ILLINOIS INSTITUTE OF TECHNOLOGY

CS520 Data Integration, Warehousing, and Provenance

1. Introduction

IIT DBGroup

Boris Glavic

http://www.cs.iit.edu/~glavic/

http://www.cs.iit.edu/~glavic/cs520/

http://www.cs.iit.edu/~dbgroup/

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

0) Course Info
1) **Introduction**
2) Data Preparation and Cleaning
3) Schema matching and mapping
4) Virtual Data Integration
5) Data Exchange
6) Data Warehousing
7) Big Data Analytics
8) Data Provenance

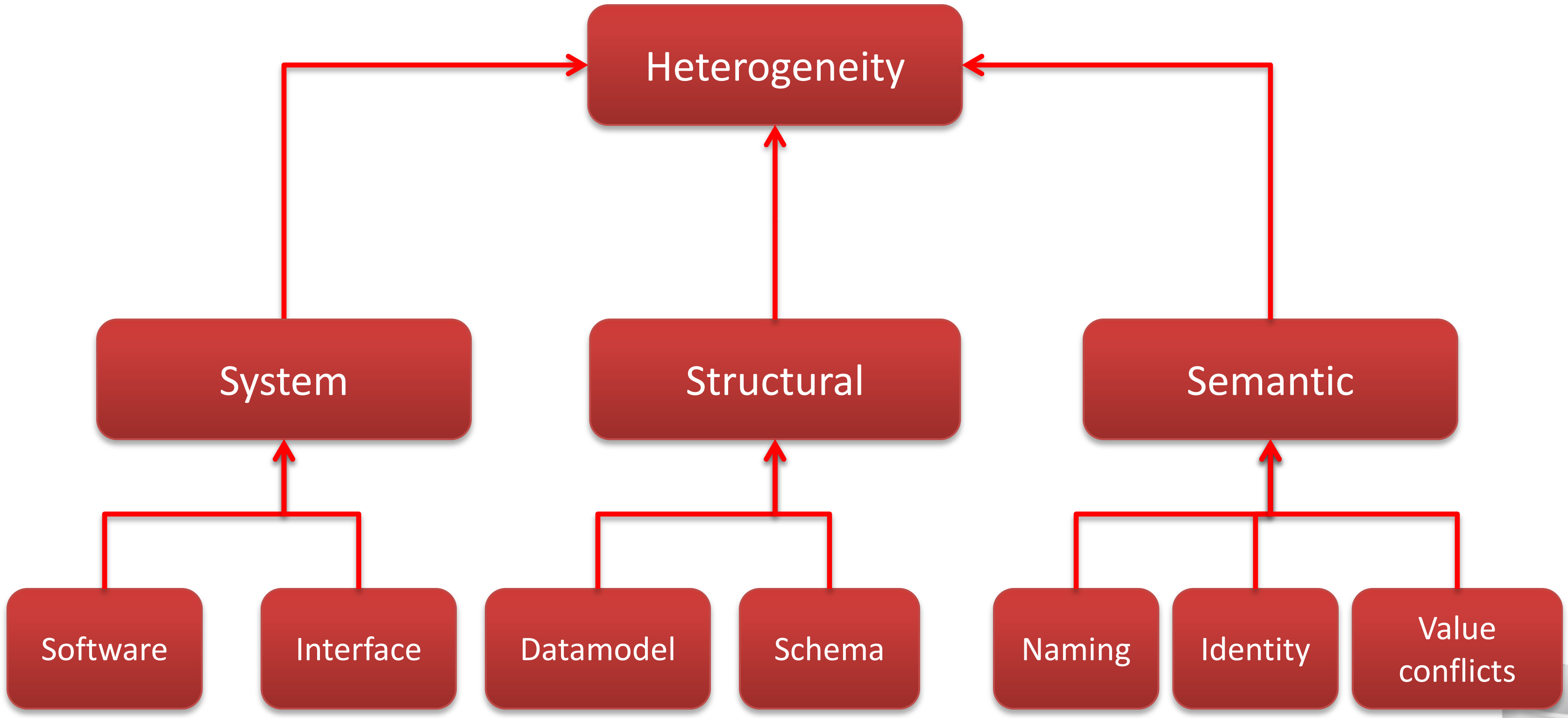
Overview

- Topics covered in this part
  - Heterogeneity and Autonomy
  - Data Integration Tasks
  - Data Integration Architectures (Methods)
  - Some Formal Background (sorry!)

