Bx Tutorial, Database Flavor: Updatable or Invertible Mappings

James F. Terwilliger

Microsoft Research

Inside the Dark, Miserable Mind of the Database Researcher

org.microsoft.research.james

Corporate Overlord

1


LOST
CONFUSED
UNSURE
UNCLEAR
PERPLEXED
DISORIENTED
BEWILDERED

2

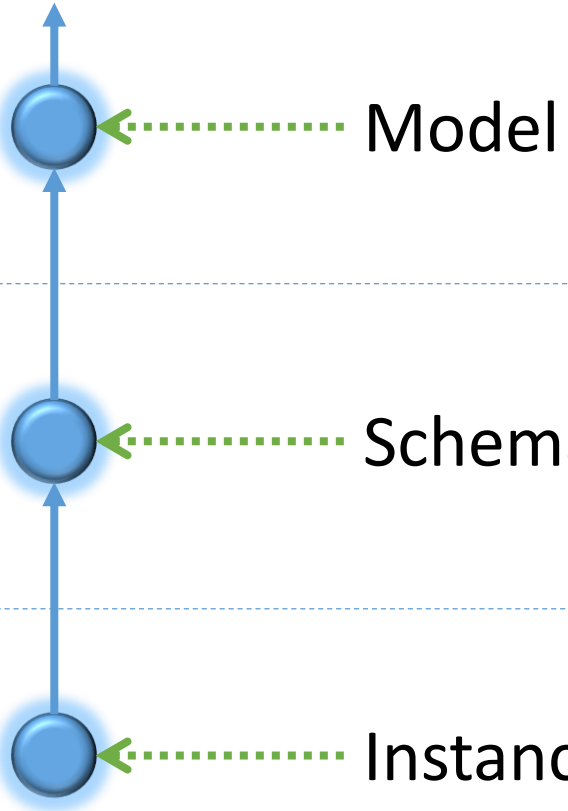
3

1

Words, words, words…

The Meta-Muddle

This way be dragons.



M2 Model

M1 Schema

M0 Instance

OMG!!!1!

Object-Oriented

Relational XML (grr…)

Entity-Relationship

Class

Table intent XML Schema

ER diagram

Object

Table extent XML document

“Database”

“Query”

Flowery prose for “function”

Q: $S_1 \rightarrow S_2$

Two schemas, almost always from the same model

What do they all have in common?

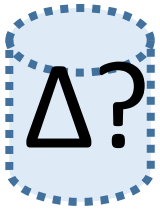
Relational algebra	$\pi_c \sigma_{a=2}(T \bowtie_{b=d} U)$
Relational calculus	{<c>| ∃a,b <a,b,c>∈T ∧ ∃d,e,f <d,e,f>∈U ∧ b=d ∧ a=2}
SQL	select c from T join U on T.b=U.d where a = 2
Datalog	Answer(c) := T(2,b,c),U(b,e,f).
Source-**T**arget **T**uple-**G**enerating **D**ependencies	$\forall_c((\exists_{b,e,f} T(2,b,c) \wedge U(b,e,f)) \rightarrow Answer(c))$
XQuery (grr…)	for $d in doc(“data.xml”)/data for $t in $d/t for $u in $d/u where $t/a=2 and $t/b=$t/g return $t/c

Declarative versus Operational

1. State intent
2. ????
3. Profit!

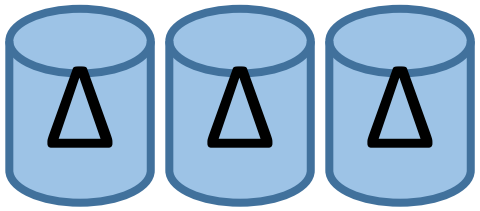
- Can the query be answered?
- Does the query have a unique answer?
- What is the fastest way to run a query?
- Can the query be inverted in some fashion? (Usually unspecified or operational)

“Logical Data Independence”



Conceptual Model

Logical Model



Physical Model

Views, external schemas, client programs

Logical Data Independence

This is where we keep trying to apply it again

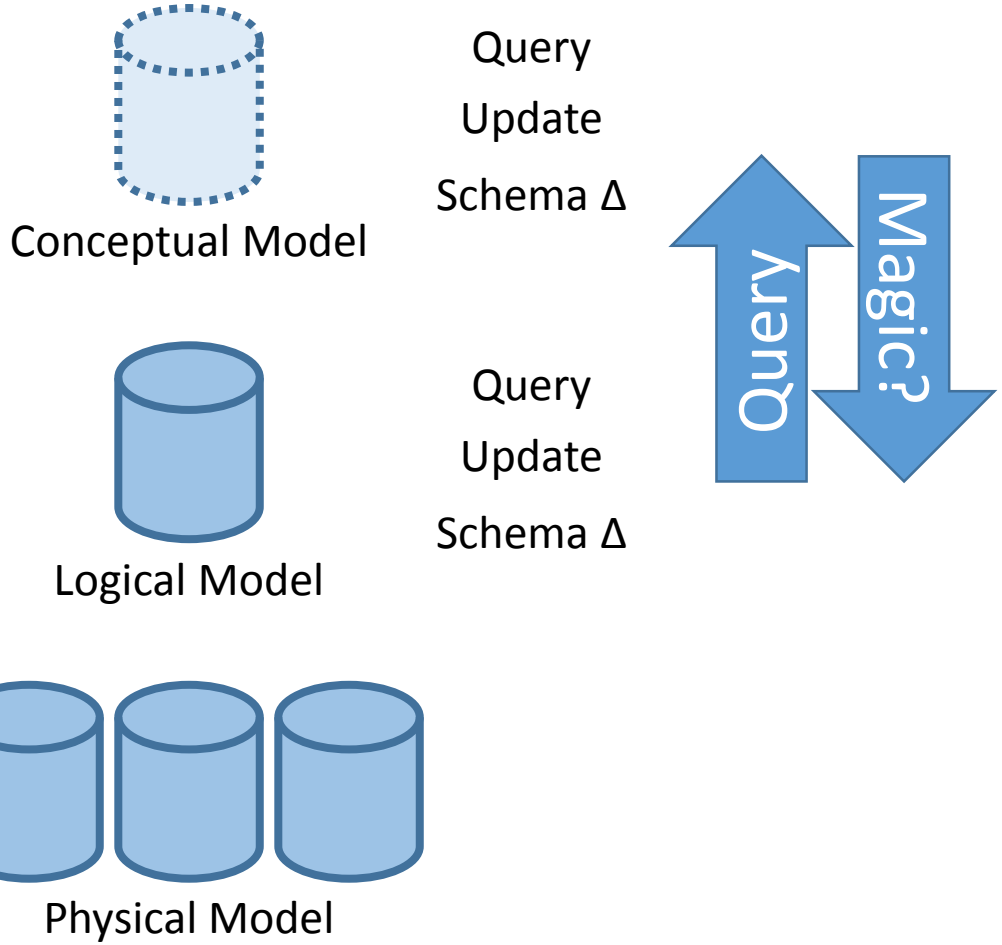
Tables, schema, query surface, regularity

Physical Data Independence

This is where declarative programming is awesome

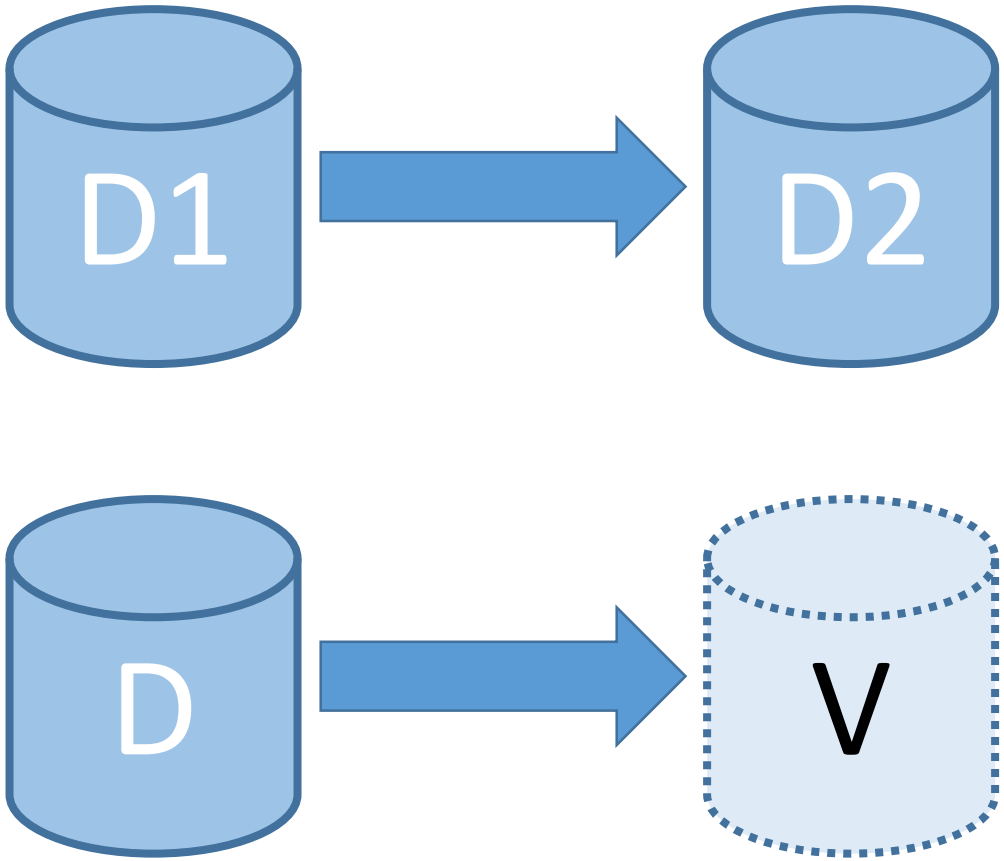
Physical storage, layout on disk, madness

“Logical Data Independence”



“I should be able to use the objects at my layer without needing to worry about the nonsense at the other layers.”

