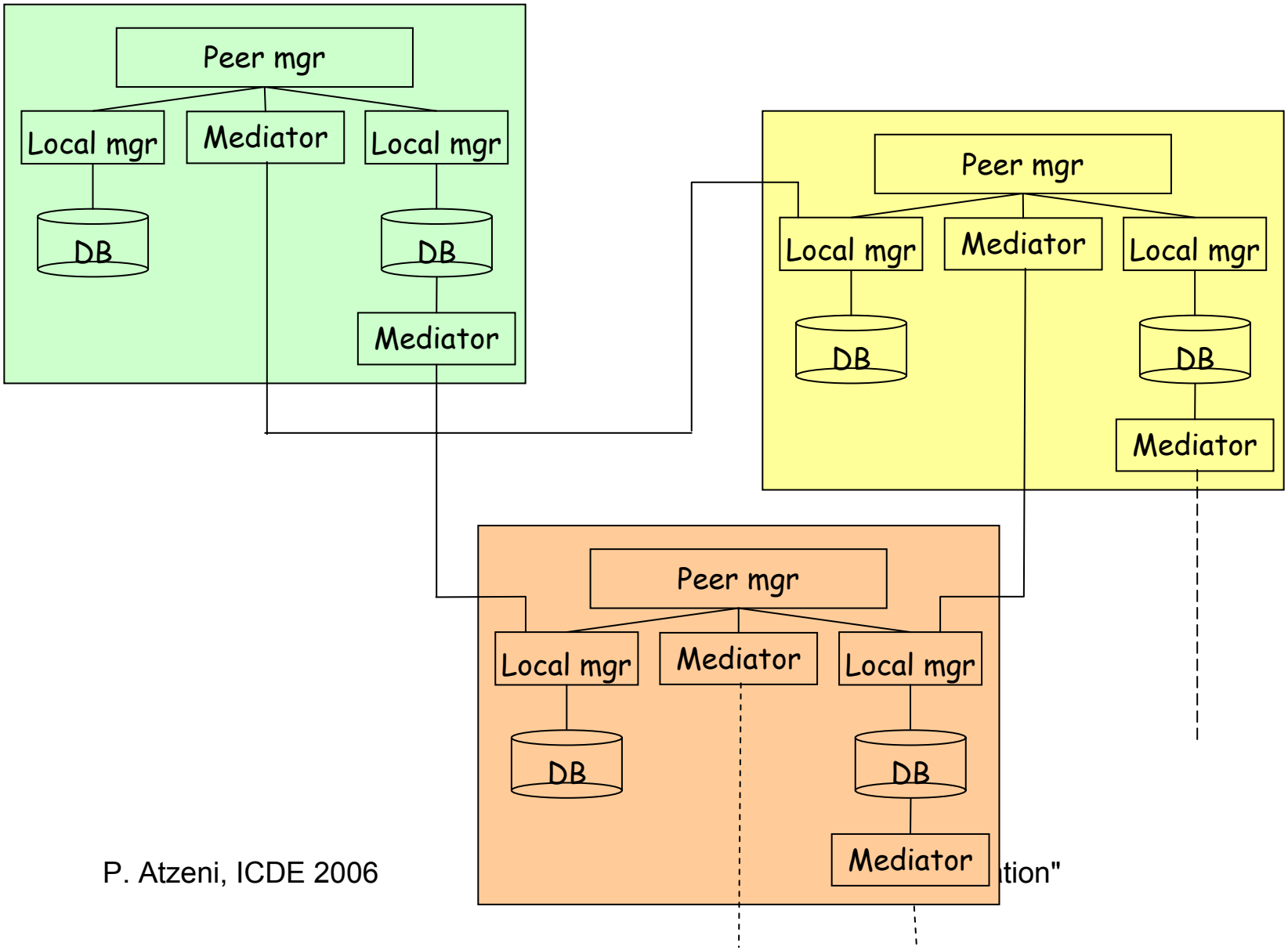
# Data Warehousing System



# Intermediate solutions in practice



# Peer-based architecture



# Data is not just in databases

- Mail messages
- Web pages
- Spreadsheets
- Textual documents
- Palmtop devices, mobile phones
- Multimedia annotations (e.g., in digital photos)

- and, more important, XML documents

# The same data in the same form?

- Adaptivity:
  - Personalization: content adapted to the user
    - upon system's decision
    - upon user's request
  - Customization: structure adapted to the user
    - according to the user's role
    - upon user's request
  - Context dependence
    - User, Device, Network, Place, Time, Rate

# A general need

- We have data at various places, and data has to be
  - exchanged
  - replicated
  - migrated
  - integrated
  - adapted

# A major difficulty

- Heterogeneity
  - System level
  - Model level
  - Structural (different structure for similar data)
  - Semantic (different meaning for the same structure)
- Many efforts, but current techniques are mostly manual and *ad hoc*

# A direction for the solutions

- Be general!
  - *ad hoc* solution could work in-the-small, but they
    - are repetitive and time consuming
    - do not scale
    - are not maintainable

- Historical notes:
  - W. C. McGee: Generalization: Key to Successful Electronic Data Processing. J. ACM 1959
  - Indeed, databases are the result of generalization applied to secondary storage management!

# Generality requires …

- … general, high-level descriptions of problems within the family of interest:
  - Metadata:
    - “data about data”
    - (formal or informal) description of structures and meaning

- General solutions leverage on metadata (and then operate on data as a consequence)

# Outline

- Introduction and motivation
- **Model management**
- Schema and data translation
- Models, schemas, mappings
- Information capacity dominance and equivalence
- Schema translation
- Data exchange

# A wider perspective

- (Generic) Model Management:
  - A proposal by Bernstein et al (2000 +)
  - Includes a set of operators on
    - schemas and
    - mappings between schemas

# Terminology: a warning

| Model Mgmt people | Traditional DB people |
|---|---|
| Meta-metamodel | Metamodel |
| Metamodel | Model |
| Model | Schema |

# Schemas and mappings

- More on the issue later
- For the time being:
  - Schema:
    - a set of elements, related in some way to one another
  - Mapping:
    - a set of correspondences (pair of elements) or
    - its reification, a third schema related to the other two via two sets of correspondences

# Model mgmt operators, a first set

- map = Match (S1, S2)
- S3 = Merge (S1, S2, map)
- S2 = Diff (S1, map)
- and more
  - map3 = Compose(map1, map2)
  - S2 = Select(S1, pred)
  - Apply(S, f)
  - list = Enumerate(S)
  - S2 = Copy(S1)
  - …

# Match

- map = Match (S1, S2)
  - given
    - two schemas S1, S2
  - returns
    - a mapping between them
- the “classical” initial step in data integration:
  - find the common elements of two schemas and the correspondences between them

# Merge

- S3 = Merge (S1, S2, map)
  - given
    - two schemas and a mapping between them
  - returns
    - a third schema (and two mappings)
- the “classical” second step in data integration:
  - given the correspondences, find a way to obtain one schema out of two

# Diff

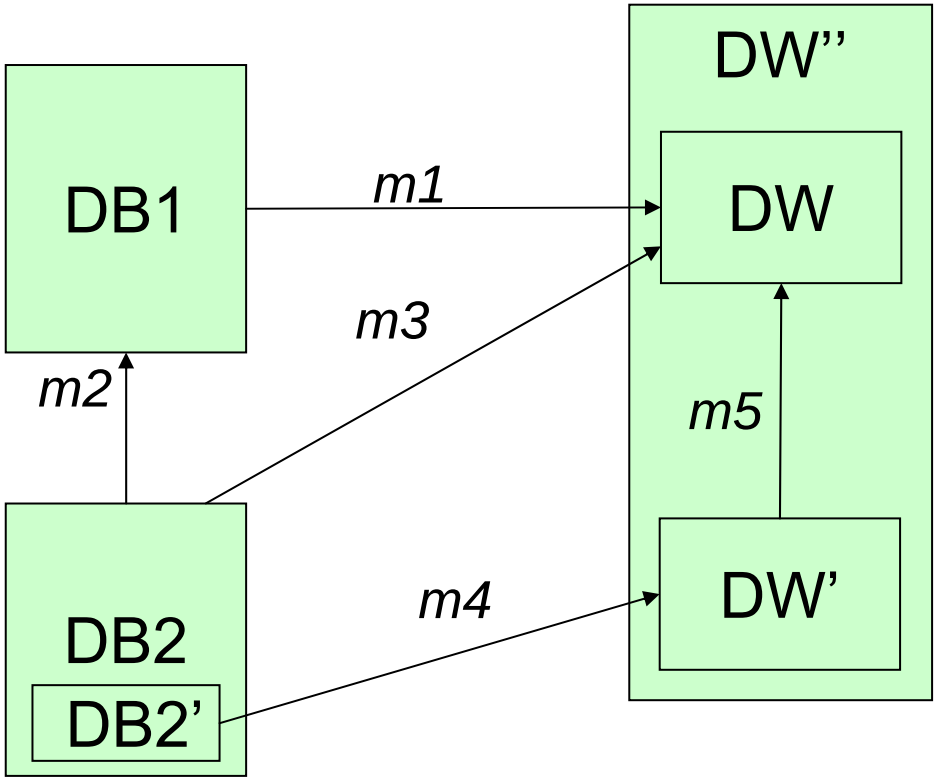
- S2 = Diff (S1, map)
  - given
    - a schema and a mapping from it (to some other schema, not relevant)
  - returns
    - a (sub-)schema, with the elements that do not participate in the mapping

# Example

(Bernstein and Rahm, ER 2000)

- A  database (a “source”), a data warehouse and a mapping between the two
- We want to add a second source, with some similarity to the first one

# Example, the "solution"




m2 = Match(DB1,DB2)

m3= Compose(m2,m1)

DB2'=Diff(DB2,m3)

DW', m4 user defined

m5 = Match(DW,DW')

DW'' = Merge(DW,DW',m5)

# Magic does not exist

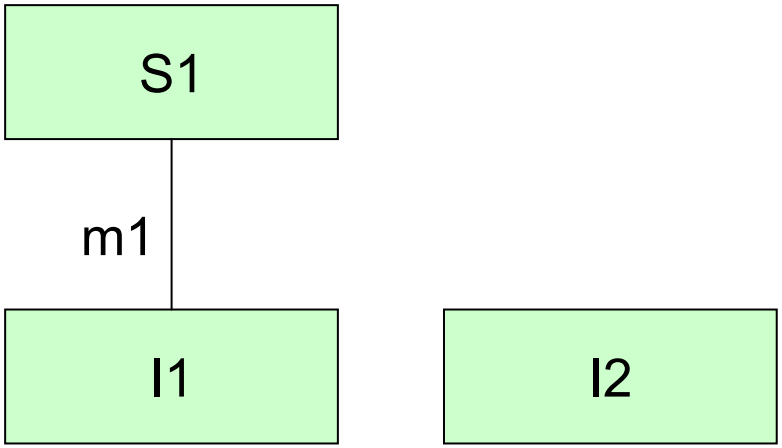
- Operators might require human intervention:
  - Match is the main case
- Scripts involving operators might require human intervention as well (or at least benefit from it):
  - a full implementation of each operator might not always available
  - a mapping might require manual specification
  - incomparable alternatives might exist

# The “data level”

- The major operators have also an extended version that operates on data, and not only on schemas
- Especially apparent for
  - Merge

# We also have heterogeneity

- Round trip engineering (Bernstein, CIDR 2003)
- A specification, an implementation, then a change to the implementation: want to revise the specification
- We need a translation from the implementation model to the specification one



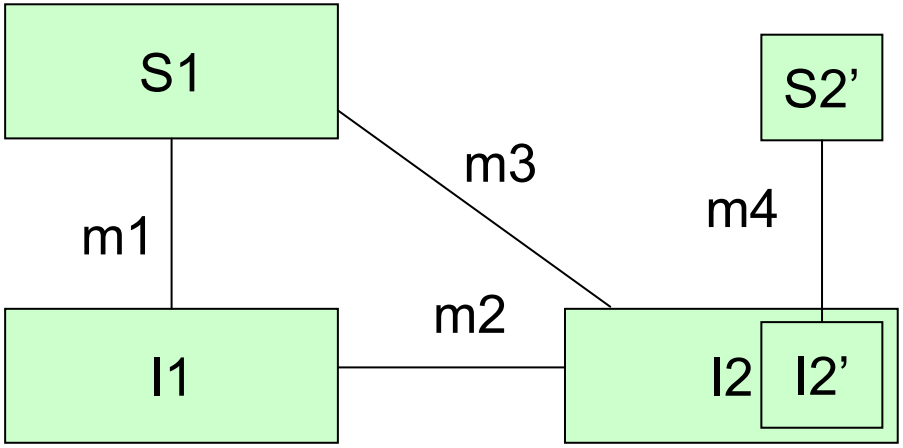
# Model management with heterogeneity

- The previous operators have to be “model generic” (capable of working on different models)
- We need a “translation” operator
  - <S2, map12> = ModelGen (S1)

# ModelGen, an additional operator

- <S2, map12> = ModelGen (S1)
  - given
    - a schema (in a model)
  - returns
    - a schema (in a different data model) and a mapping between the two
- A “translation” from a model to another
- I should call it “SchemaGen” …
- We should better write
  - <S2, map12> = ModelGen (S1,mod2)

# Round trip engineering




m2 = Match (I1,I2)
m3 = Compose (m1,m2)
I2’= Diff(I2,m3)
<S2’,m4 > = ModelGen(I2’)
… Match, Merge