1.1 Heterogeneity +Autonomy

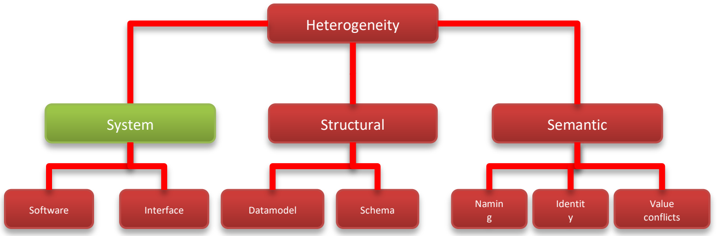
ILLINOIS INSTITUTE OF TECHNOLOGY

- Taxonomy of Heterogeneity



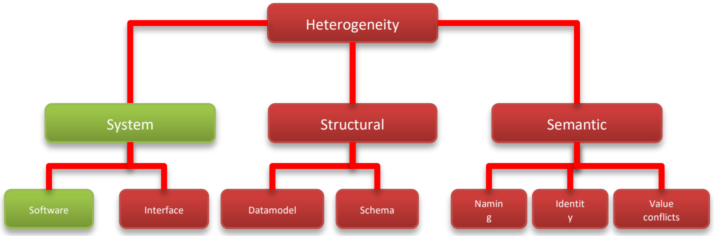
1.1 System Heterogeneity

- Hardware/Software
  - Different hardware capabilities of sources
  - Different protocols, binary file formats, …
  - Different access control mechanism
- Interface Heterogeneity
  - Different interfaces for accessing data from a source
    - HTML forms
    - XML-Webservices
    - Declarative language



1.1 System Heterogeneity

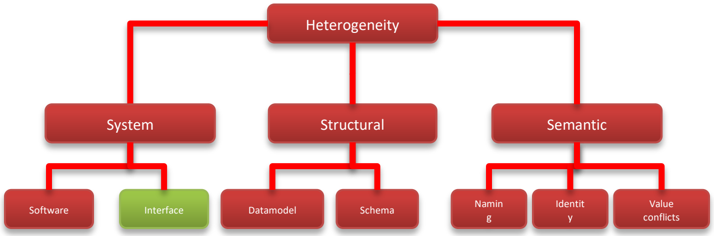
- Hardware/Software
  - Different hardware capabilities of sources
    - **Mobile phone vs. server:** Cannot evaluate cross-product of two 1GB relations on a mobile phone
  - Different protocols, binary file formats, …
    - **Order information stored in text files:** line ending differs between Mac/Window/Linux, character encoding
  - Different access control mechanism
    - **FTP-access to files:** public, ssh authentication, ..



1.1 System Heterogeneity

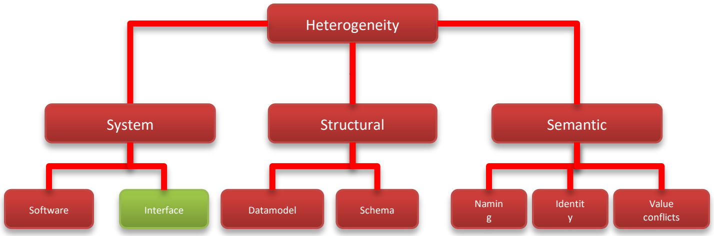
ILLINOIS INSTITUTE OF TECHNOLOGY

- Interface Heterogeneity
  - Different interfaces for accessing data from a source
    - HTML forms
    - Services (SOA)
    - Declarative language
    - Files
    - Proprietary network protocol
    - …