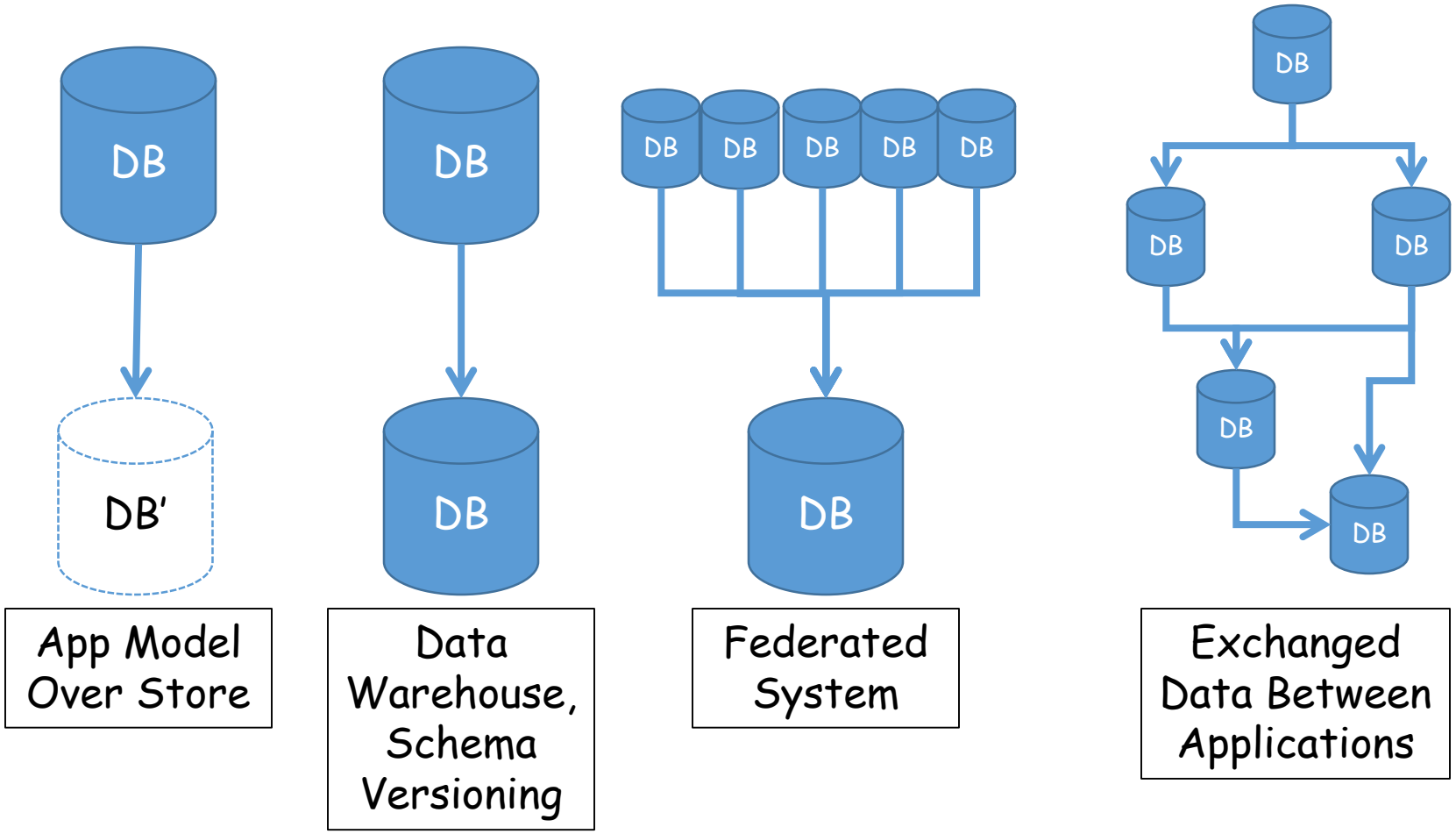
“Mapping”



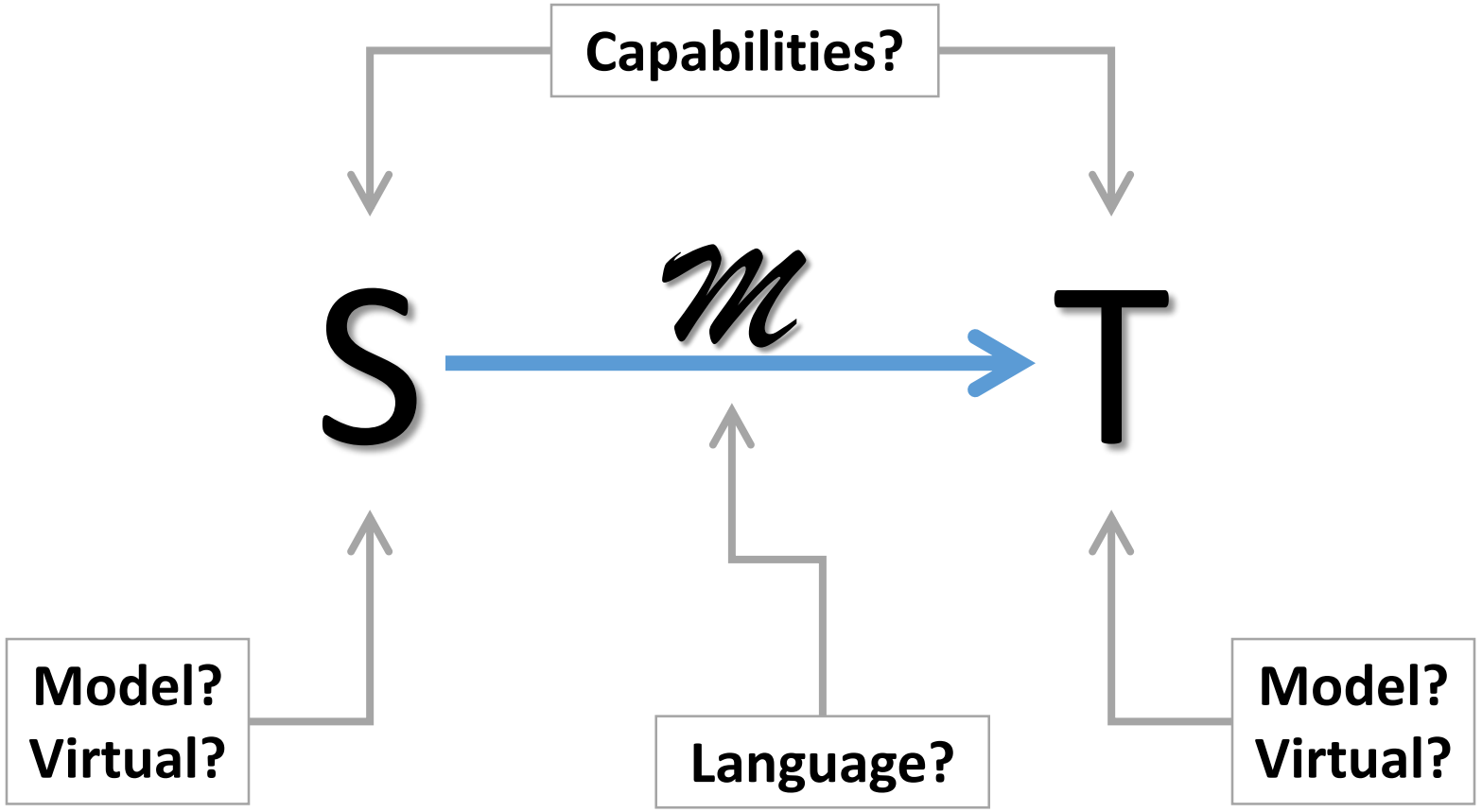
- Use a query as the specification language
- Prefer declarative over procedural
- Uni-directional

Down periscope!

How Does the DB Field Use Mappings?