# Outline

- Introduction and motivation
- Model management
- **Schema and data translation**
- Models, schemas, mappings
- Information capacity dominance and equivalence
- Schema translation
- Data exchange

# Schema and data translation, a long standing issue

- Schema translation:
  - given schema S1 in model M1 and model M2
  - find a schema S2 in M2 that “corresponds” to S1
- Schema and data translation:
  - given also a database D1 for S1
  - find also a database D2 for S2 that “contains the same data” to D1

# Schema and data translation, a long standing issue

- Translations from a model to another have been studied since the 1970's
- Whenever a new model is defined, techniques and tools to generate translations are studied
- However, proposals and solutions are usually model specific

# Model specific solutions

- Given an ER schema, find the suitable relational schema that “implements” it
  - the original paper (Chen 1976) contains the basics
  - further discussions by Markowitz and Shoshani 1989
  - illustrated in every textbook

# A similar problem: data exchange

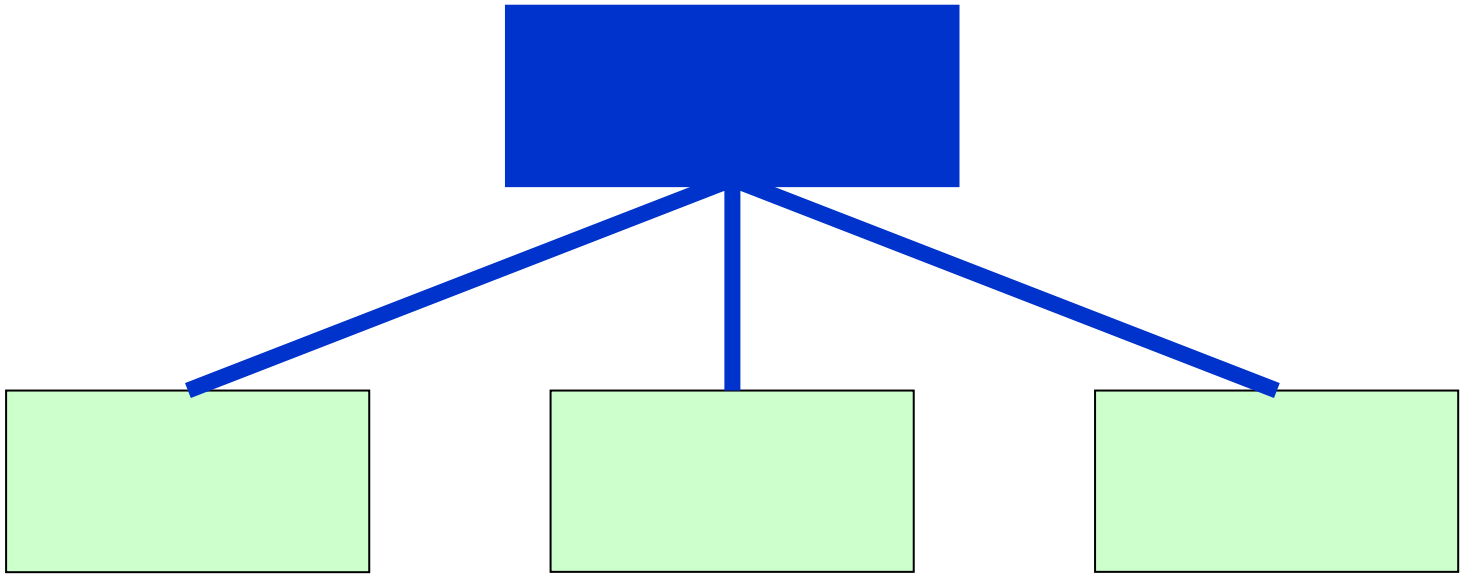
- Given a source S1 and a target schema S2 (in different models or even in the same one), find a translation, that is, a function that given a database D1 for S1 produces a database D2 for S2 that “correspond” to D1

# Summarizing

- Integration
- Schema translation
- Data exchange

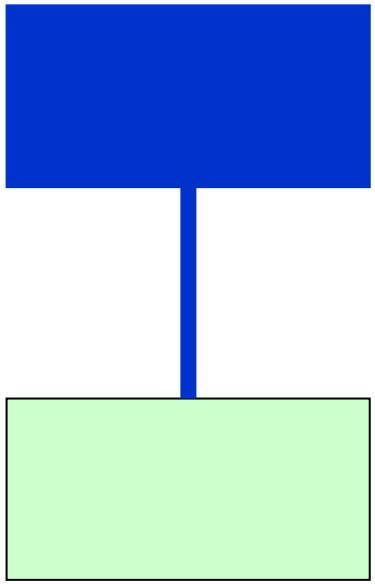
# Integration

- Given two or more source databases, build an integrated schema or database



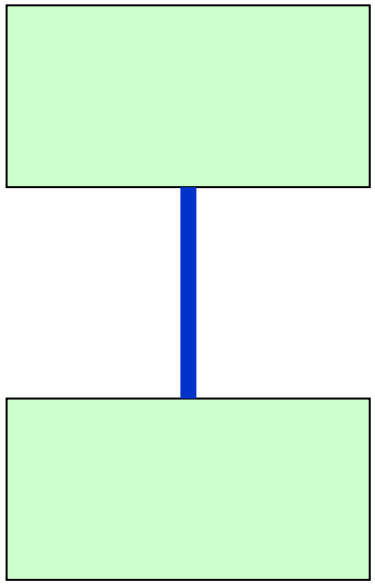
# Schema translation

- Given a schema find another one with respect to some specific goal (better quality, another model, …)



# Data exchange

- Given a source and a target database, find a transformation from the former to the latter



# A common requirement

- Schema translation and data exchange share a requirement, “correctness”:
  - The result of the exchange process should be a database with the same data as the original one
  - The result of the translation should be a schema capable of handling the databases with the same information as the original one
- We would like the target database to be equivalent to the source one
- and the target schema to be equivalent to the source one

# Outline

- Introduction and motivation
- Model management
- Schema and data translation
- **Models, schemas, mappings**
- Information capacity dominance and equivalence
- Schema translation
- Data exchange

# Schemas …

- … in heterogeneous settings
- Various approaches, on the basis of various coordinates:
  - Do we describe models or just consider models as subsumed by a rather general one?
  - Do we look for simplicity (and simplification) or for details?

# Simple representations

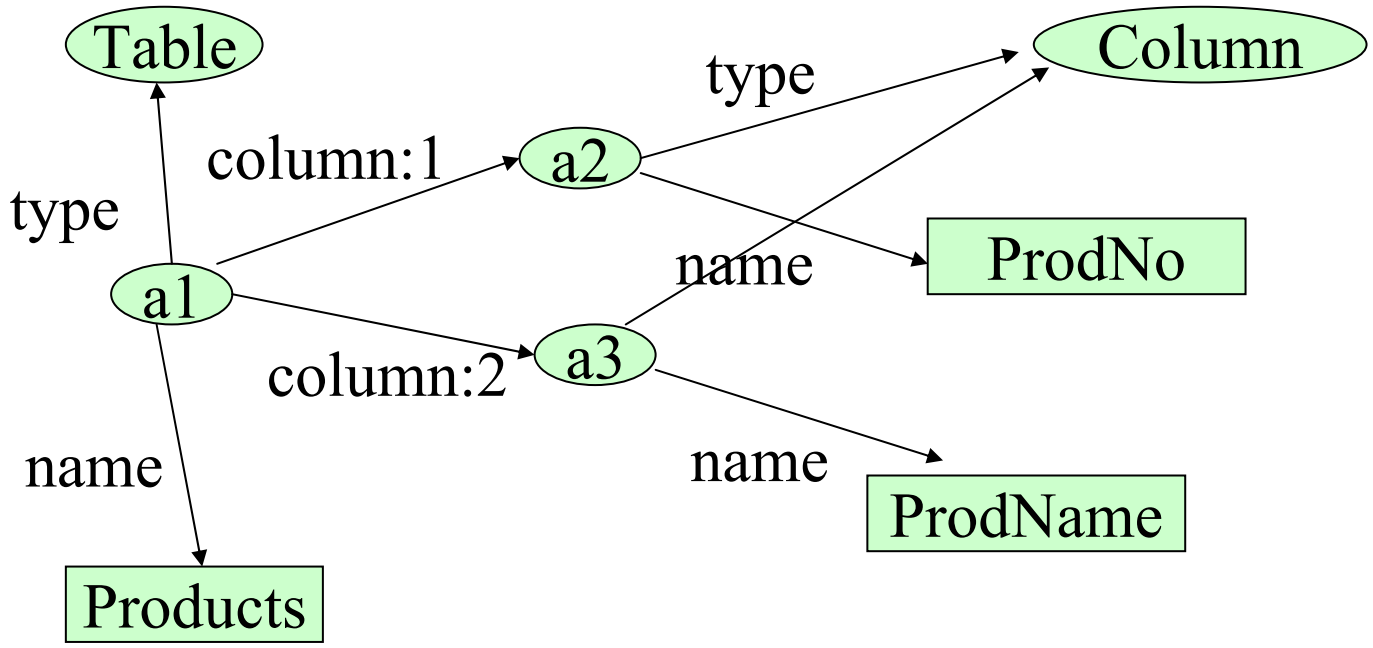
- Simple graph based models (corresponding to simple nested relation or nested object)
- There is no goal of being exhaustive, but to show the main ideas
  - "… our model is minimalist. The data structure we use consists of ordered labeled trees. We claim that this simple model is general enough to capture the essence of formats we are interested in." (Abiteboul, Cluet, Milo 1997-2002)
    Also represents instances

# A bit more complex

- Graphs, with more typing and constraints
  - Miller et al 1994: Schema Intention Graphs, SIG
    - various types of nodes
    - annotations (= cardinality constraints)
    - allow for the description of instances
  - Melnik et al 2003; includes metalevel information

# A graph representation

(Melnik et al 2003)



# Metalevels

- An idea that has existed for a while:
  - Mark and Roussopoulos, ER 1983

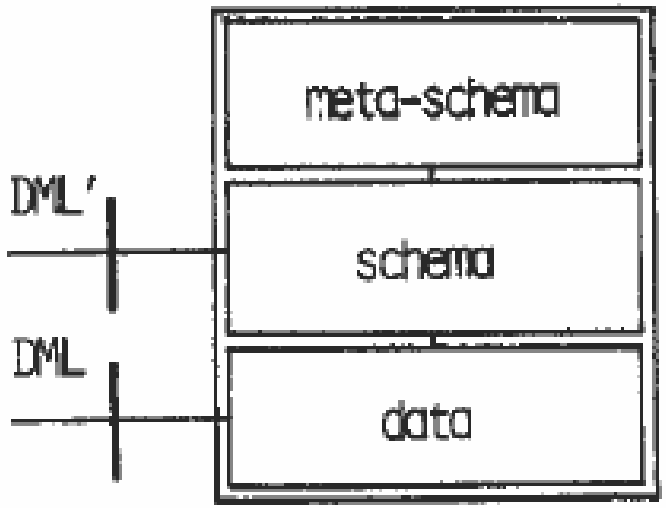



fig. 5

# Metamodels

- The goal of a metamodel is the description of models
- In terms of what?
  - A model is a set of constructs
    - What is the universe of constructs?
- We would like to be able to define “any possibile model”
  - What does this mean?

# “Any possible model”?

- Each model has its own constructs
- Each model gives the definition (the semantics) of the constructs in a different way
- Each model introduces specific features that have no counterpart in other models
- New models with new features could be introduced

# However

(Atzeni & Torlone 1996)

- The constructs in the various models are rather similar:
  - they can be classified into a small number of categories (“metaconstructs")
  - Translations can be defined on metaconstructs, and there are “standard”, accepted ways to deal with translations of metaconstructs
- That is:
  a metamodel approach

# Constructs: a classification

(Hull & King 87)