Metadata Management

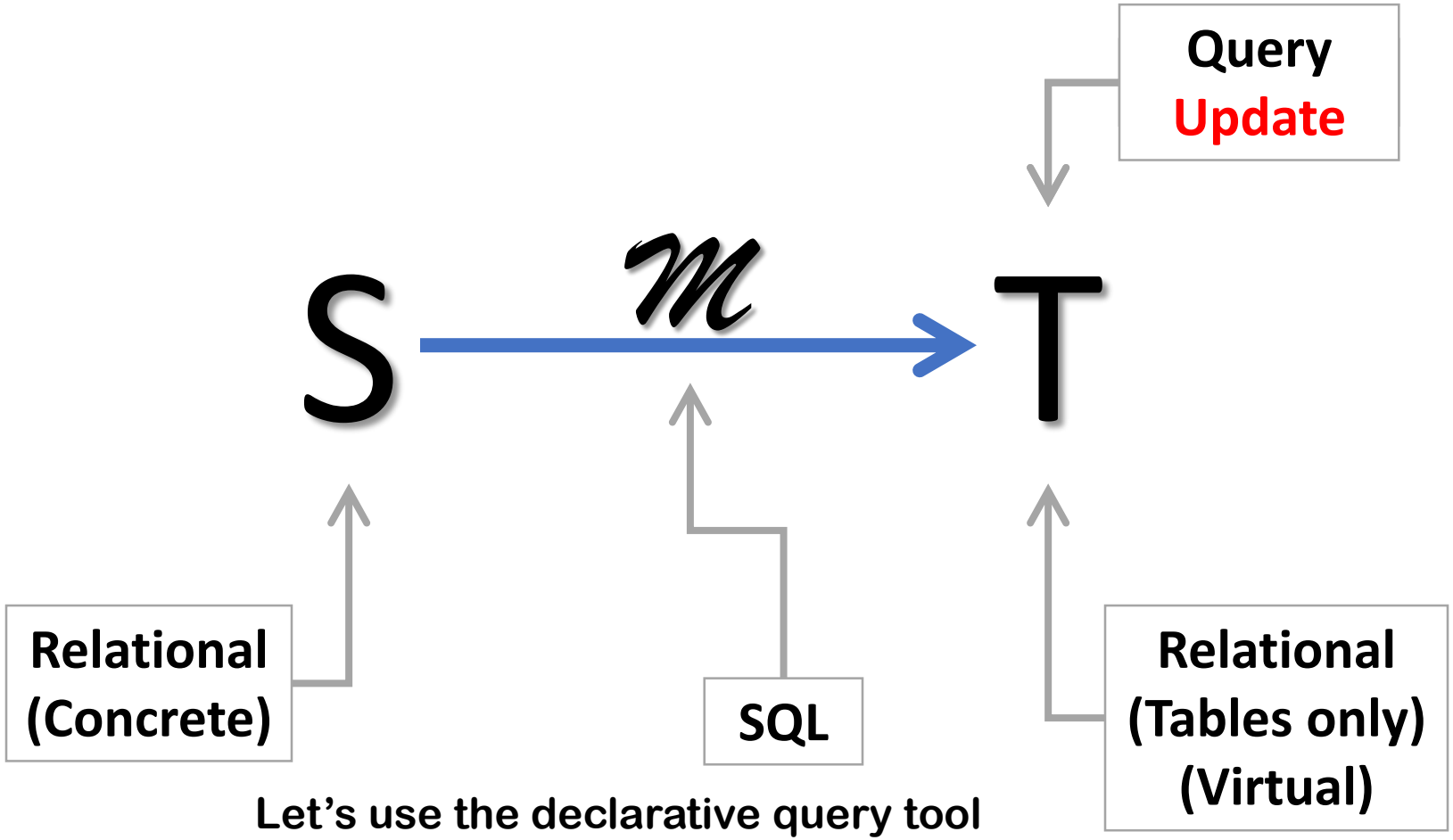


The View Update Problem

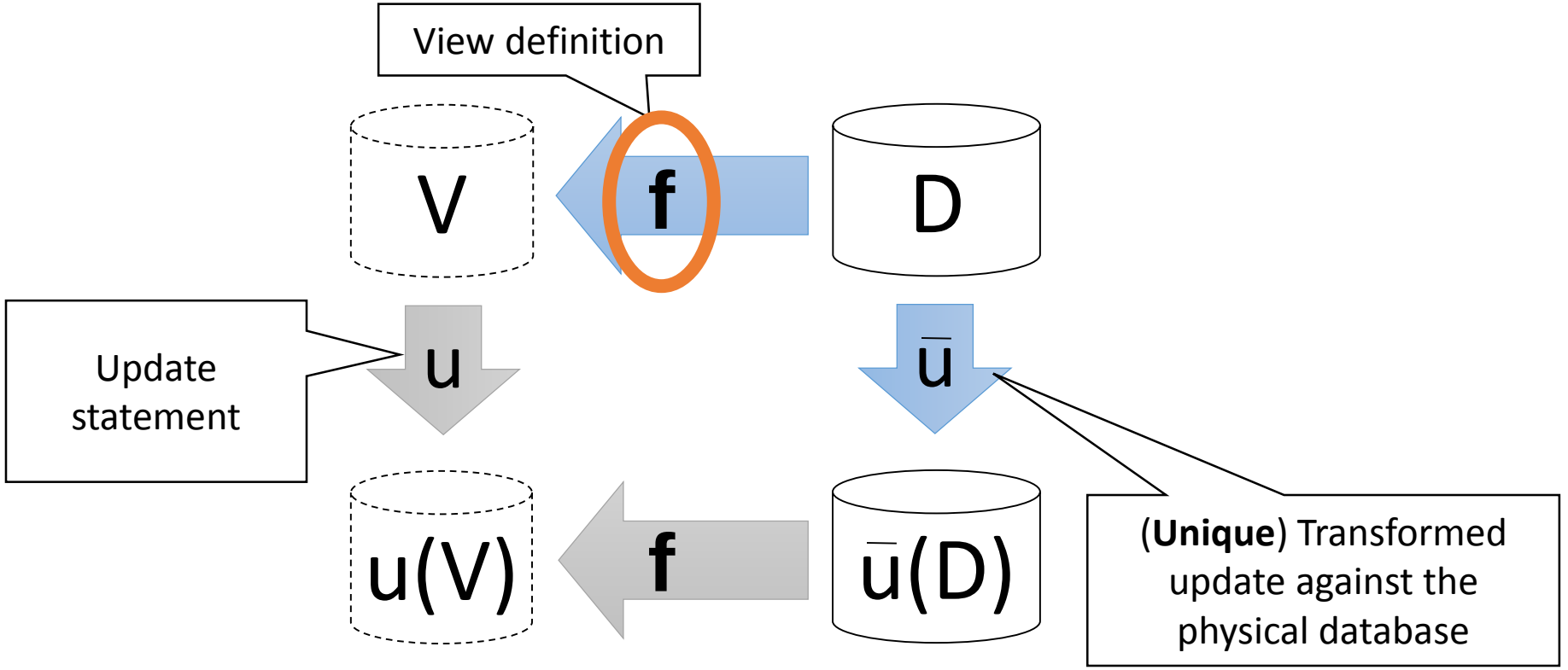
Concrete Database	Application Model, External Schema

- Early work abstracted away the exact language of M, focusing on what it means to be an updatable view
- As work progressed, focus shifted somewhat to a choice of M – SQL – and deciding when an update policy can be computed

The View Update Problem

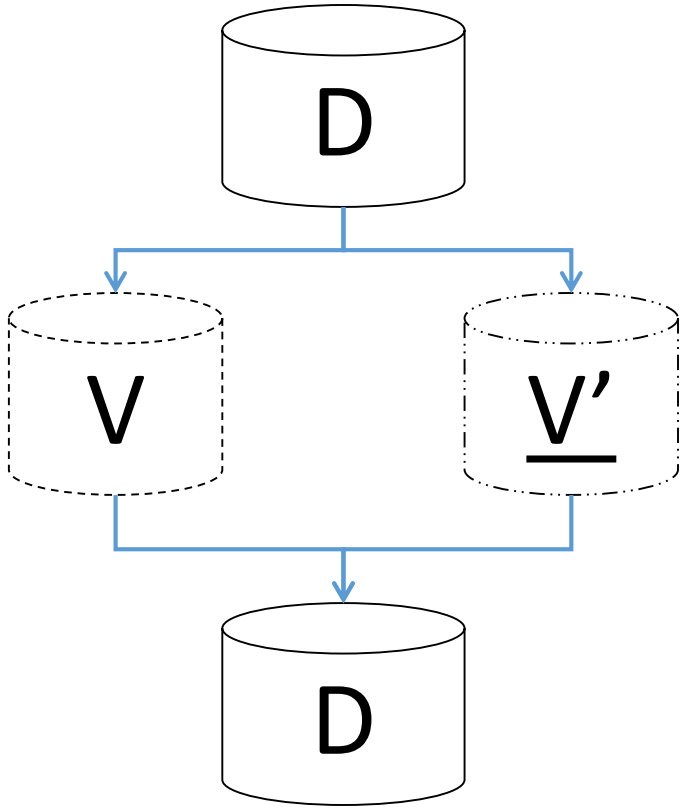


View Updates: The Basics



Update translations available for some syntactic restrictions on f

Constant Complement (Semantics of View Updates)



- Updates leave the view complement unchanged
- Complement may not be unique (must be chosen to determine update semantics)

Update Uniqueness

$$V = T1 \cap T2$$

When I delete a row from V…

- Delete from T1?
- Delete from T2?
- Delete from both?

NB: Not a problem for insertions…

Great! Where Can I Get It?

- Most database vendors do not implement past the SQL92 standard
  - View must have:
    - No set operators
    - No distinct, no grouping
    - No expressions in the SELECT clause
    - No joins or multiple FROM items
    - No smoking, talking, or chewing gum
  - Basically, only simple select/project queries

View Update Limitations (Among Many)

- Large queries are hard to debug (and read!)
- Given a large query, how to report to the user why a query is not updatable?
- DB → Table, not DB → DB
- Syntactic restrictions are very strict

- It is assumed that a query language can make a good view expression language

“Instead Of” Triggers

```
CREATE TRIGGER UPDATE_MY_LOGINS
INSTEAD OF UPDATE ON MY_LOGINS
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
UPDATE USERS U
SET system = n.system, login = n.login, password =
encrypt(n.password)
WHERE system = o.system AND login = o.login AND U.user =
USER$
```

“Instead Of” Triggers

```
CREATE TRIGGER UPDATE_MY_LOGINS
INSTEAD OF UPDATE ON MY_LOGINS
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
UPDATE USERS U
SET system = n.system, login = n.login, password =
encrypt(n.password)
WHERE system = o.system AND login = o.login AND U.user =
USER$
```

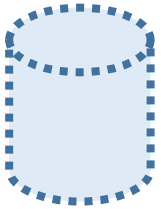
“Instead Of” Triggers

```
CREATE TRIGGER UPDATE_MY_LOGINS
INSTEAD OF UPDATE ON MY_LOGINS
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
UPDATE USERS U
SET system = n.system, login = n.login, password =
encrypt(n.password)
WHERE system = o.system AND login = o.login AND U.user =
USER$
```

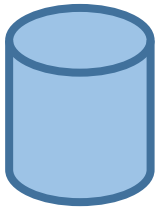
“Instead Of” Triggers

```
CREATE TRIGGER UPDATE_MY_LOGINS
INSTEAD OF UPDATE ON MY_LOGINS
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
UPDATE USERS U
SET system = n.system, login = n.login, password =
encrypt(n.password)
WHERE system = o.system AND login = o.login AND U.user =
USER$
```

The Real World (and a large opportunity)

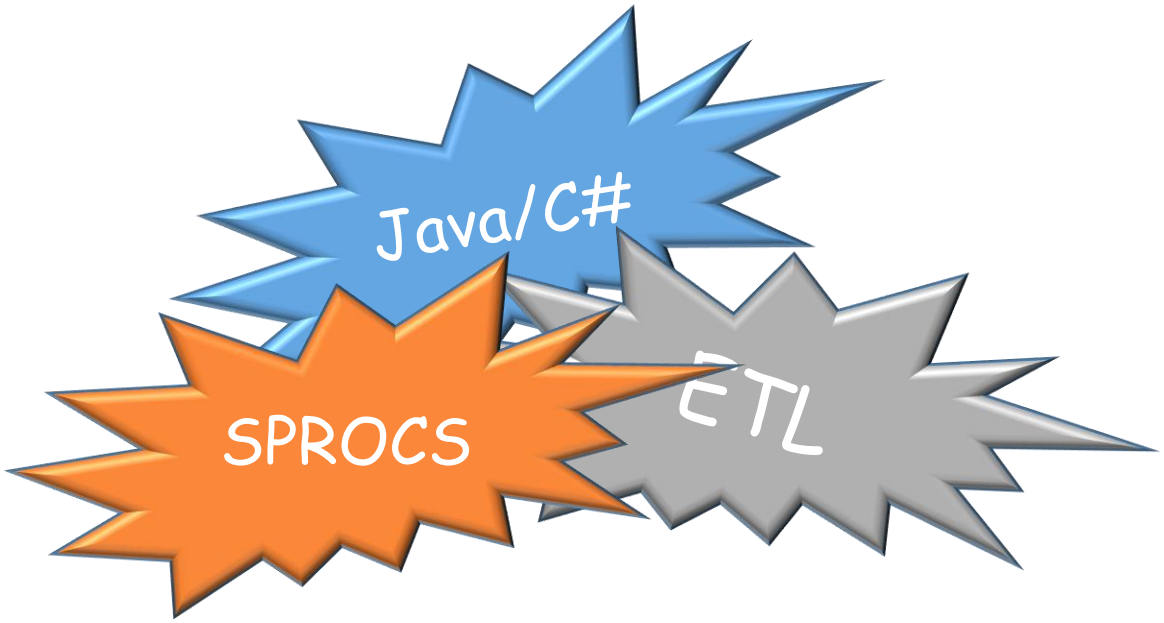


Conceptual Model



Logical Model

- Too expressive for mapping language (e.g., pivot)
- Too hard to define inverse of mapping fragment
- Too difficult to enforce policies (e.g., immutability)
- Mapping consistency against evolution is hard



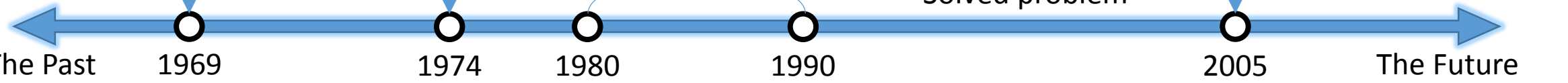
Timeline

R. Fagin, P. Kolaitis, R. Miller, and L. Popa. "**Data exchange**: semantics and query answering." Theoretical Computer Science, 336(1):89–124, 2005.

- Relational Model
- Relational Calculus
- Relational Algebra

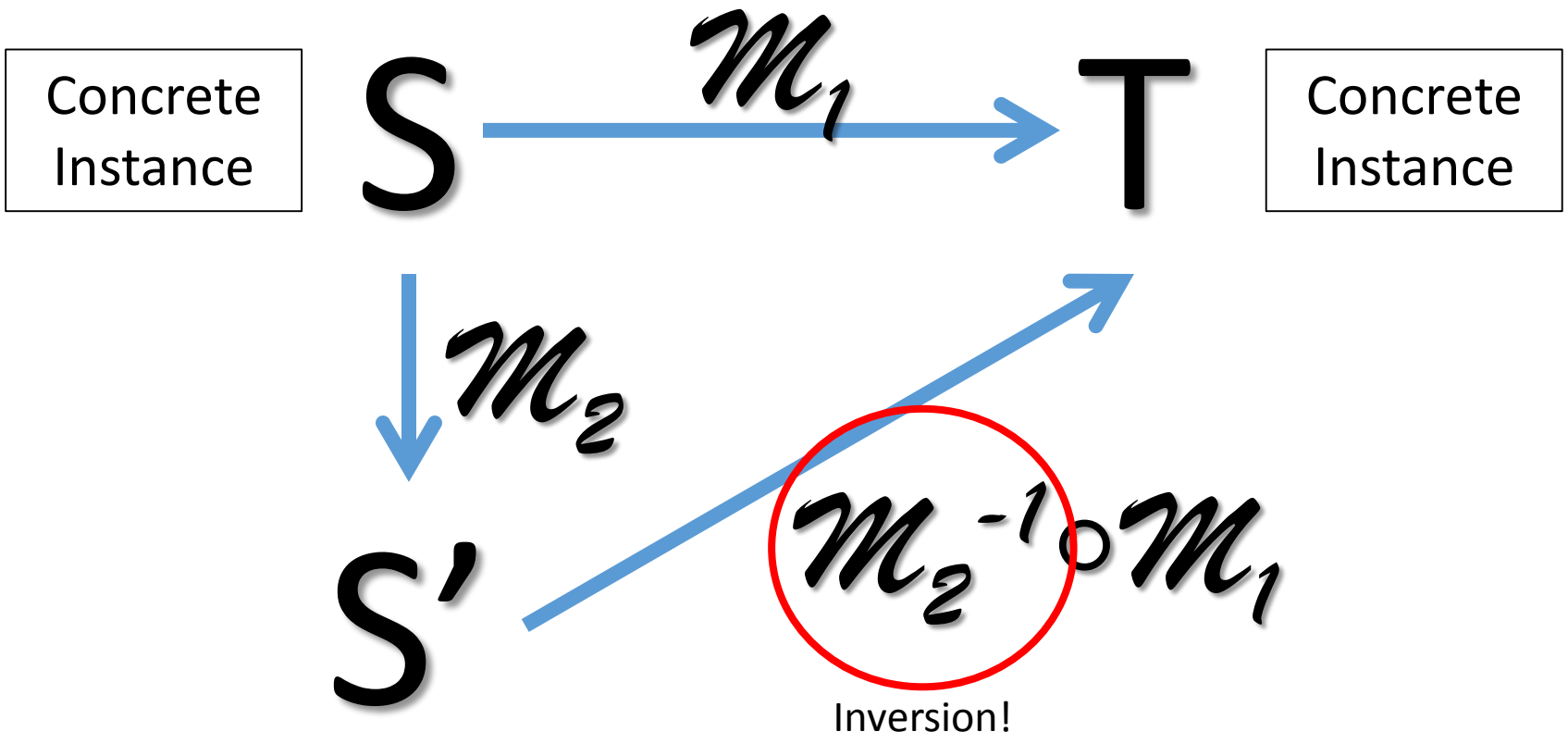
- SQL

- View updates
- Constant complement
- Query containment



"This is relevant to my interests."

Data Exchange



$$
\Big(\;
\begin{array}{l}
\forall \mathrm{src}\,\forall \mathrm{dest}\,\forall \mathrm{airl}\,\forall \mathrm{dep}\,\Big(\,\mathrm{FLIGHT}(\mathrm{src},\mathrm{dest},\mathrm{airl},\mathrm{dep}) \longrightarrow \\
\quad \exists \mathrm{f\#}\,\exists \mathrm{arr}\,\big(\,\mathrm{ROUTES}(\mathrm{f\#},\mathrm{src},\mathrm{dest}) \wedge \mathrm{INFO\_FLIGHT}(\mathrm{f\#},\mathrm{dep},\mathrm{arr},\mathrm{airl})\,\big)\Big) \\
\\
\forall \mathrm{city}\,\forall \mathrm{dest}\,\forall \mathrm{airl}\,\forall \mathrm{dep}\,\forall \mathrm{country}\,\forall \mathrm{popul}\,\Big(\\
\quad \mathrm{FLIGHT}(\mathrm{city},\mathrm{dest},\mathrm{airl},\mathrm{dep}) \wedge \mathrm{GEO}(\mathrm{city},\mathrm{country},\mathrm{popul}) \longrightarrow \\
\qquad \exists \mathrm{phone}\,\mathrm{SERVES}(\mathrm{airl},\mathrm{city},\mathrm{country},\mathrm{phone})\,\Big) \\
\\
\forall \mathrm{city}\,\forall \mathrm{dest}\,\forall \mathrm{airl}\,\forall \mathrm{dep}\,\forall \mathrm{country}\,\forall \mathrm{popul}\,\Big(\\
\quad \mathrm{FLIGHT}(\mathrm{src},\mathrm{city},\mathrm{airl},\mathrm{dep}) \wedge \mathrm{GEO}(\mathrm{city},\mathrm{country},\mathrm{popul}) \longrightarrow \\
\qquad \exists \mathrm{phone}\;\mathrm{SERVES}(\mathrm{airl},\mathrm{city},\mathrm{country},\mathrm{phone})\,\Big)
\end{array}
\;\Big) \text{-} 1
$$

Maximal Recovery

Given a mapping f:

Best case: find f^{-1} such that $f^{-1} \circ f \equiv id$ (Fagin Inverse)

Alternative: find f^{-1} such that $f^{-1} \circ f \cong id$ relative to some equivalence

Maximal recovery: compute f^{-1} such that $f^{-1} \circ f = g$, where:
- If f is invertible, then g=id
- If f is not invertible, then g is the function that recovers at least as much sound data as any other function

More Maximal Recovery

The good news:

- The maximal recovery of f is computable from f. (!)

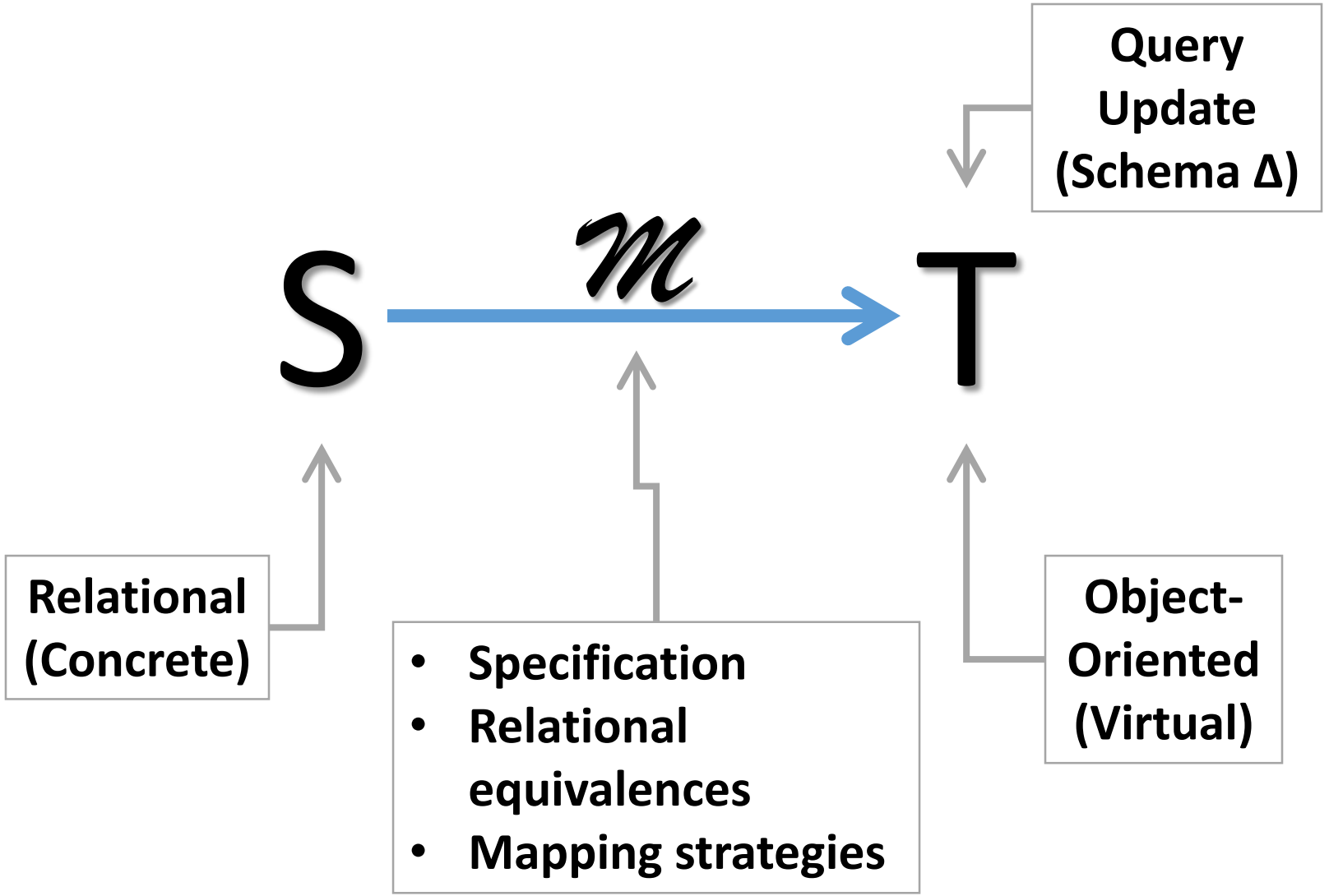
The bad news:

- The inverse of f is not necessarily expressible as an st-tgd.
- Some fairly simple mappings do not have an inverse and must rely on maximal recovery.

Object-Relational Mappings: Hi Richard!

- Applications written in an object-oriented language have object-oriented data tiers
- Persistence is a relational database
- “Impedance mismatch”
  - Map object constructs to relational constructs
- MUST BE BIDIRECTIONAL (Full logical data independence)
- Spanning models

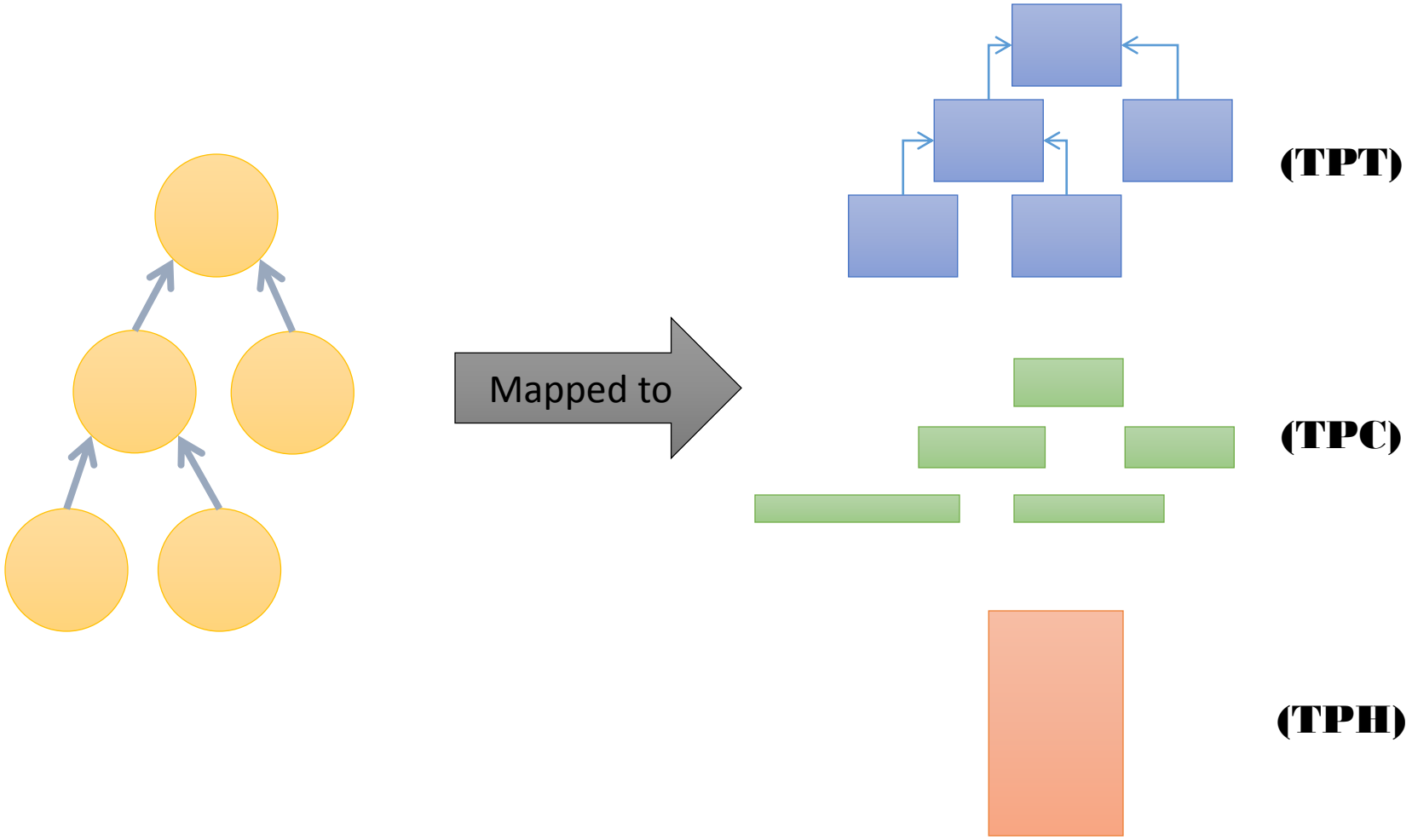
Object-Relational Mappings



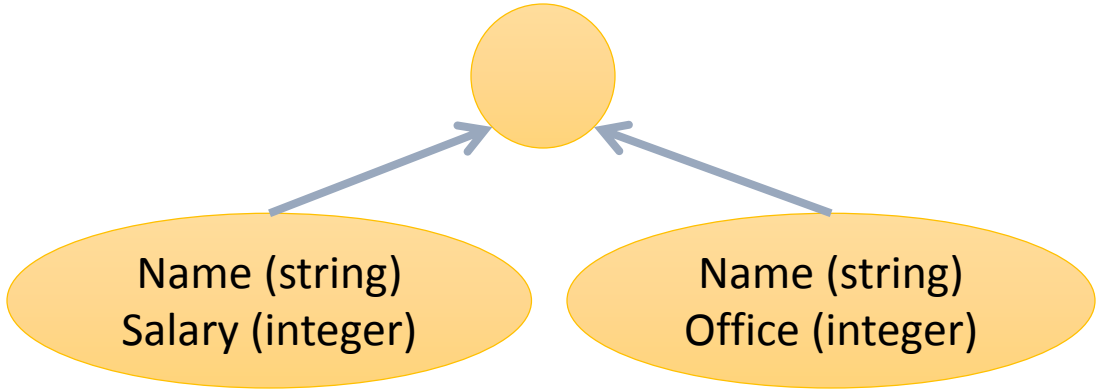
An O-R Mapping Is…

- … generally an operational specification rather than a declarative query or set of queries
- … tailored more to the purpose of mapping inheritance and relationships to relations rather than a general-purpose mapping

Mapping Patterns



Mapping Patterns: TPH Sub-Categories



Fully disjoint

Name1 (string)
Name2 (string)
Salary (integer)
Office (integer)

Clear column provenance

Reuse by column

Name (string)
Salary (integer)
Office (integer)

Clear name reuse

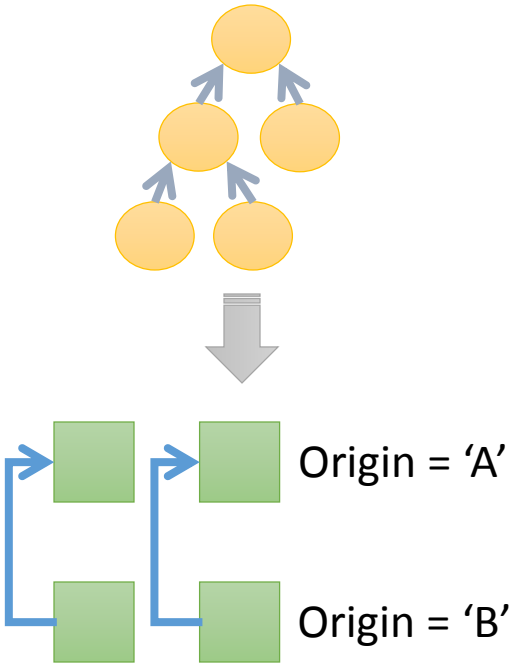
Reuse by domain

String1 (string)
Integer1 (integer)

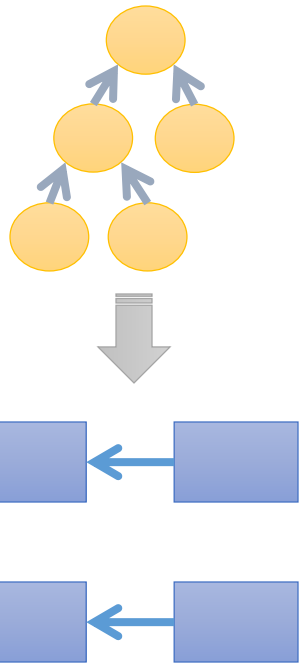
Maximum data density

Mapping Patterns: Etc.