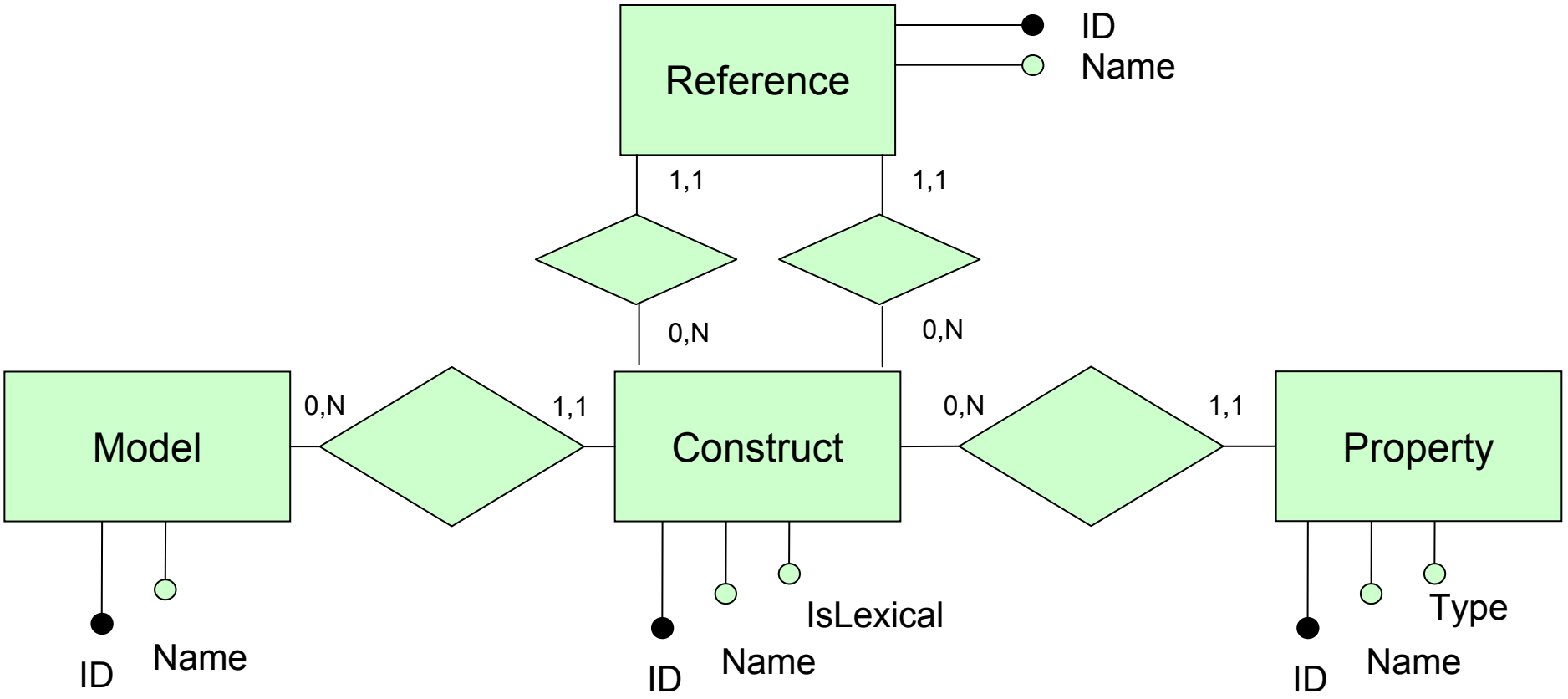
- Lexical types: sets of printable values
  - Domain
- Abstract types
  - Entity type , set of objects in the world
  - Class , set of objects in the system
- Aggregation: a construction based on (subsets of) cartesian products
  - Relationship in the E-R model
  - Relation in the relational model
- Function
  - Attribute in the E-R model
  - Function in a functional data model
- Hierarchies

# Constructs and models

- We can fix a set of metaconstructs of interest (each with a set of possible variants):
  - lexical, abstract, aggregation, function, ...
  - extended if needed, but this will not be frequent
- Then a model can be defined in terms of the metaconstructs its constructs refer to
  - E.g., the ER model:
    - Abstract (called Entity)
    - Function from Abstract to Lexical (Attribute)
    - Aggregation of abstracts (Relationship)
    - …

# A metadictionary

(Atzeni, Cappellari, Bernstein 2005)



# A metadictionary handling various models

| Model | |
|---|---|
| **OID** | **Name** |
| 1 | Relational |
| 2 | Entity-Relationship |

| Construct | | | |
|---|---|---|---|
| **OID** | **Model** | **Name** | **IsLex** |
| 1 | 1 | Rel_Table | F |
| 2 | 1 | Rel_Column | T |
| 3 | 2 | ER_Entity | F |
| 4 | 2 | ER_Attribute | T |
| 5 | 2 | ER_Relationship | F |

| Reference | | | |
|---|---|---|---|
| **OID** | **Name** | **Construct** | **Target** |
| 30 | Table | 2 | 1 |
| 31 | Entity | 4 | 3 |
| 32 | Entity1 | 5 | 3 |
| 33 | Entity2 | 5 | 3 |

nd

| Property | | | |
|---|---|---|---|
| **OID** | **Name** | **Construct** | **Type** |
| 11 | Name | 1 | String |
| 12 | Name | 2 | String |
| 13 | IsKey | 2 | Boolean |
| 14 | IsNullable | 2 | Boolean |
| 15 | Type | 2 | String |
| 16 | Name | 3 | String |
| 17 | Name | 4 | String |
| 18 | IsIdentifier | 4 | Boolean |
| 19 | IsNullable | 4 | Boolean |
| 20 | Type | 4 | String |
| 21 | IsFunct1 | 5 | Boolean |
| 22 | IsOptional1 | 5 | Boolean |
| 23 | Role1 | 5 | String |
| 24 | IsFunct2 | 5 | Boolean |
| 25 | IsOptional2 | 5 | Boolean |
| 26 | Role2 | 5 | String |

# More approaches

- Barsalou and Gangopadhyay 1992
  - A metamodel with a few predefined constructs, which can be specialized
- Bowers and Delcambre ER 2003
  - A uniform description of models and schemas, which allows the access to models, schemas and data in the same context (same query)

# Mappings

- What is a mapping between two schemas?
  - a function (e.g., an SQL query or view)
    - In what direction?
      - GAV (global as a view of local)
      - LAV (local as a view of global)
  - a predicate
  - a set of correspondences (pair of elements)
  - a function with two or more arguments
  - a third schema (a reified mapping) related to the other two via two sets of correspondences

# Outline

- Introduction and motivation
- Model management
- Schema and data translation
- Models, schemas, mappings
- **Information capacity dominance and equivalence**
- Schema translation
- Data exchange

# “Database Equivalence”

- Studied for 30 years!
- Some representative papers
  - McGee: A Contribution to the Study of Data Equivalence. IFIP Working Conference Data Base Management 1974
  - Borkin: Data Model Equivalence. VLDB 1978
  - Biller: On the equivalence of data base schemas - a semantic approach to data translation. Inf. Syst. (1979)
  - Atzeni, Ausiello, Batini, Moscarini: Inclusion and Equivalence between Relational Database Schemata. Theor. Comput. Sci. (1982)
  - Lien: On the Equivalence of Database Models. J. ACM (1982)
  - Hull: Relative Information Capacity of Simple Relational Database Schemata. SIAM J. Comput. (1986)
  - Miller, Ioannidis, Ramakrishnan: The Use of Information Capacity in Schema Integration and Translation. VLDB 1993
  - Miller, Ioannidis, Ramakrishnan: Schema equivalence in heterogeneous systems: bridging theory and practice. Inf. Syst. (1994)
  - ...

# Relative information capacity

- A notion used to compare database schemas
  - The information capacity of a schema is the ability of the schema to hold information
- The information capacity of two schemas can be
  - “the same” (information capacity equivalence)
  - comparable (information capacity dominance)
  - incomparable
- The approach is based on mappings between allowed database instances

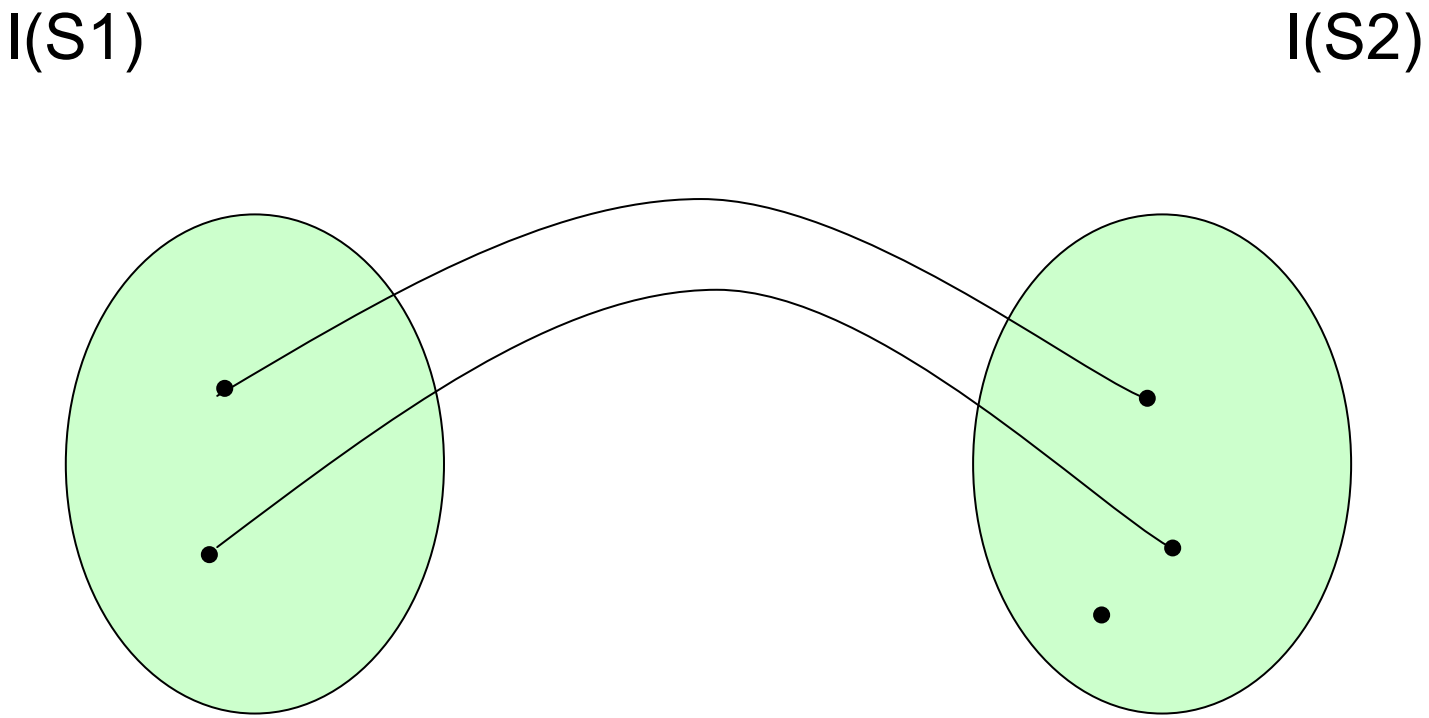
# Dominance and equivalence

- Schemas S1, S2
- S2 dominates S1
  - Whatever can be represented by S1 can be represented by S2
- S1 and S2 are equivalent
  - S1 dominates S2 and S2 dominates S1
  - Whatever can be represented by S1 can be represented by S2 and viceversa

# Information capacity dominance

- Schemas S1, S2
- The basic idea
  - Whatever can be represented by S1 can be represented by S2
- More in detail:
  - For every instance I1 of S1, there is an instance I2 of S2, with the same information

# Comparing information capacity



# “With the same information”

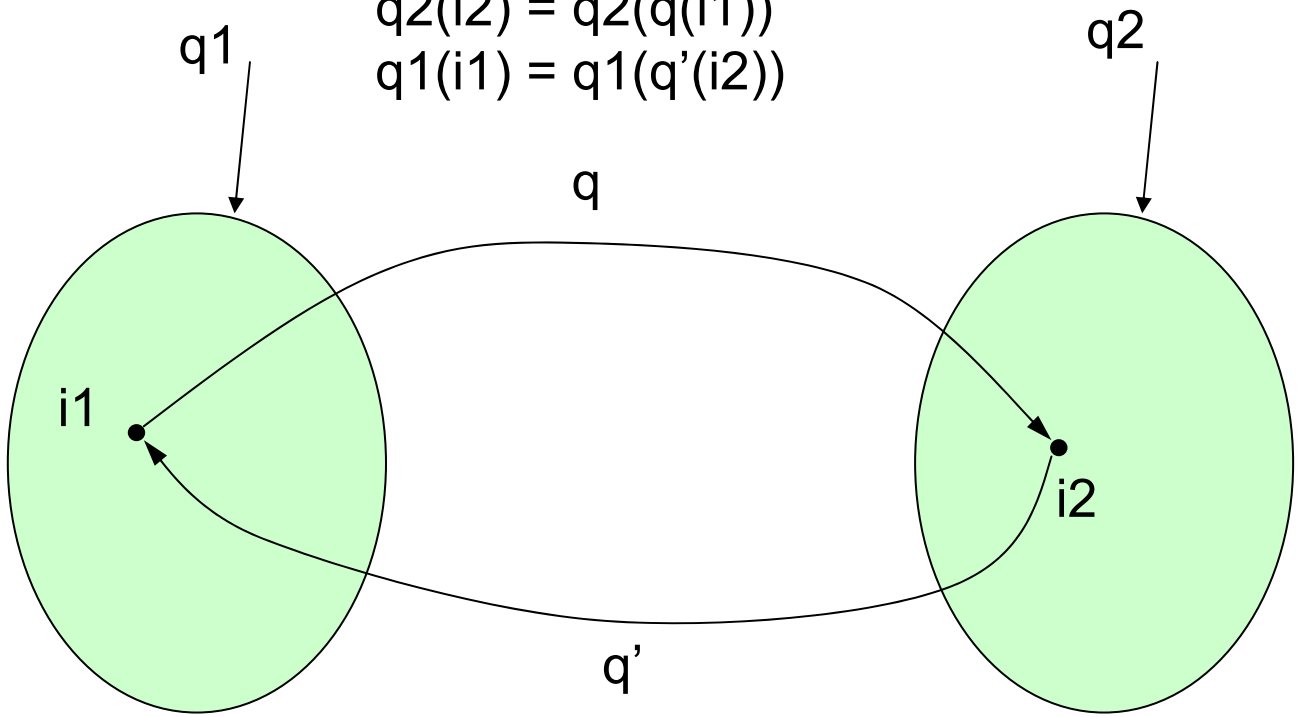
- When do two database instances i1 and i2 have the same information?
  - Represent the same facts of the real world – a “semantic” notion (Borkin 1978, Biller 1979)
    - but: how do you describe this semantics?
  - Provide, via queries, the same data
    - For every query q1 on i1 there is a query q2 on i2 that gives the same result
      - this works if there are q and q' that convert instances from a schema to the other
        - » i2 = q(i1) and i1 = q'(i2)
        - » q and q’ are inverse
    - Ok, but then it depends on the query language used to express q and q’ (Atzeni et al 1982)