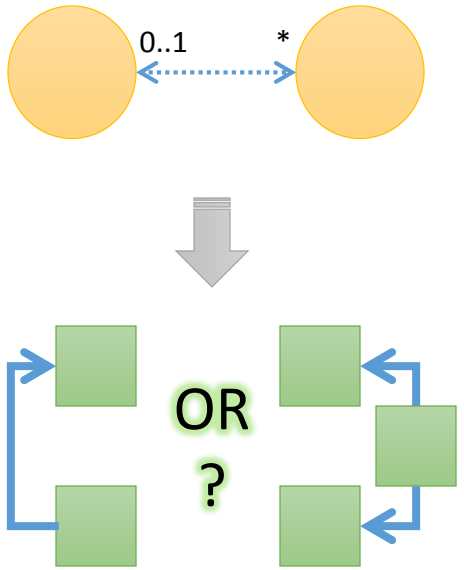
Horizontal Partitioning



Vertical Partitioning



Association Join Tables



ORM Product Space

- Ruby on Rails
- Hibernate/NHibernate
- SQLAlchemy
- Entity Framework
- TopLink

- Some major tradeoffs:
  - Expressiveness
  - Specification style

Hibernate Example

```
<hibernate-mapping>
 <class name="eg.hibernate.mapping.dataobject.Person" table="TB_PERSON" polymorphism="implicit">
  <id name="id" column="ID">
   <generator class="assigned"/>
  </id>
  <set name="rights" lazy="false">
   <key column="REF_PERSON_ID"/>
   <one-to-many class="eg.hibernate.mapping.dataobject.Right" />
  </set>
  <joined-subclass name="eg.hibernate.mapping.dataobject.Individual"
 table="TB_INDIVIDUAL">
   <key column="id"/>
   <property name="firstName" column="FIRST_NAME" type="java.lang.String" />
   <property name="lastName" column="LAST_NAME" type="java.lang.String" />
  </joined-subclass>
  <joined-subclass name="eg.hibernate.mapping.dataobject.Corporation"
 table="TB_CORPORATION">
   <key column="id"/>
   <property name="name" column="NAME" type="string" />
   <property name="registrationNumber" column="REGISTRATION_NUMBER" type="string" />
  </joined-subclass>
 </class>
</hibernate-mapping>
```

Client Class

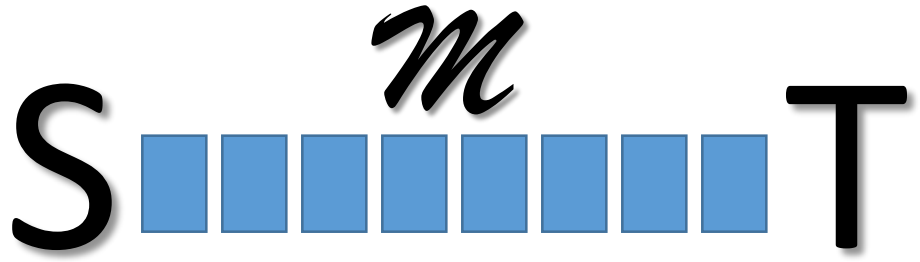
Store Table

TPT-Style Mapping

TPT-Style Mapping

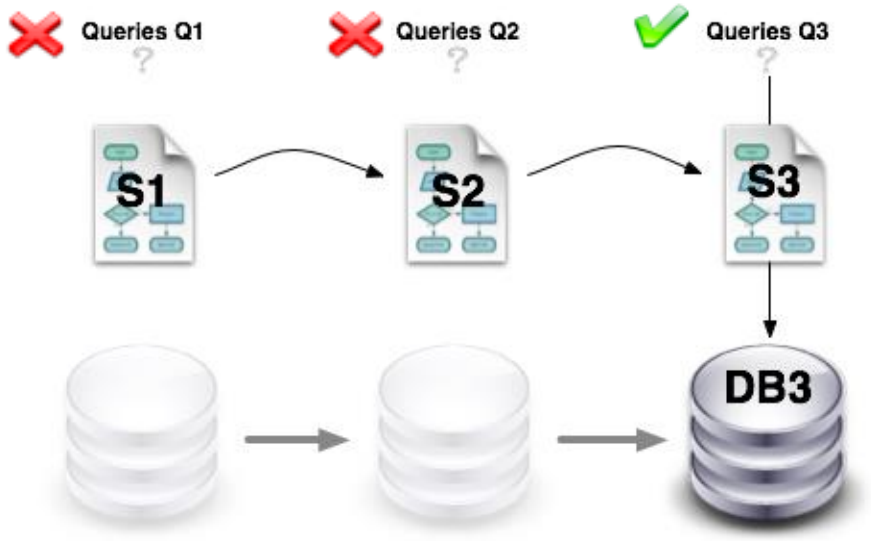
XML fragments almost correspond to individual O-to-R transformations

In General, Two Approaches



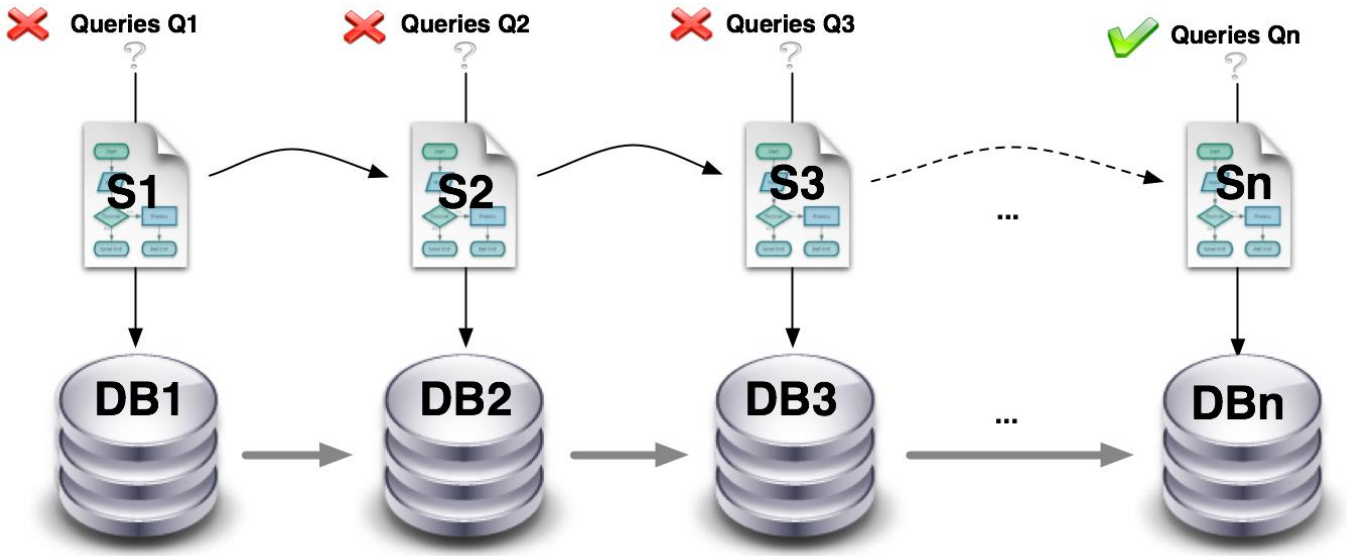
“Interactivity”

Schema Evolution: common practice



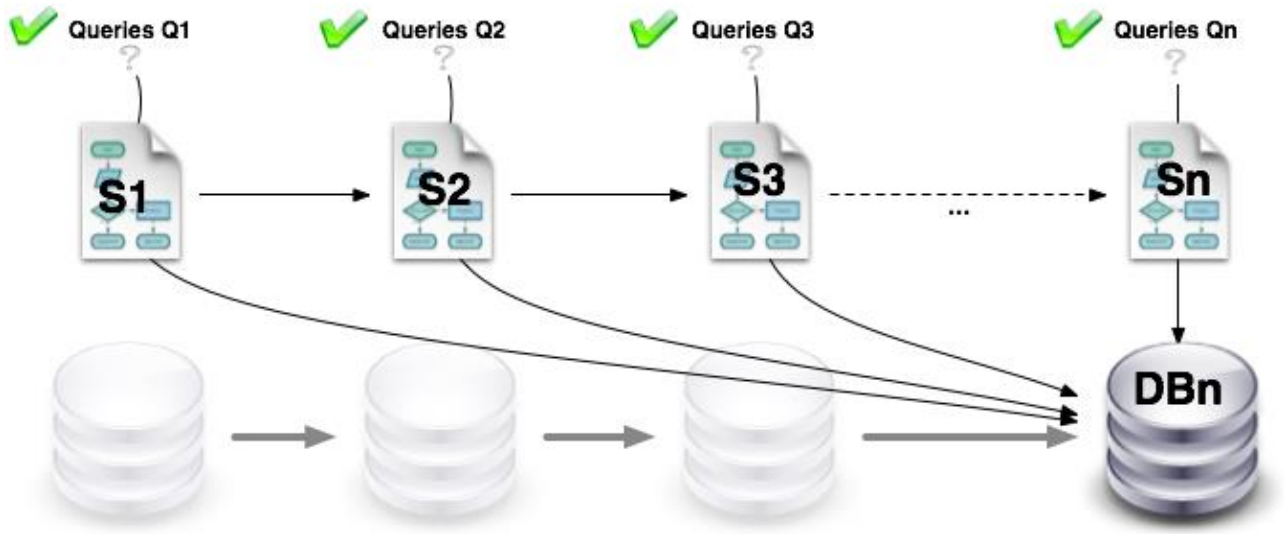
- **Evolution in the *real* world:**
  - The DBA defines an SQL DDL script modifying *S2* into *S3*
  - The DBA defines an SQL DML script migrating data from *DB2* to *DB3*
  - **Queries** in *Q2* might **fail**, the DBA **adapts** them **manually** as in *Q3* = *Q2'* + *Q3_new* (new queries added on *S3*)

Schema Evolution: common practice



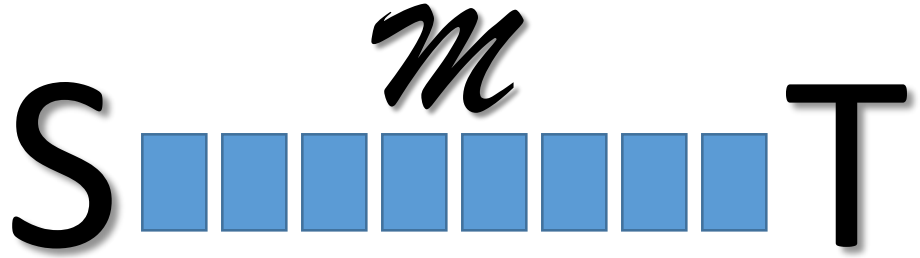
- DB Administrator (DBA) nightmares:
  - **Data Migration:** Data loss, redundancy, efficiency of the migration, efficiency of the new design
  - **Impact on Queries** and applications

Schema Evolution: Ideal World



- Evolution in an *ideal* world:
  - Evolution design is **assisted** and **predictable**
  - **Data migration** scripts are **generated** automatically
  - Legacy **Queries (and updates, views, integrity constraints,…)** are **automatically adapted** to fit the new schema

Not Our First Rodeo



- S and T may not belong to the same data model
- Assume the existence of a union model
- S and T are just “special cases” in the union model, conforming to one or the other of the union summands
- NO UNIFIED THEORY

3

Can’t we all just get along?

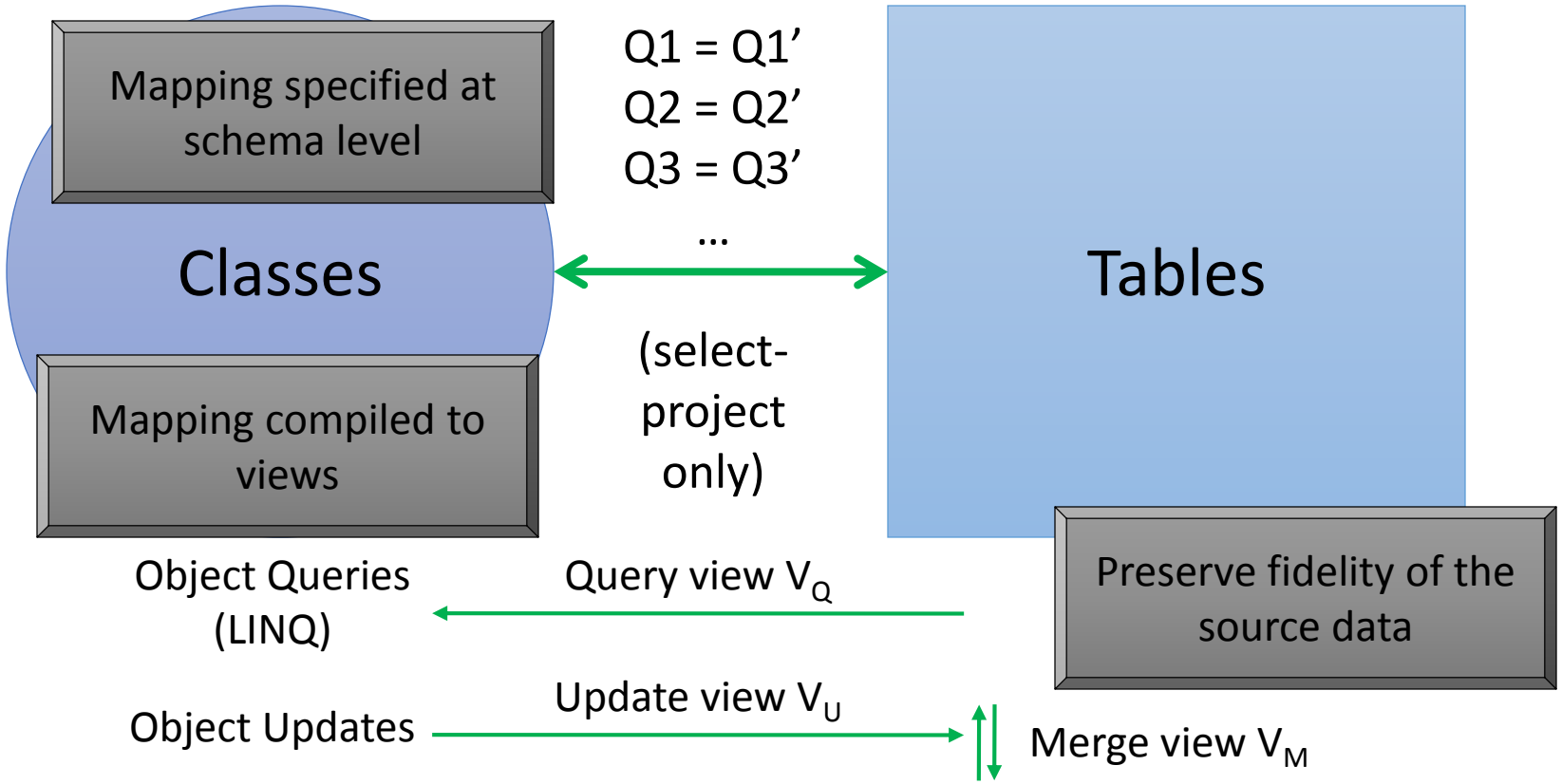
Erik Meijer, via Twitter:

“Not only was Ted Codd not a developer; our friend the Reverend Thomas Bayes wasn't one either. We are still suffering from the side-effects.”

Entity Framework (EF): A Brief Overview

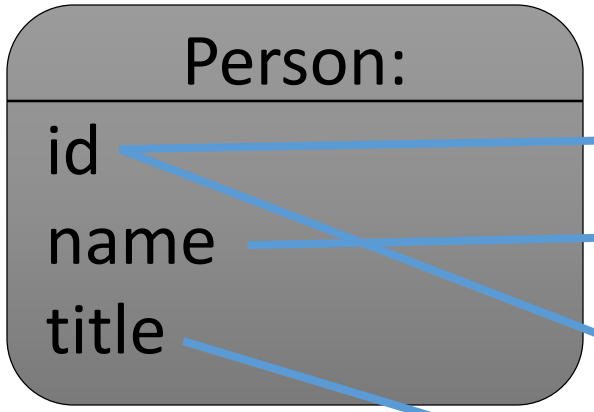
Client-side (Objects):

Store side (Relations):



EF Simple Example

Client-side (Classes):



Store side (Relations):

Person1(

id integer PRIMARY KEY,

name varchar(50),

)

Person2(

id integer PRIMARY KEY,

title varchar(50),

details varchar(2000)

)

$\pi_{id, name}$ Person = $\pi_{id, name}$ Person1

$\pi_{id, title}$ Person = $\pi_{id, title}$ Person2

Person = $\pi_{id, name, title}$ Person1 $\bowtie$ Person2

Entity Framework: Major Results

- Validation procedure ensures that a collection of mapping fragment roundtrips
  - Each client state maps to a valid state
  - Client state travel to store and back is invariant
  - Guarantees query and update safety

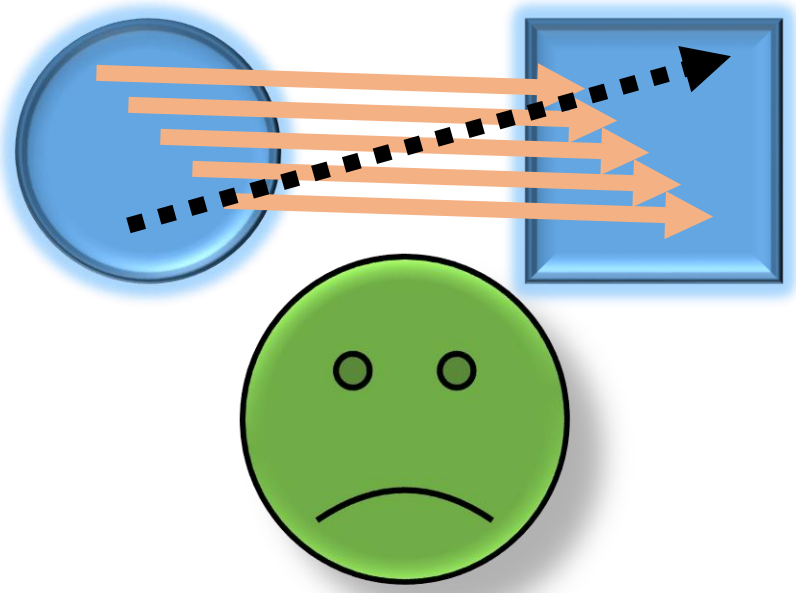
- Mapping compilation procedure expressive enough for common mapping scenarios, and many uncommon ones
  - All of the mapping schemes previously noted