# Query dominance

q2(i2) = q1(i1)

i2 = q(i1) & i1 = q'(i2)

q2(i2) = q2(q(i1))
q1(i1) = q1(q'(i2))



# More precisely

- S2 dominates S1
  - if there are functions (queries) q and q' such that, for every instance i1 of S1
    - i2 = q(i1) is a legal instance of S2
    - q'(i2) = i1
  - q and q' are the same for all instances
- S1 and S2 are equivalent
  - if
    - S1 dominates S2 and
    - S2 dominates S1

# Notions of dominance (and equivalence)

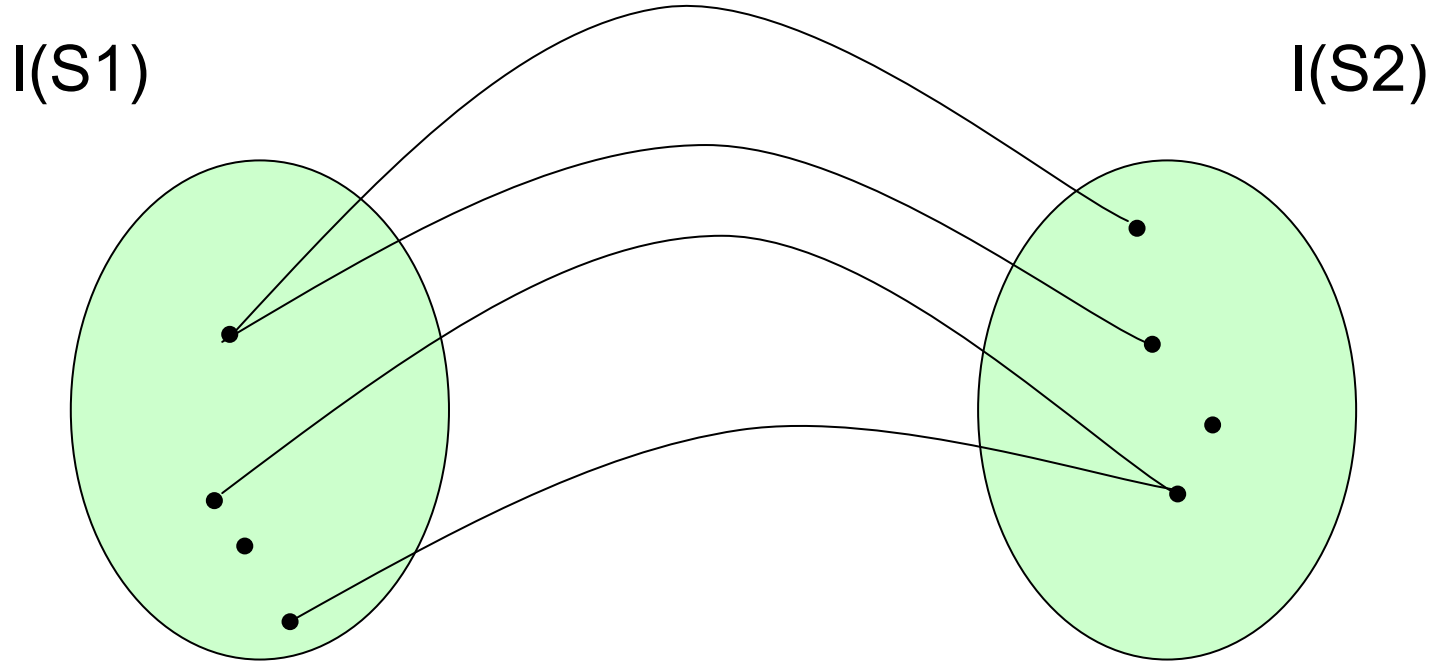
- Depending on the power of the query language (Hull 1986)
  - Absolute (any kind of mapping)
  - Internal (no new domain values)
  - Generic (based on the notion of generic queries – domain elements are treated as uninterpreted)
  - Calculous (the mappings are relational calculus queries)

# Comments

- Relative information capacity
  - Elegant work, good foundational ideas
  - Difficult to use
    - Mainly negative results
    - Semantics can be circumvented with tricks

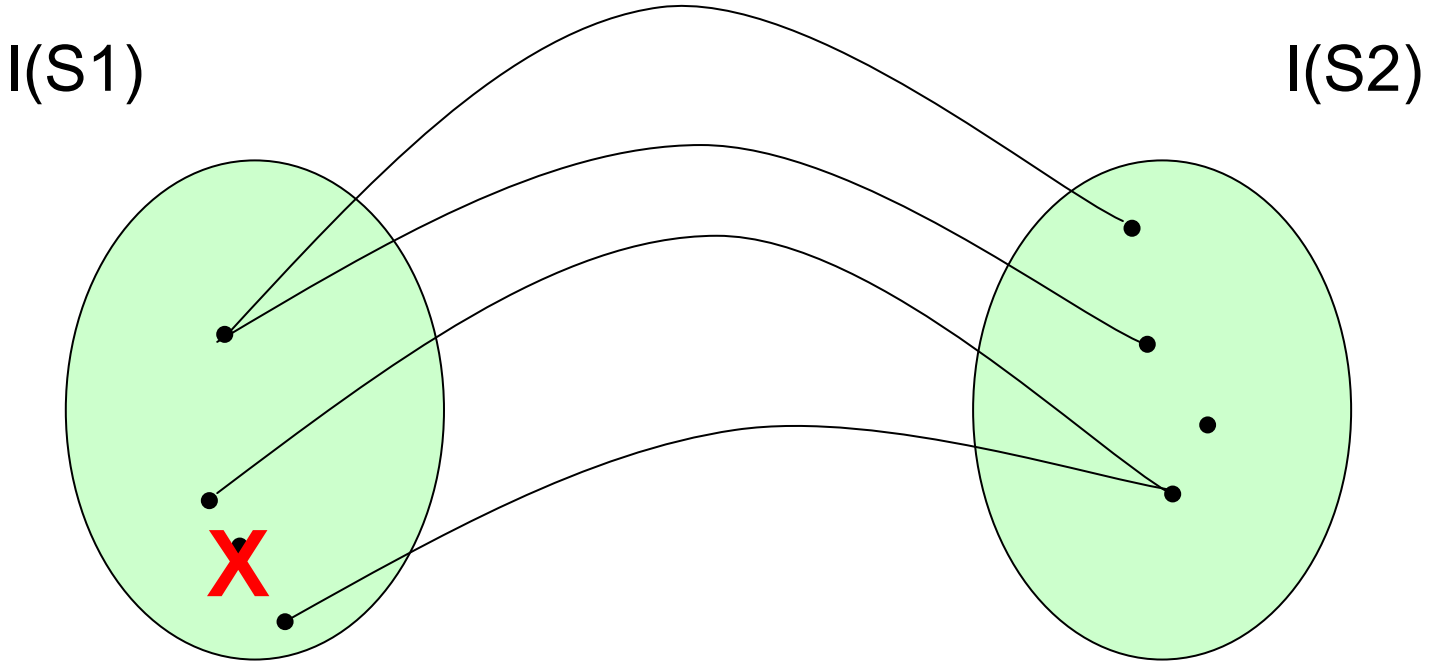
# Another point of view

- Schemas S1 and S2
- A mapping f (binary relation) from I(S1) and I(S2)



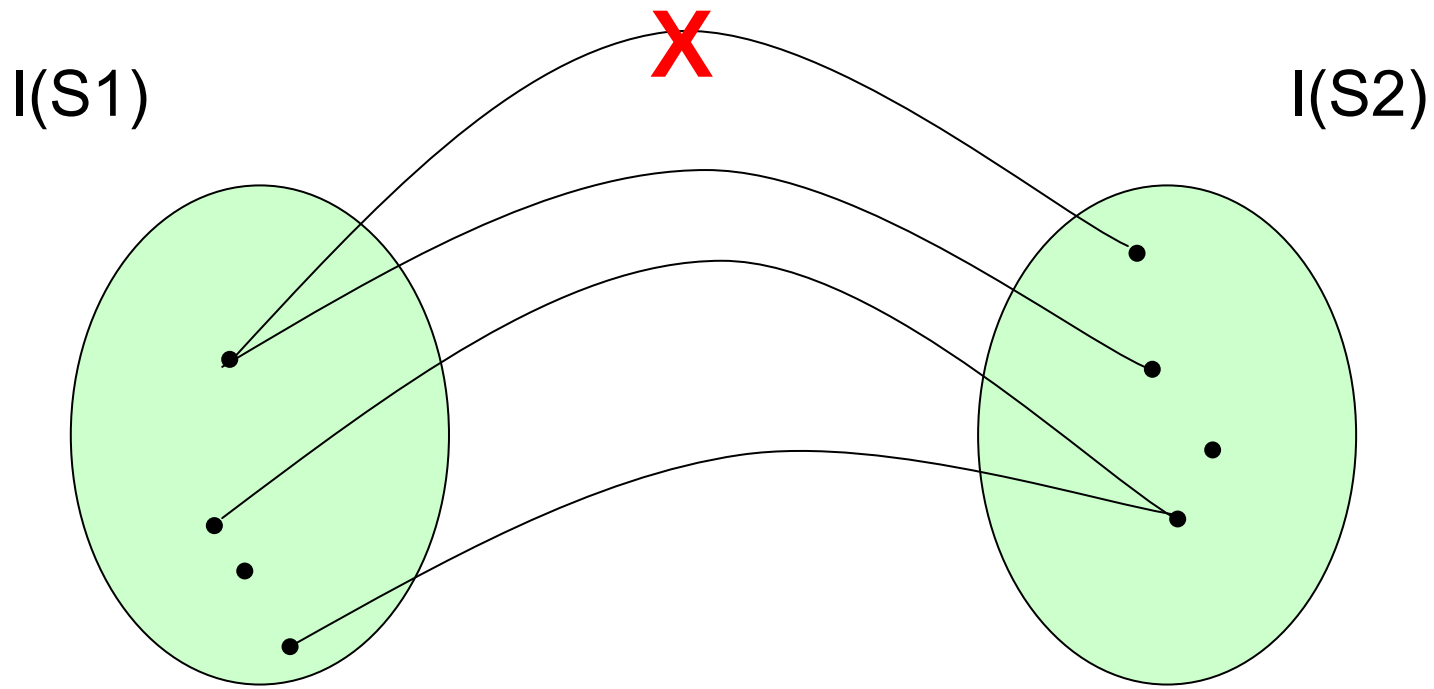
# Properties of the mapping, 1

- f is total if it is defined on every element of I(S1)



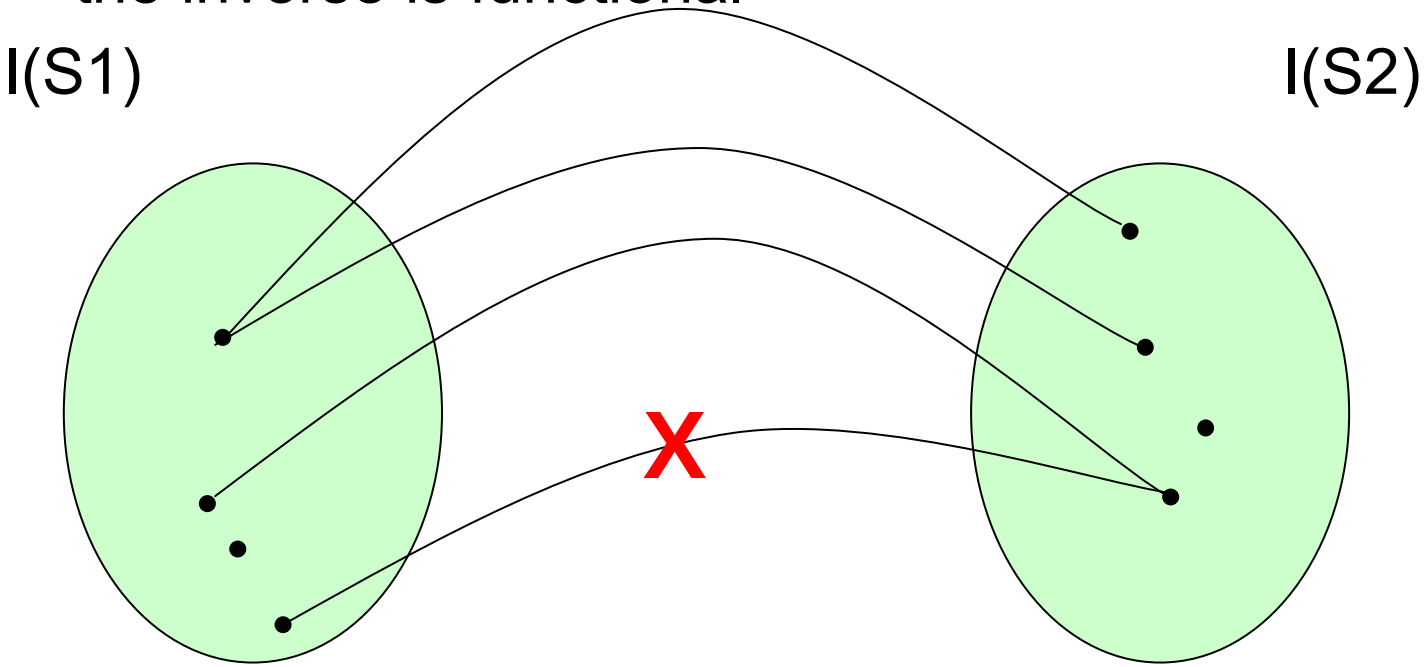
# Properties of the mapping, 2

- f is functional if for every element of I(S1) there is at most one associated element in I(S2)



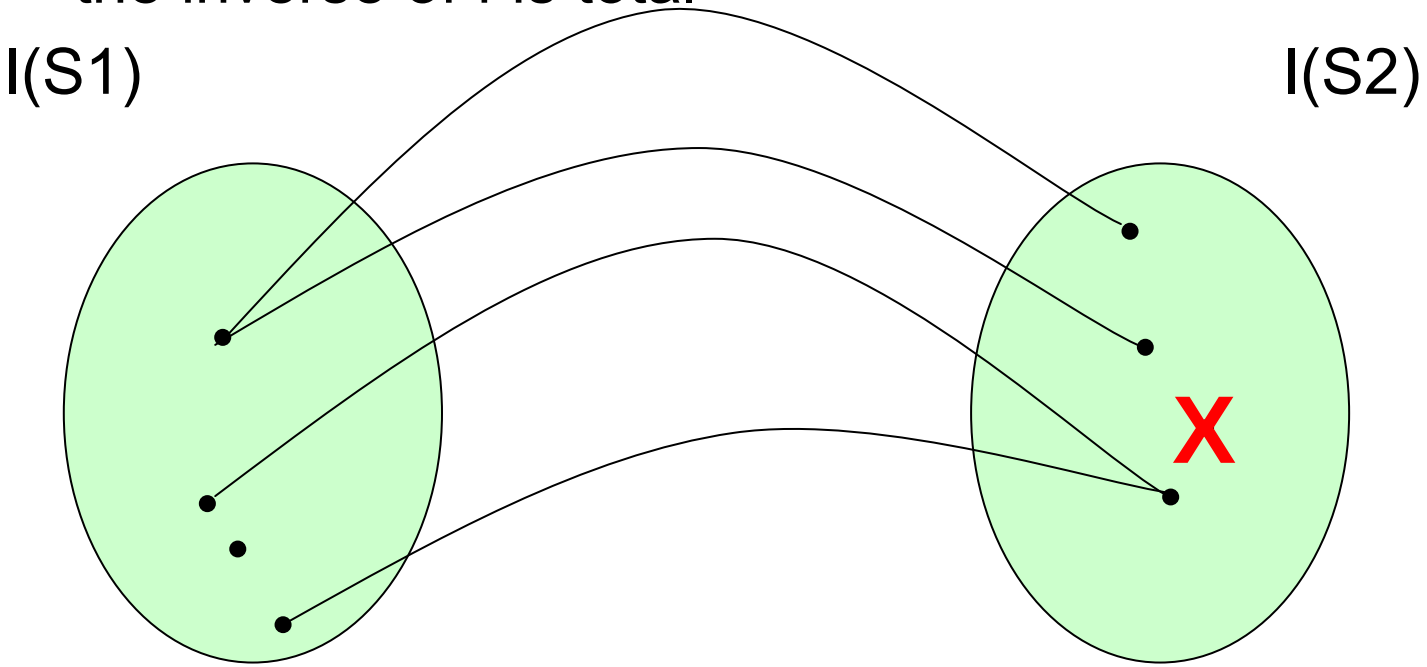
# Properties of the mapping, 3

- f is injective if no two elements of I(S1) are associated with the same element of I(S2)
  - the inverse is functional



# Properties of the mapping, 4

- f is surjective if every element of I(S2) is associated with at least one element of I(S1)
  - the inverse of f is total



# Information capacity and properties of the mapping

- If the mapping is functional, total and injective, then
  - S2 dominates S1
- Note:
  - This implies that it is a bijection between
    I(S1) and a subset of I(S2)

# Information capacity and properties of the mapping, 2

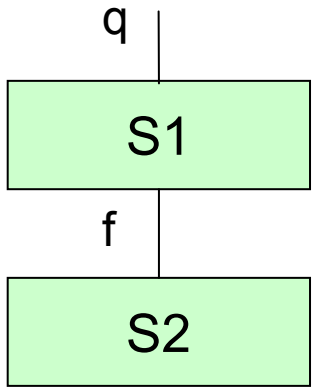
- If the mapping is functional, total, injective, and surjective then
  - S2 and S1 have equivalent information capacity
- Note:
  - This implies that it is a bijection between

    I(S1) and I(S2)

# Schema integration and translation tasks: a taxonomy of goals

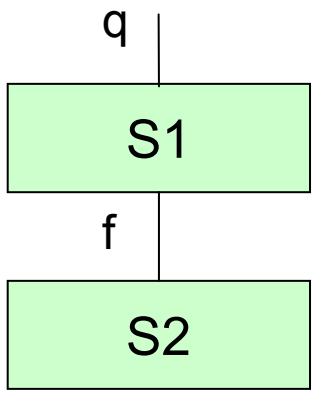
(Miller et al 1993)

- Schemas S1 and S2, with S1 used as interface for s2
- G1: querying via S1 the data handled by S2 (a view)
- G2: G1 + viewing the whole db of S2 through S1
- G3: G2 + updating the db of S2 through S1
- G4: querying S2 via S1 and S1 via S2



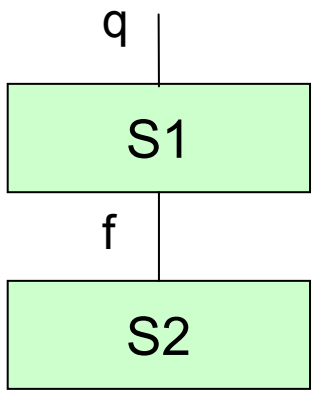
# View

- G1: querying via S1 the data handled by S2 (a view)
  - We need a total function f (the view definition) from I(S1) to I(S2)
  - q(i1) = q(f(i2) = (q ○ f) (i2)
  - No need for equivalence nor for dominance



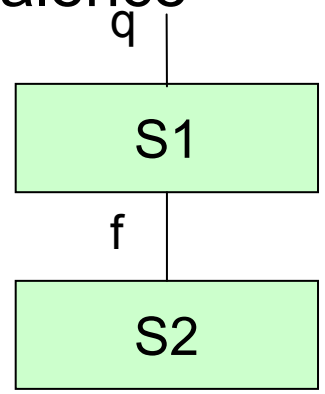
# Total view

- G2: querying via S1 the data handled by S2 (a view)
  + viewing the whole db of S2 through S1
  - No information should be lost
  - The view definition f must be a total injective function from I(S1) to I(S2)
  - dominance is required



# Total updatable view

- G4: querying via S1 … + viewing the whole db of S2 through S1 + updating the db of S2 through S1
  - Every allowed instance of S1 should correspond to an instance of S2
  - The view definition f must be an injective surjective function : a bijection
  - We need equivalence

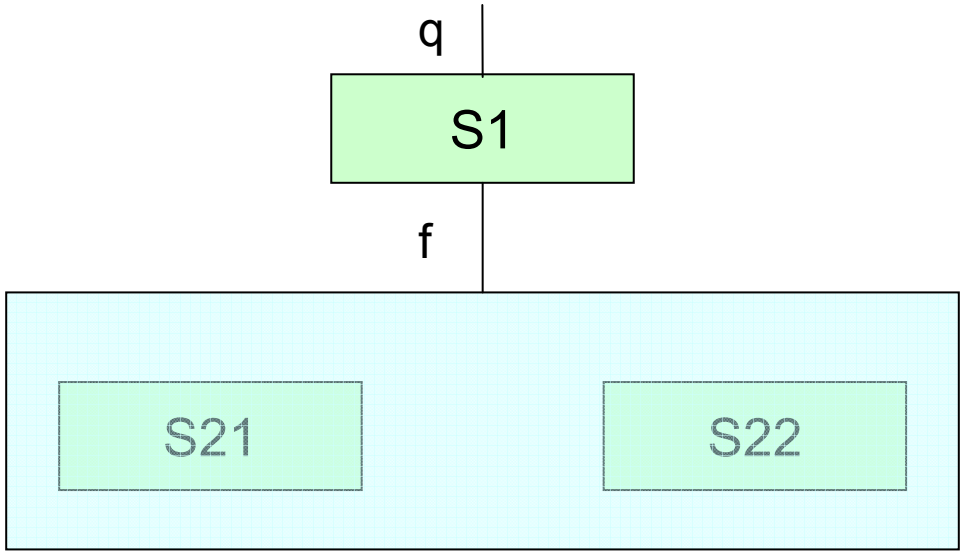


# Bidirectional view

- G4: querying S2 via S1 and S1 via S2
  - There are two mappings, but they are meaningful in practice if one is the inverse of the other
  - So, the mapping f should be total and functional, and its inverse should also be total and functional:
    - Totality of the inverse means surjectivity of f
    - Functionality means injectivity of f

    Therefore, the mapping should be bijective:
    - equivalence

# Database integration

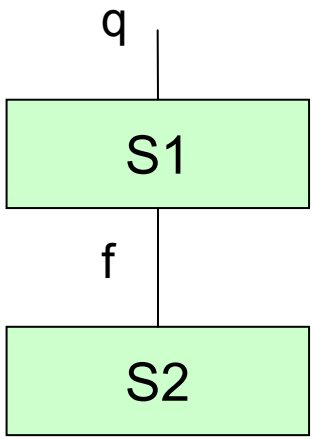



- f a mapping from the union of the local schemas to the integrated view
- Goal: total (possibly updatable) view
  - S1 dominates S2 or S1 equivalent to S2

# Schema transformation and translation

- One level up (Miller et al 1993):
  - F1 and F2 families of schemas
  - A schema transformation is a total function from F1 to F2
  - A schema transformation is a translation if F1 and F2 refer to different models
- A transformation or translation is useful if it induces a mapping between the sets of instances of the involved schemas

# Schema translation

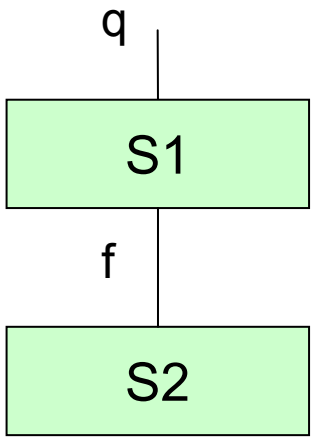



- Different models, within integration or separately
- Goal in terms of information capacity:
  - S1 should allow everything allowed by S2
    - S1 dominates S2
  - … and viceversa
    - S1 equivalent to S2

# A difficulty in schema translation

- Sometimes there is no equivalent scheme in the target model
  - Source: E-R with cardinality constraints and the target E-R model without them
- There could be two or more dominating, incomparable target schemes:
  - the source scheme is an E-R model with is-a relationships and the target an E-R model without them

# Data exchange




- Goal:
  - the schemas are given
  - What are the desirable properties of f?

# An interesting contribution

(Miller et al 1993)

- Many practical pieces of work have been analyzed within the formal framework (and by means of a unifying formalism)
- Various assumptions have been clarified
- “Conclusions”: formally justified transformations
  - offer significant benefits
  - are useful even when the transformations are not rigorously justified

# Integration and translation in practice

(Miller et al 94)

- General characterization of dominance and equivalence is difficult (often undecidable)
- Integration and design process requires many “local”, standard steps
- If the transformations are restricted to being composition of the local ones, it would suffice to study the formal properties of these local steps
- With the use of information capacity dominance and equivalence, some wrong choices can be avoided

# Outline

- Introduction and motivation
- Model management
- Schema and data translation
- Models, schemas, mappings
- Information capacity dominance and equivalence
- **Schema translation**
- Data exchange

# Schema translation

(Atzeni et al 1991- ongoing)

- Goal
  - a model-independent data dictionary, a component of an integrated (flexible, open) CASE tool
  - support to the translation of schemas and data from a model to another
- Motivation
  - many data models exist

# An ideal goal

(Atzeni and Torlone, 1990s)

- An environment that:
  - allows the definition of any possible model
  - given two models M1 and M2, and a scheme S1 of M1 (the **source** scheme and model),
    - generates a scheme S2 of M2 (the **target** scheme and model), corresponding (**equivalent**) to S1
    - and, for each database D1 over S1, generates an **equivalent** database D2 over S2

# ModelGen, as we saw

- <S2, map12> = ModelGen (S1)
  - given
    - a schema (in a model)
  - returns
    - a schema (in a different data model) and a mapping between the two
- A “translation” from a model to another
- We should call it “SchemaGen” …

# So we anticipated the needs …

- Our original proposal (Atzeni and Torlone 1996, 1997) satisfies most of the requirements for ModelGen, except for
  - The generation of a mapping between the source and the target schema
  - The possibility of generating flexible and modifiable transformations, to be integrated in complex processes

# Is the goal realistic?

- What does “any possible model" mean?
  - We have discussed the idea of a metamodel, which is effective (not universal, but extendible)
- What does “corresponding" (or “equivalent") mean?
  - We have discussed information capacity, with its difficulties