Entity Framework Opportunities

Query View
+
Update View
+
Merge View

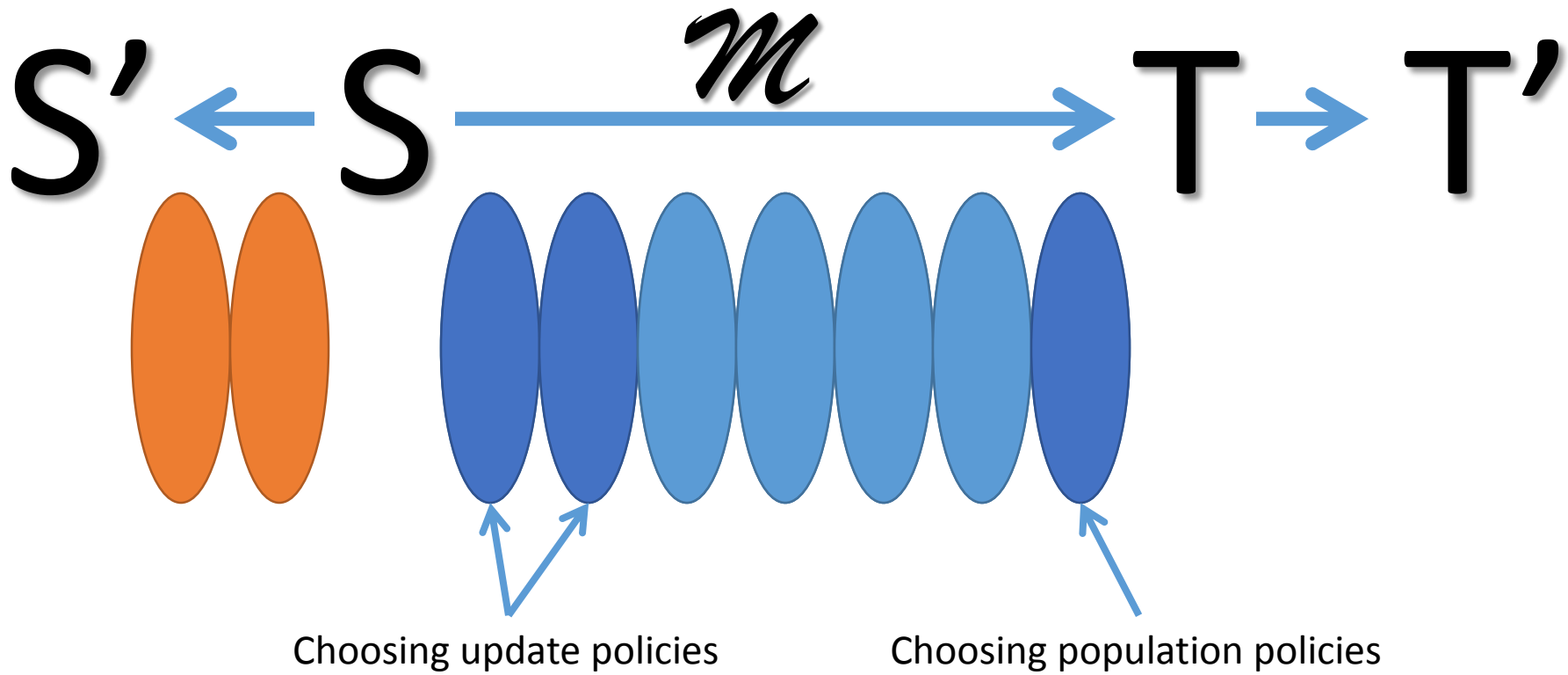
≡

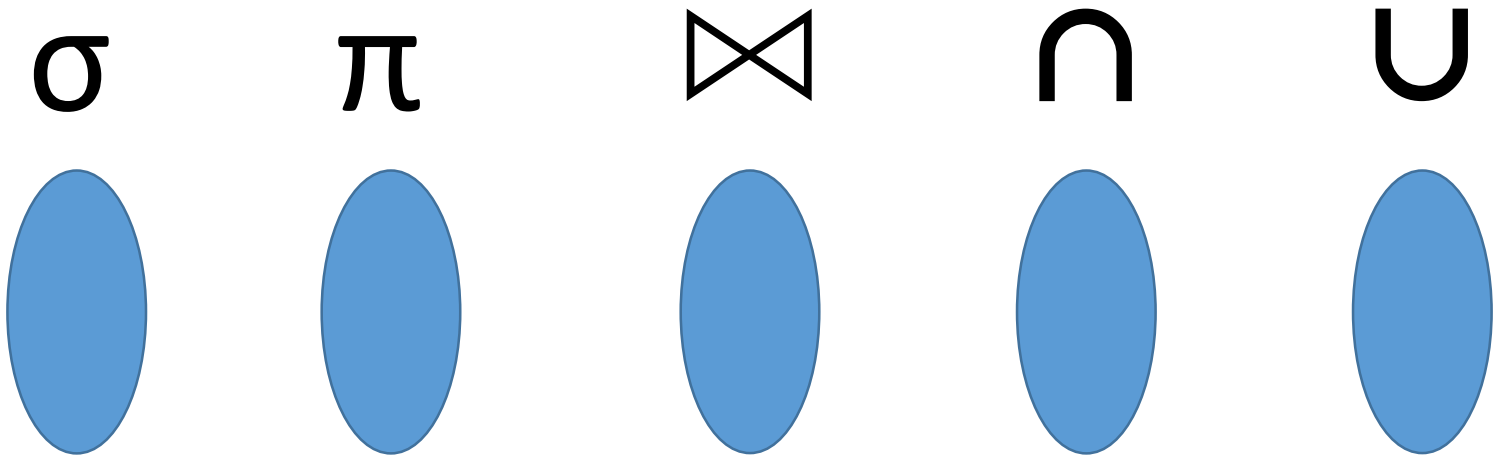
PutGet
+
GetPut



Invalid mappings make me sad

Can TGGs do a better job of construction and debugging?


S'
S
$\mathscr{M}$
T
T'
Choosing update policies
Choosing population policies

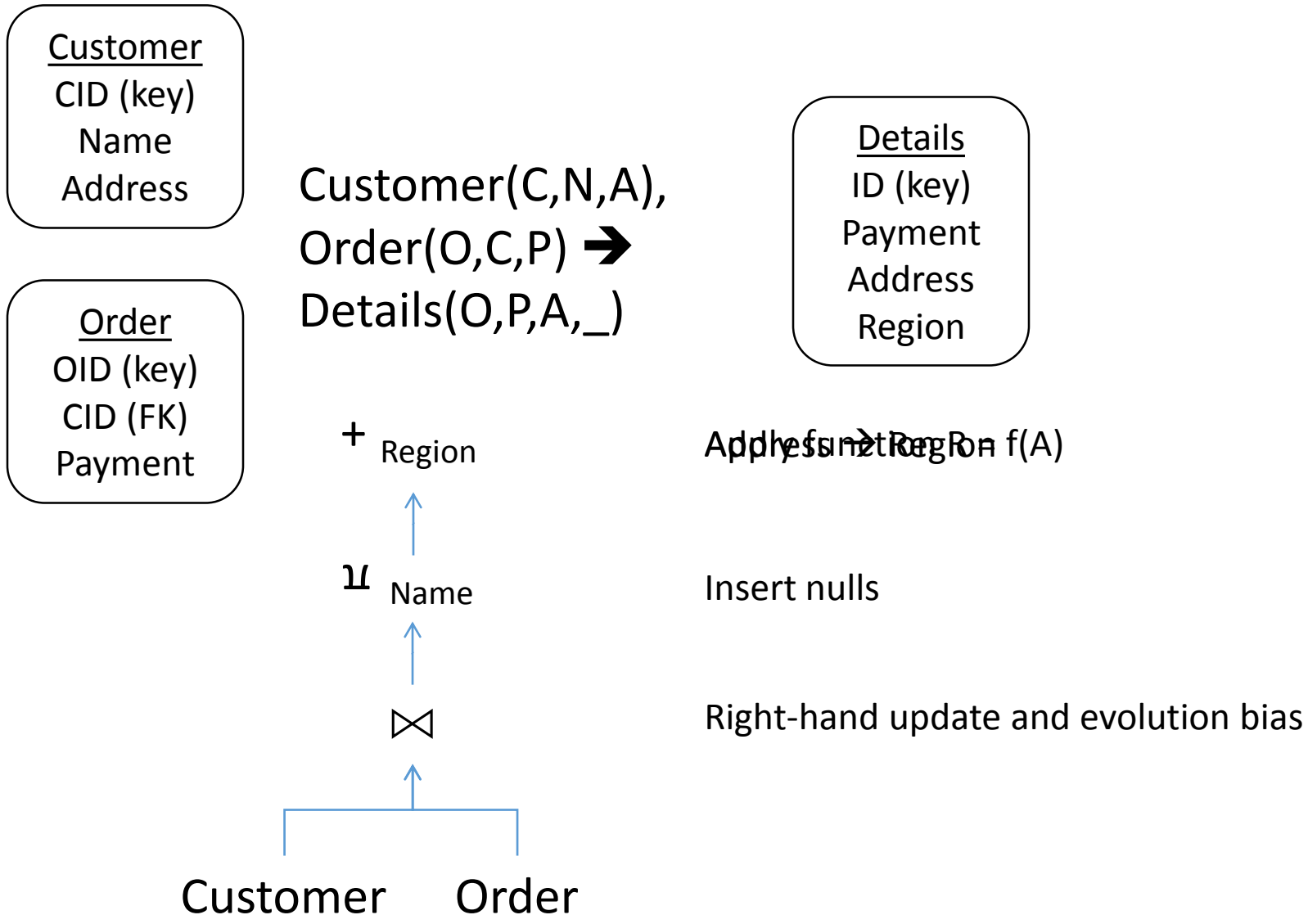


Data updates based on:

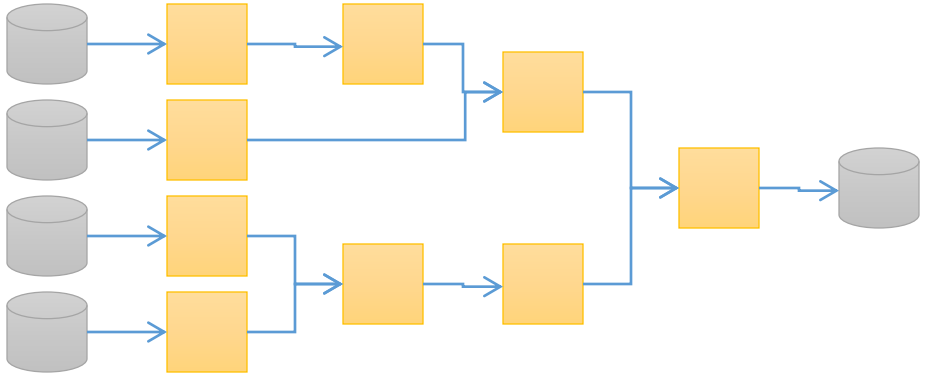
- Functional dependencies (default)
- Environment variables
- Nulls or distinguished values
- Direction bias

Schema update policies/alternatives

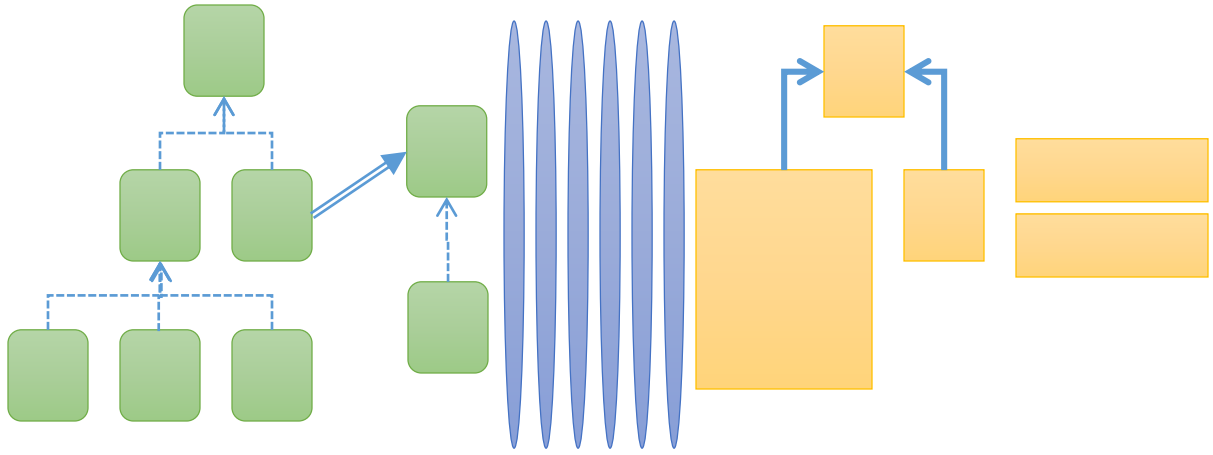
Some introductory work has been done in this space, but at a speculative level. Let's solve this thing!



Extract
Transform
Load



Object
Relational
Mapping



See Database Researchers In Their Natural Habitat!

BX 2014: Deadline Dec. 7!

Tutorial deadline Jan. 6!

Thank You!