# The Supermodel

- A model that includes all the constructs (in their most general forms)
  - aggregations of lexicals (e.g., tables)
  - components of aggregations of lexicals (e.g., columns)
  - abstracts (e.g., entities or classes)
  - attributes of abstracts
  - binary aggregations of abstracts
- Since the supermodel is a model, we can describe it in a meta-dictionary (with one model only)

# Supermodel: benefits

- each scheme for any model is also a scheme for the supermodel, up to renaming
- translations can be performed within the supermodel
- each translation from the supermodel SM to a target model is also a (possibly redundant) translation from any other model to M:
  - Given n models, we need n translations, not $n^2$

# The metadictionary for the supermodel

| Construct | | |
|---|---|---|
| **OID** | **Name** | **IsLex** |
| 1 | AggregationOfLexicals | F |
| 2 | ComponentOfAggrOfLex | T |
| 3 | Abstract | F |
| 4 | AttributeOfAttribute | T |
| 5 | BinaryAggregationOfAbstracts | F |

| Reference | | | |
|---|---|---|---|
| **OID** | **Name** | **Construct** | **Target** |
| 30 | Aggreg. | 2 | 1 |
| 31 | Abstract | 4 | 3 |
| 32 | Abstract1 | 5 | 3 |
| 33 | Abstract2 | 5 | 3 |

| Property | | | |
|---|---|---|---|
| **OID** | **Name** | **Construct** | **Type** |
| 11 | Name | 1 | String |
| 12 | Name | 2 | String |
| 13 | IsKey | 2 | Boolean |
| 14 | IsNullable | 2 | Boolean |
| 15 | Type | 2 | String |
| 16 | Name | 3 | String |
| 17 | Name | 4 | String |
| 18 | IsIdentifier | 4 | Boolean |
| 19 | IsNullable | 4 | Boolean |
| 20 | Type | 4 | String |
| 21 | IsFunct1 | 5 | Boolean |
| 22 | IsOptional1 | 5 | Boolean |
| 23 | Role1 | 5 | String |
| 24 | IsFunct2 | 5 | Boolean |
| 25 | IsOptional2 | 5 | Boolean |
| 26 | Role2 | 5 | String |

# A dictionary for ER schemas and a dictionary for OO schemas

| ER_Entity | | |
|---|---|---|
| **OID** | **Schema** | **Name** |
| 301 | 1 | Employees |
| 302 | 1 | Departments |

| ER_Attribute | | | | | | |
|---|---|---|---|---|---|---|
| **OID** | **Schema** | **Name** | **isIdent** | **isNullable** | **Type** | **EntityOID** |
| 401 | 1 | EmpNo | T | F | Int | 301 |
| 402 | 1 | Name | F | F | Text | 301 |
| 404 | 1 | Name | T | F | Char | 302 |
| 405 | 1 | Address | F | F | Text | 302 |

| OO_Class | | |
|---|---|---|
| **OID** | **Schema** | **Name** |
| 201 | 3 | Clerks |
| 202 | 3 | Offices |

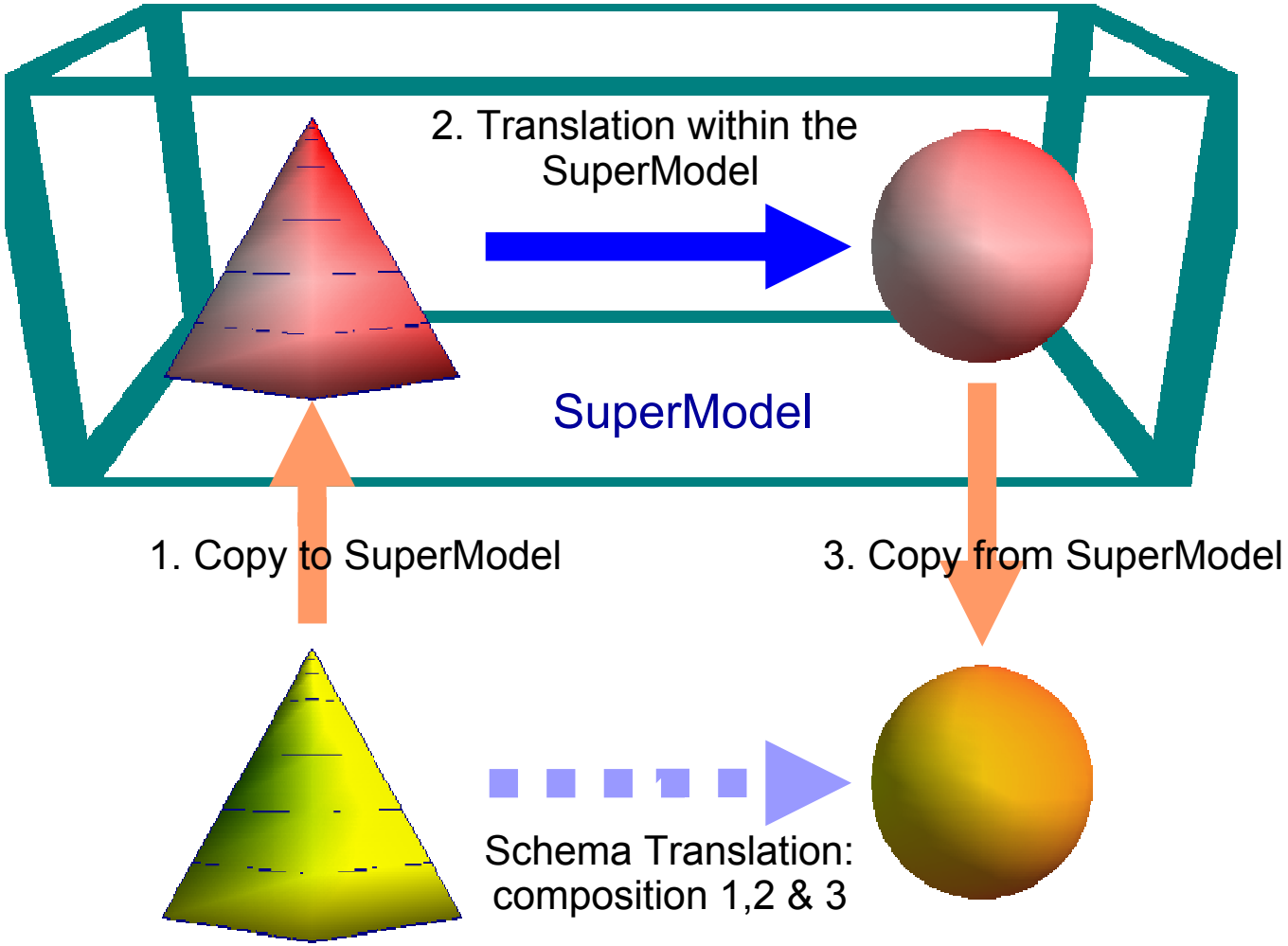
| OO_Property | | | | | | |
|---|---|---|---|---|---|---|
| **OID** | **Schema** | **Name** | **isIdent** | **isNullable** | **Type** | **ClassOID** |
| 501 | 3 | Code | T | F | Int | 201 |
| … | … | … | … | … | … | … |

# A model independent dictionary

| Abstract | | |
|---|---|---|
| **OID** | **Schema** | **Name** |
| 301 | 1 | Employees |
| 302 | 1 | Departments |
| 201 | 3 | Clerks |
| 202 | 3 | Offices |

| AttributeOfAbstract | | | | | | |
|---|---|---|---|---|---|---|
| **OID** | **Schema** | **Name** | **isIdent** | **isNullable** | **Type** | **AbstrOID** |
| 401 | 1 | EmpNo | T | F | Int | 301 |
| 402 | 1 | Name | F | F | Text | 301 |
| 404 | 1 | Name | T | F | Char | 302 |
| 405 | 1 | Address | F | F | Text | 302 |
| 501 | 3 | Code | T | F | Int | 201 |
| … | … | … | … | … | … | … |

# The translation process



# Translations

- The constructs corresponding to the same metaconstruct (e.g. entity in the E-R model and class in an object model both corresponding to Abstract) have the same "meaning"
- Translations can refer to metaconstructs, rather than to constructs (which are model specific)
  - the standard translation of entities to relations can be used whenever we need to transform Abstracts into Aggregations of lexicals

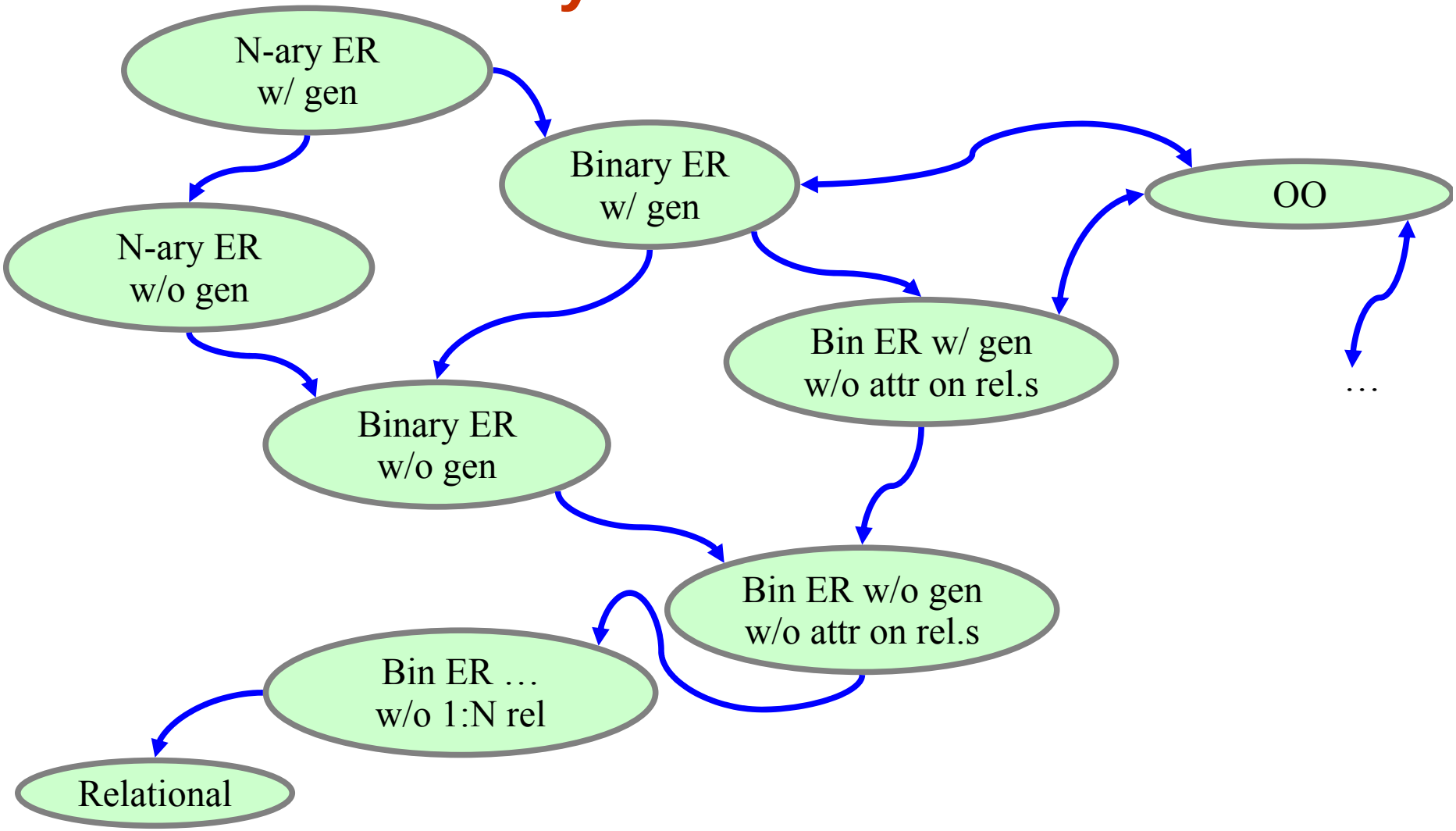
# Elementary steps

- A set of predefined basic translations, e.g.
  - eliminate n-ary aggregations; replace them with binary ones (and abstracts)
  - eliminate binary aggregations; replace them with functions
  - eliminate functions to abstracts; replace them with aggregations
  - eliminate complex attributes; replace them with simple attributes and abstracts
- Assumed to be correct and so complex translations built over them are correct by definition (an “axiomatic” approach)

# A complex translation

- From an N-ary ER model with generalizations to a simple Object model with only single valued references and no generalizations
  - Eliminate N-ary relationships (replaced by binary ones and new entities)
  - Eliminate attributes from relationships
  - Eliminate many-to-many relationships
  - Transform relationships to references
  - Eliminate generalizations

# Complex translations from a library of basic translations



# Management of translations

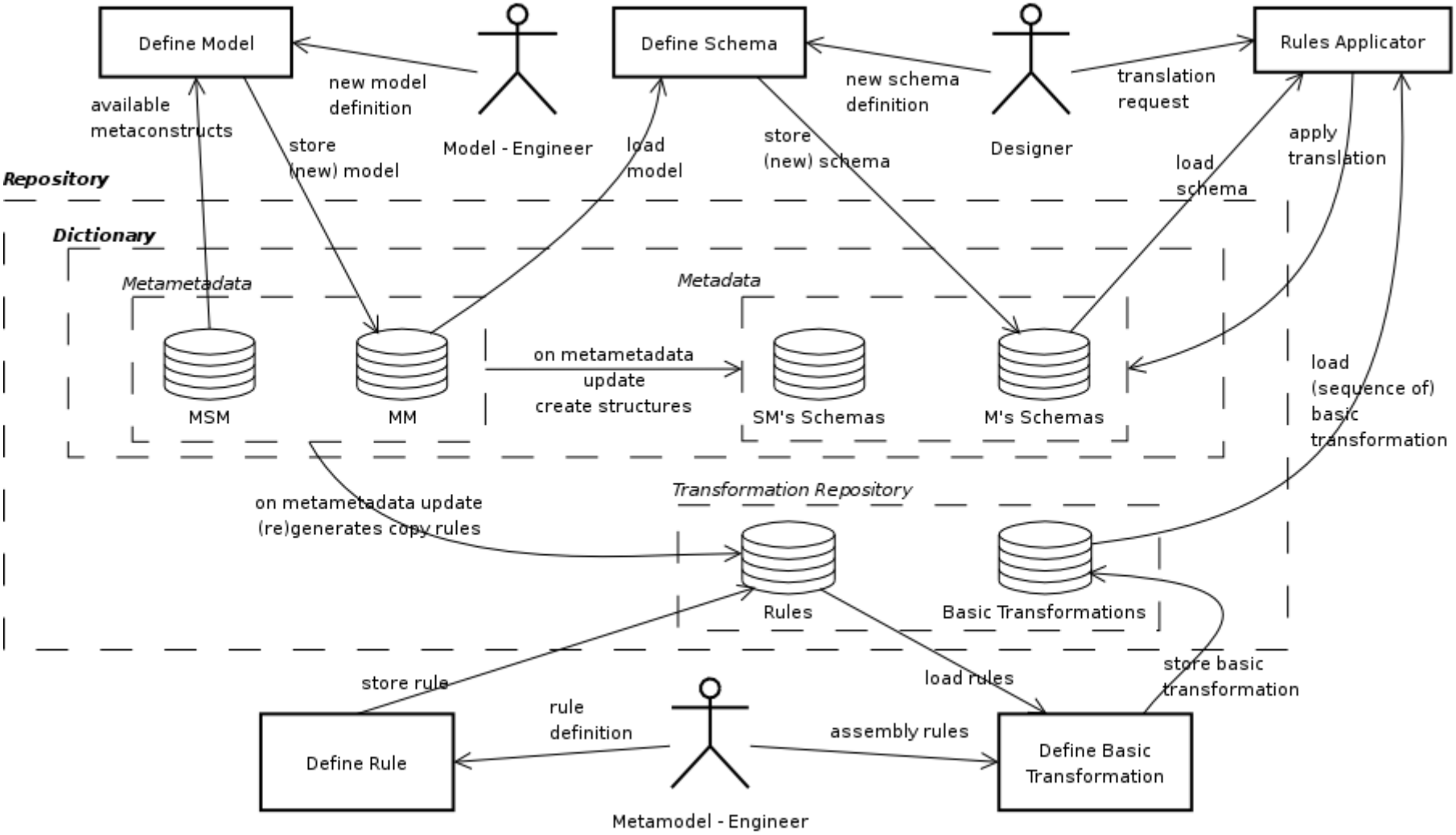
- Basic properties:
  - Correctness, minimality, …
- Construction of complex translations by picking basic translations in the library

# The current effort

(Atzeni, Cappellari and Bernstein)

- a “white box” approach
  - it exposes both the dictionary and the translations,
  - thus allowing for rapid development (and maintenance) of models and translations,
  - and for reasoning on the correctness of translations

# ModelGen: the architecture



# Translations

- Basic translations are written in a variant of Datalog, with OID invention
  - an extension of Datalog that uses functions to generate new identifiers when needed
- Skolem functions:
  - injective functions that generate "new" values (value that do not appear anywhere else; so different Skolem functions have disjoint ranges); indeed, we use a pragmatic variation of them

# More work in this direction

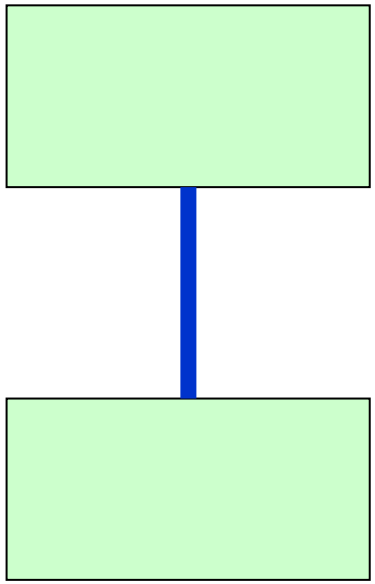
- Papotti and Torlone, 2005
- Bernstein, Melnik and Mork 2005

# Outline

- Introduction and motivation
- Model management
- Schema and data translation
- Models, schemas, mappings
- Information capacity dominance and equivalence
- Schema translation
- **Data exchange**

# Data exchange

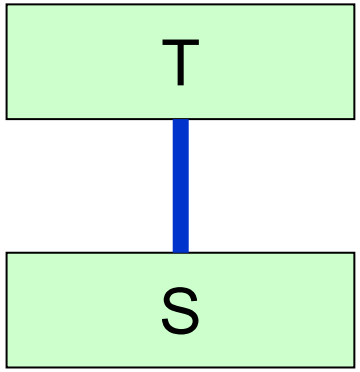
- Given a source and a target database, find a transformation from the former to the latter



# Another old problem

- An early approach:
  - Nan C. Shu, Barron C. Housel, R. W. Taylor, Sakti P. Ghosh, Vincent Y. Lum: EXPRESS: A Data EXtraction, Processing, and REStructuring System. ACM Trans. Database Syst. 2(2): 134-174 (1977)

# In Model Management terms




- Given the schemas $S_T$ and $S_S$

  schema-map = Match($S_T$, $S_S$)
- Then

  executable-map = XXXX (schema-map, $S_T$, $S_S$)
  - where

    executable-map: $I(S_T) \rightarrow I(S_S)$

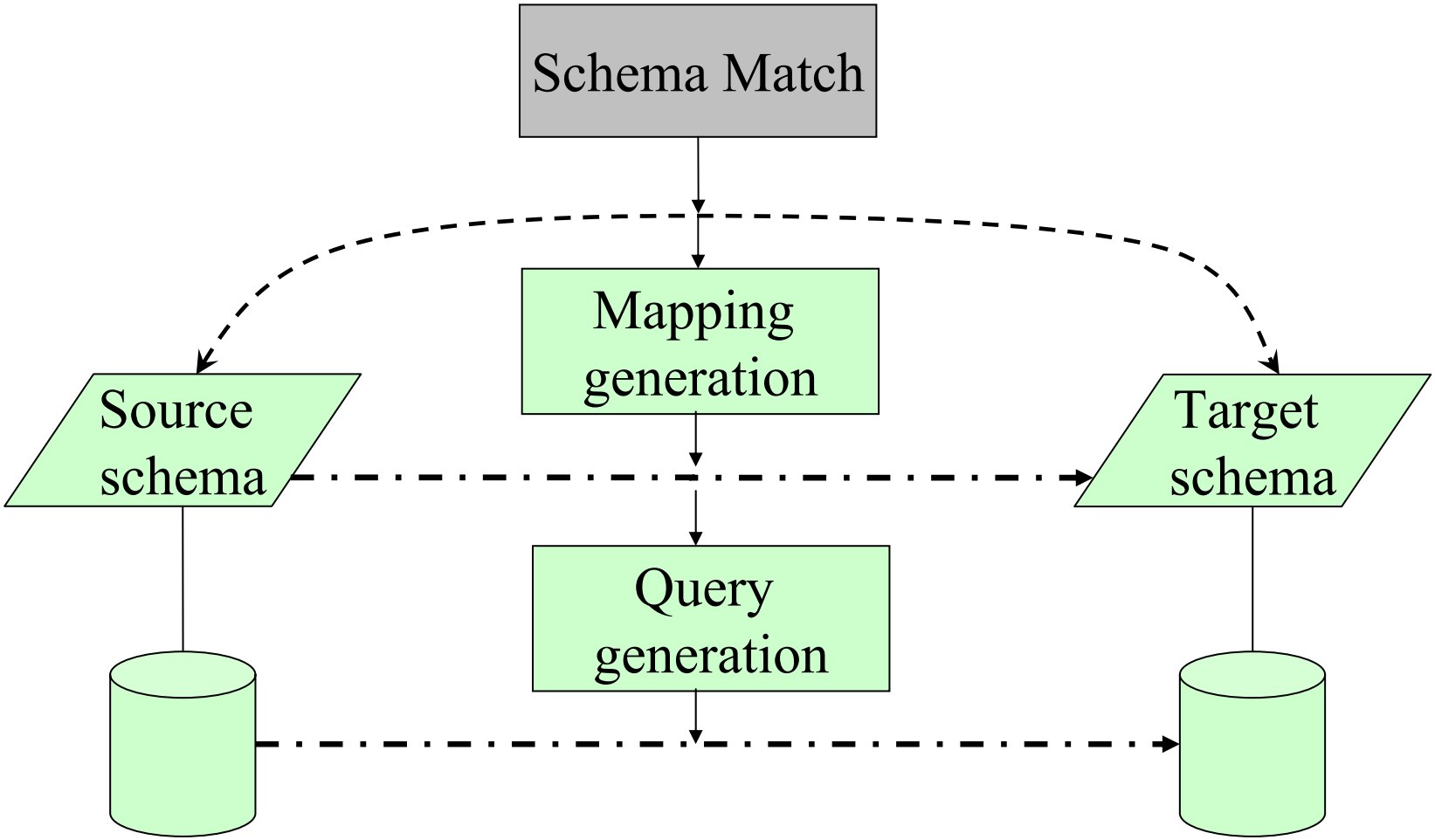
# The Clio approach

A research effort turning into a product

- Initially for relational schemas (Miller+2000)
- Evolved version for nested schemas (Popa+ 2002)
- Theoretical development for data exchange (Fagin+ 2003)
- Towards a product (Haas+ 2005)

- A major goal:
  - To use a DBMS not only as a data provider, but also as a manager of the transformation

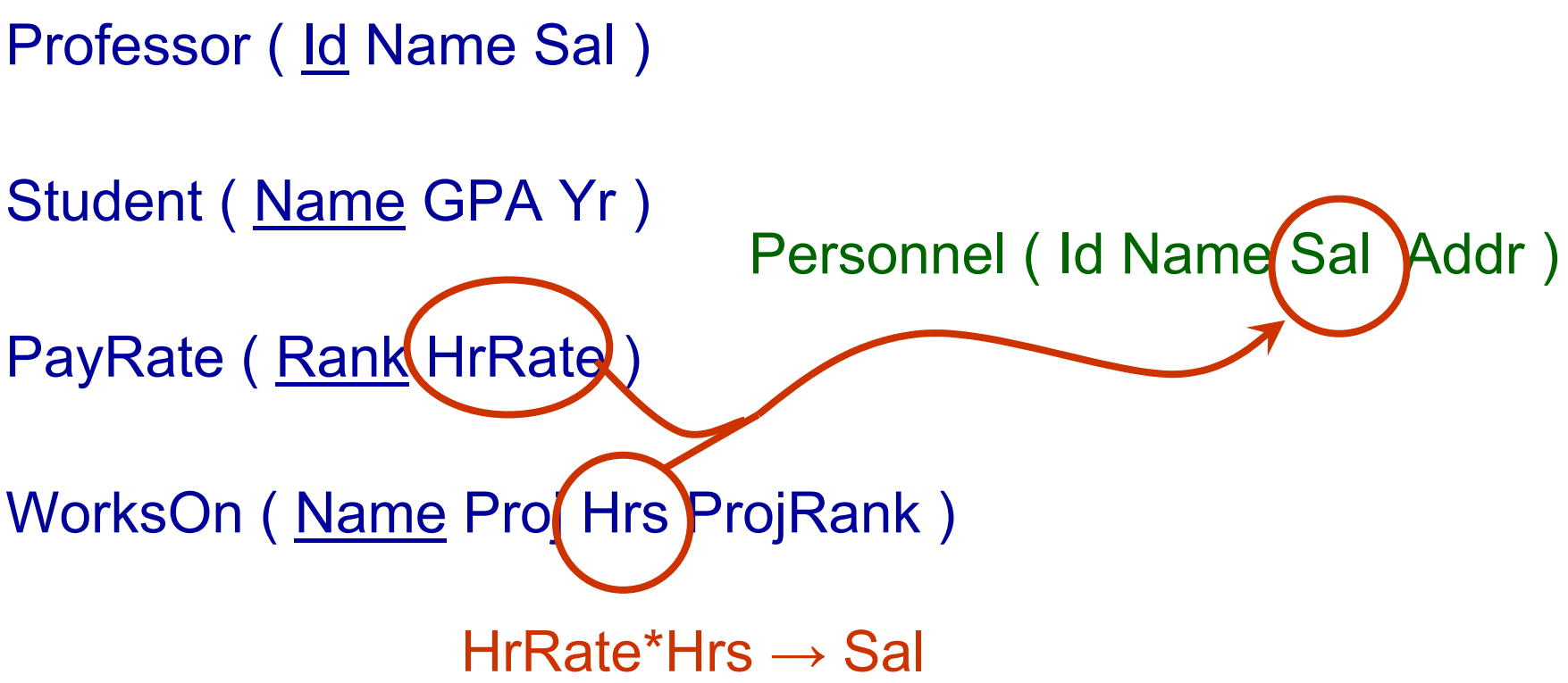
# Clio architecture



# The Clio approach

1. "Value" correspondences
   - entered by a user or
   - obtained as the result of a matching step and then refined by the user
2. Logical mappings
   - Inferred on the basis of
     - constraints (foreign keys and more)
     - structure (table or nested structure)
3. Executable queries
   - Implementation of the logical mapping

# Value correspondences




- To which values do we apply the function?
- Values in the same tuple? If so, which join to build tuples with HrRate and Hr?

# Constraints

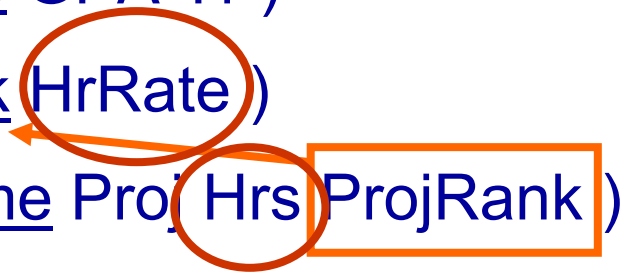
- Foreign keys are used to infer joins

Professor ( Id Name Sal )

Student ( Name GPA Yr )

PayRate ( Rank HrRate )

WorksOn ( Name Proj Hrs ProjRank )

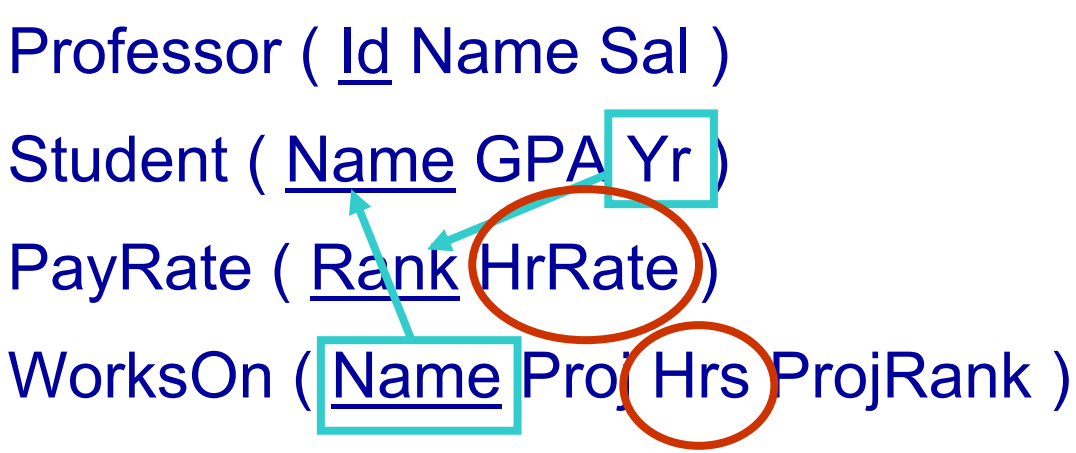


HrRate*Hrs → Sal

```
SELECT p.HrRate * W.Hrs
FROM PayRate P, WorksOn W
WHERE W.ProjRank = P.Rank
```

# Constraints

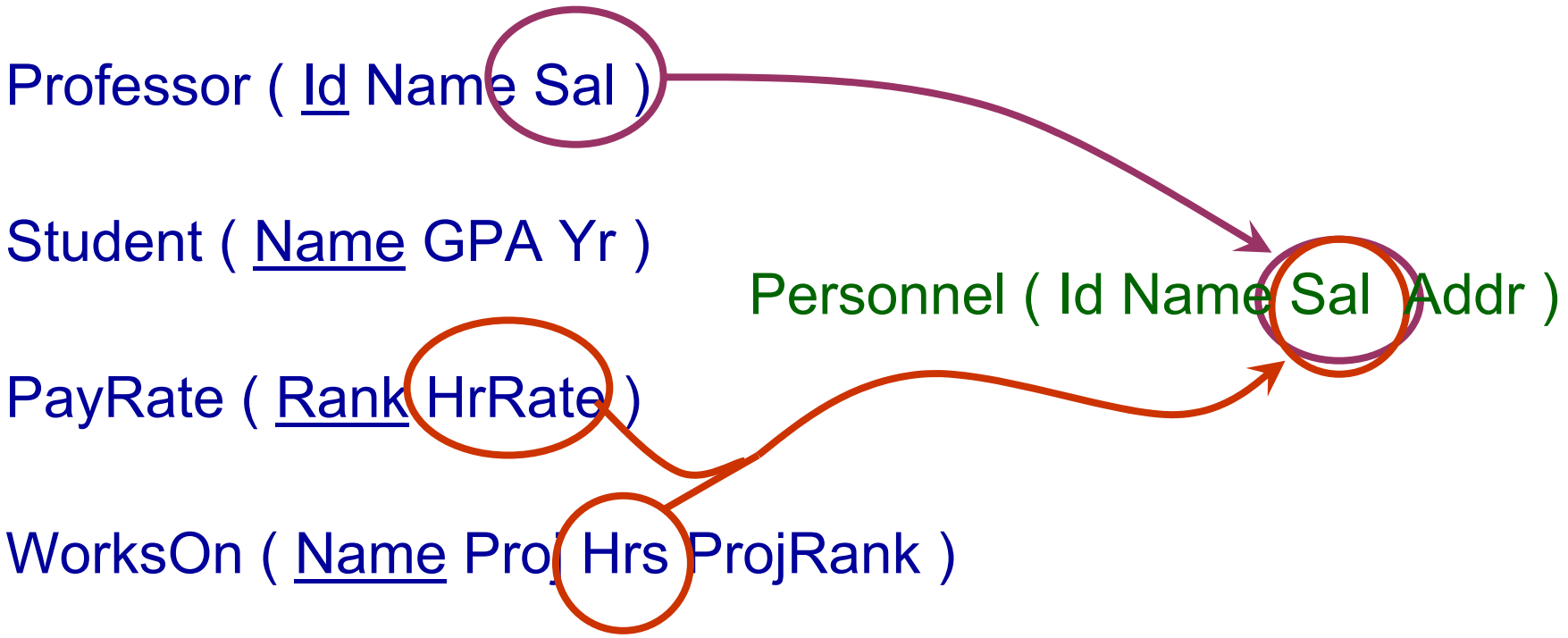
- Foreign keys are used to infer joins




HrRate*Hrs → Sal

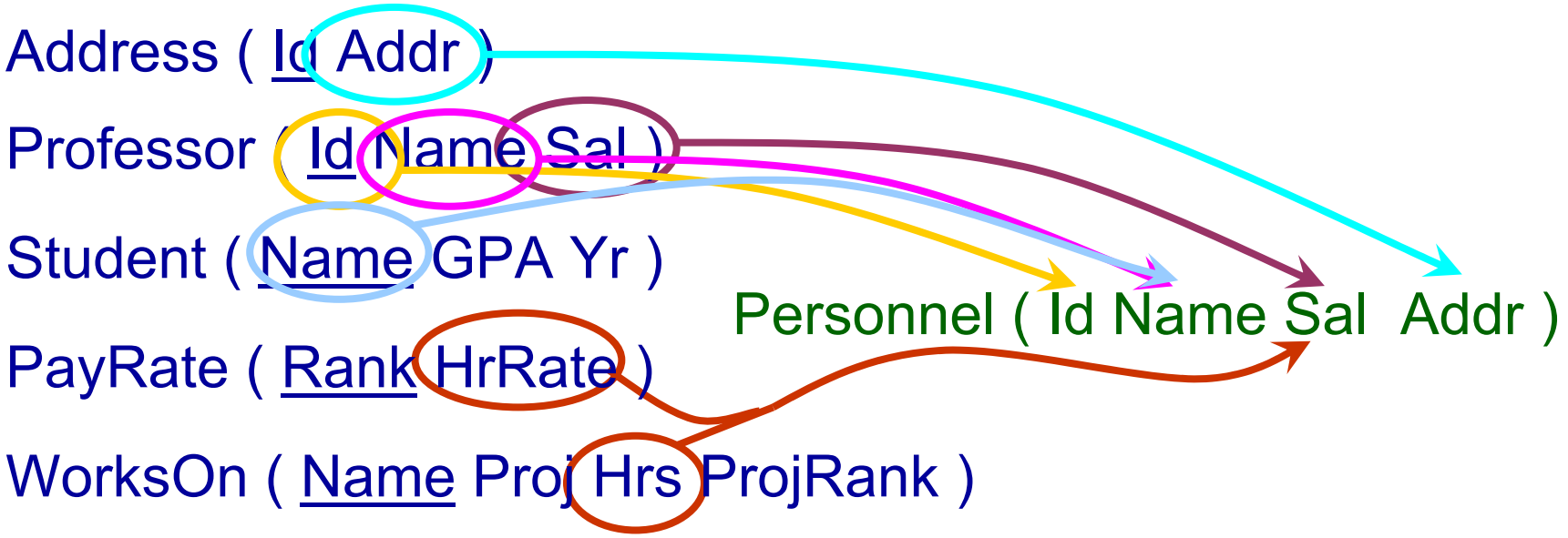
```
SELECT p.HrRate * W.Hrs
FROM PayRate P, Student S, WorksOn W
WHERE W.Name = S.Name AND S.Yr = P.Rank
```

# Value correspondences, 2



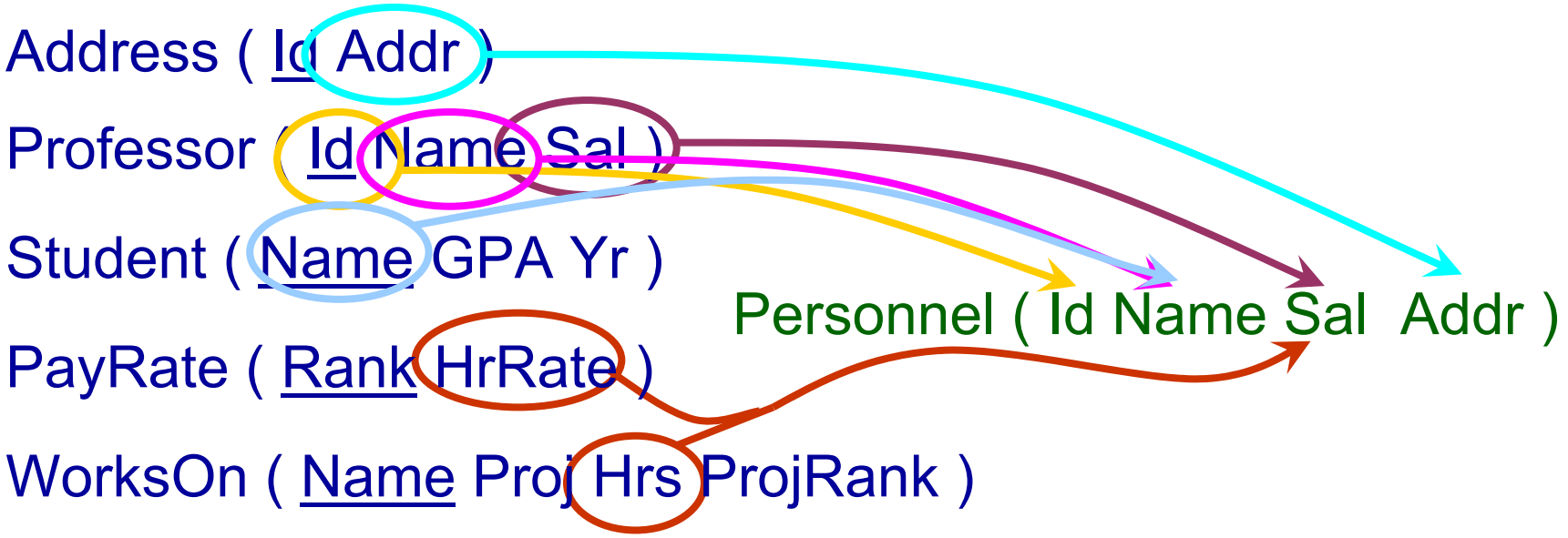

- Personnel.Sal is obtained via a union or a join?

# The process, example




- We could infer:
  - A join of Professor and Address (because of a foreign key, or of a query in the source, …)
  - A join of Student, WorksOn and PayRate (just seen)

# The process, example




```
SELECT P.Id, P.Name, P.Sal, A.Addr
FROM Professor P, Address A
WHERE A.Id = P.Id
UNION ALL
SELECT NULL AS Id, S.Name, p.HrRate * W.Hrs, NULL AS Addr
FROM PayRate P, Student S, WorksOn W
WHERE W.Name = S.Name AND S.Yr = P.Rank
```

# The algorithm

- There can be many meaningful mappings, with a combinatorial explosion
- A systematic search, with preference given to the mappings that preserve information capacity dominance or equivalence (Hull 86, MIller 94, …)
- Two groups of mappings:
  - "vertical compositions," via joins (mainly 1:N, over FK); outer joins, unless contraints …
  - "horizontal compositions," via unions (possibly after restructuring)

# Conclusion

- The “Asilomar report”:
  *A ten-year goal for database research*
  - *The information utility:*
    *make it easy for everyone to store, organize, access, and analyze the majority of human information online*

- A lot of interesting work has been done but …
- …integration, translation, exchange are still difficult…