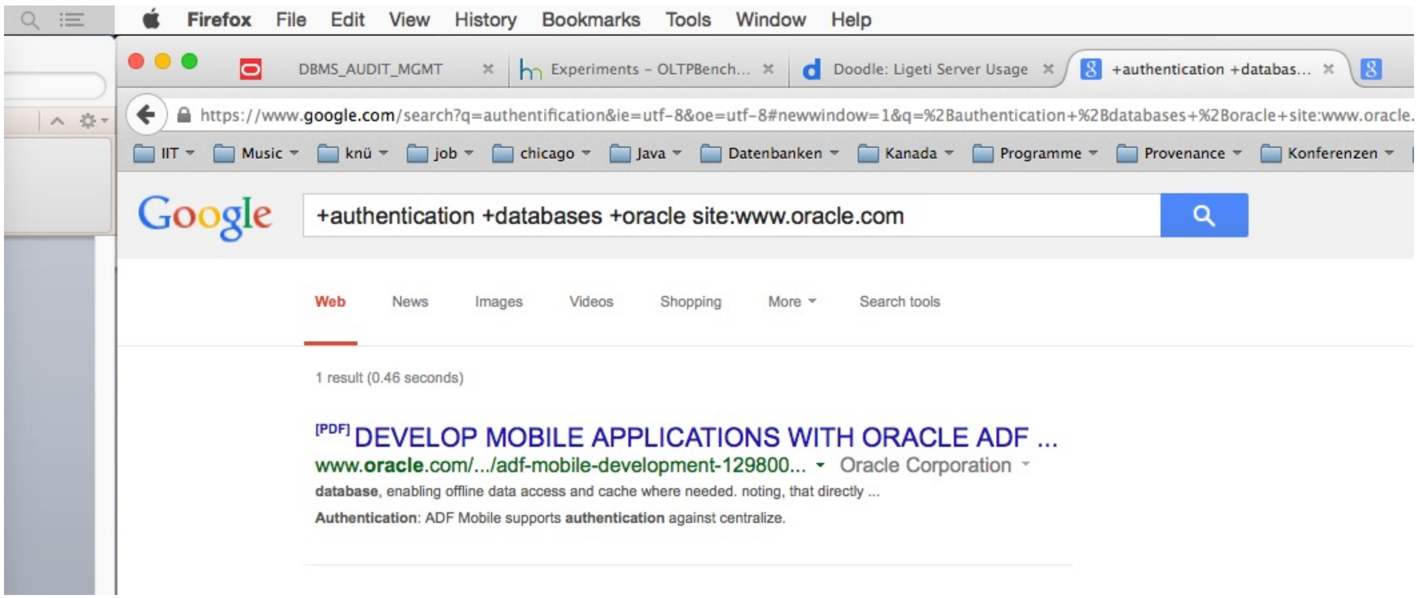
1.1 System Heterogeneity

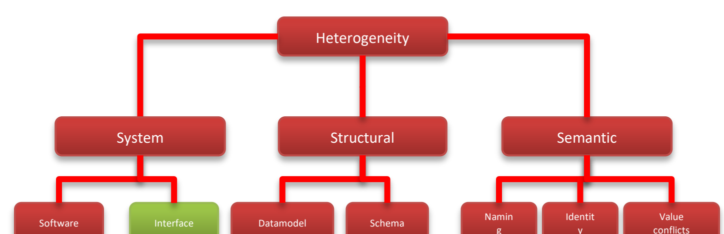
- Interface Heterogeneity – Expressiveness
  - Keyword-search vs. query language
  - **Predicates**: equality (=), inequality (<, !=)
  - **Logical connectives**: conjunctive (AND), disjunctive (OR), negation
  - **Complex operations**: aggregation, quantification
  - **Limitations**: restriction to particular tables, predicates, fixed queries with parameters, …



1.1 System Heterogeneity

- Interface Heterogeneity – Examples
  - Google search (+/-, site:, intitle:, filetype:

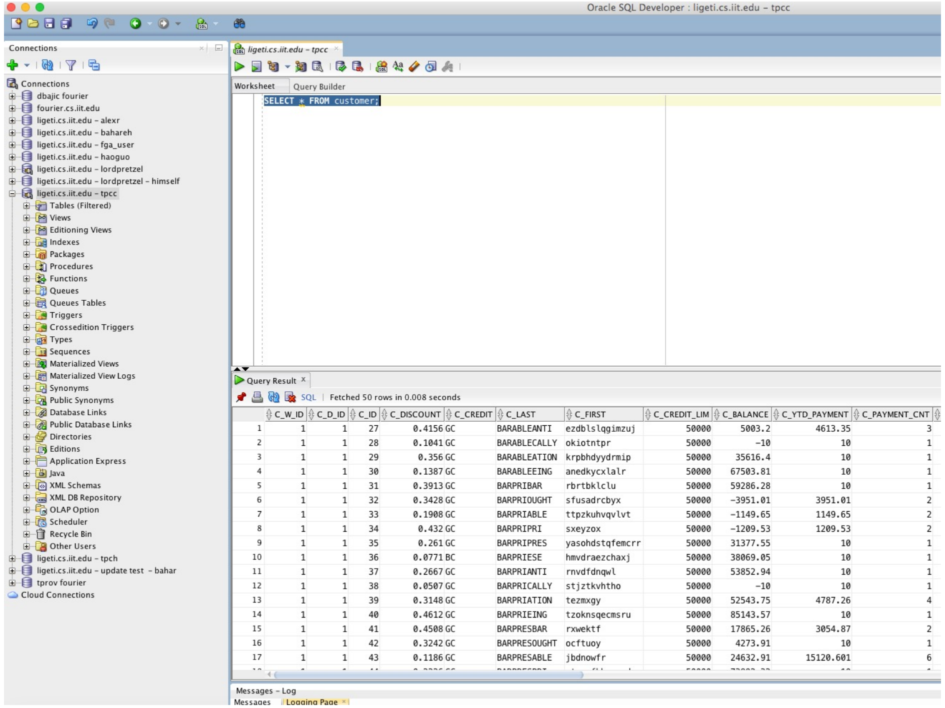


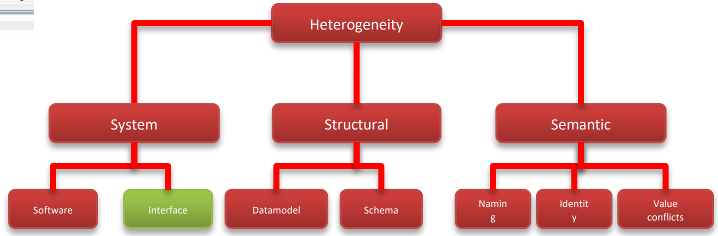


1.1 System Heterogeneity

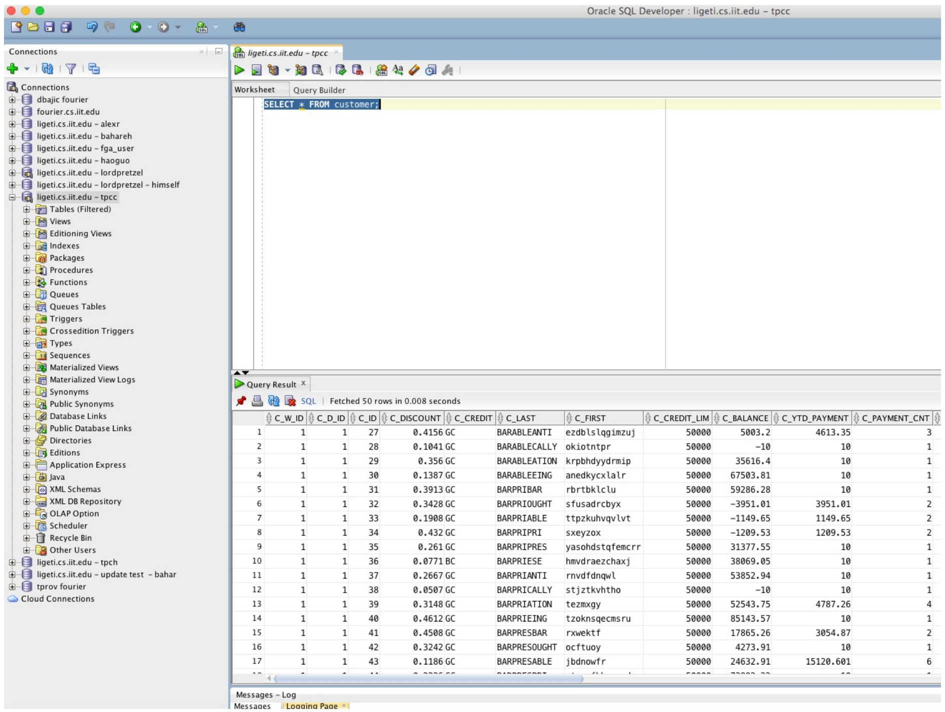
ILLINOIS INSTITUTE OF TECHNOLOGY

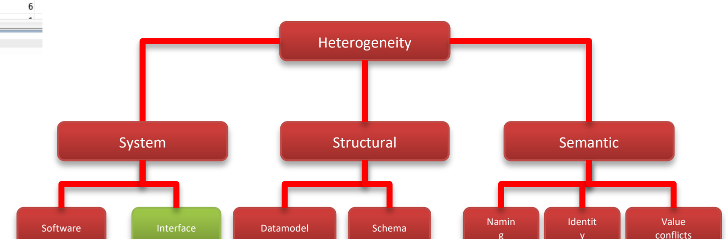
- Interface Heterogeneity – Examples
  - SQL



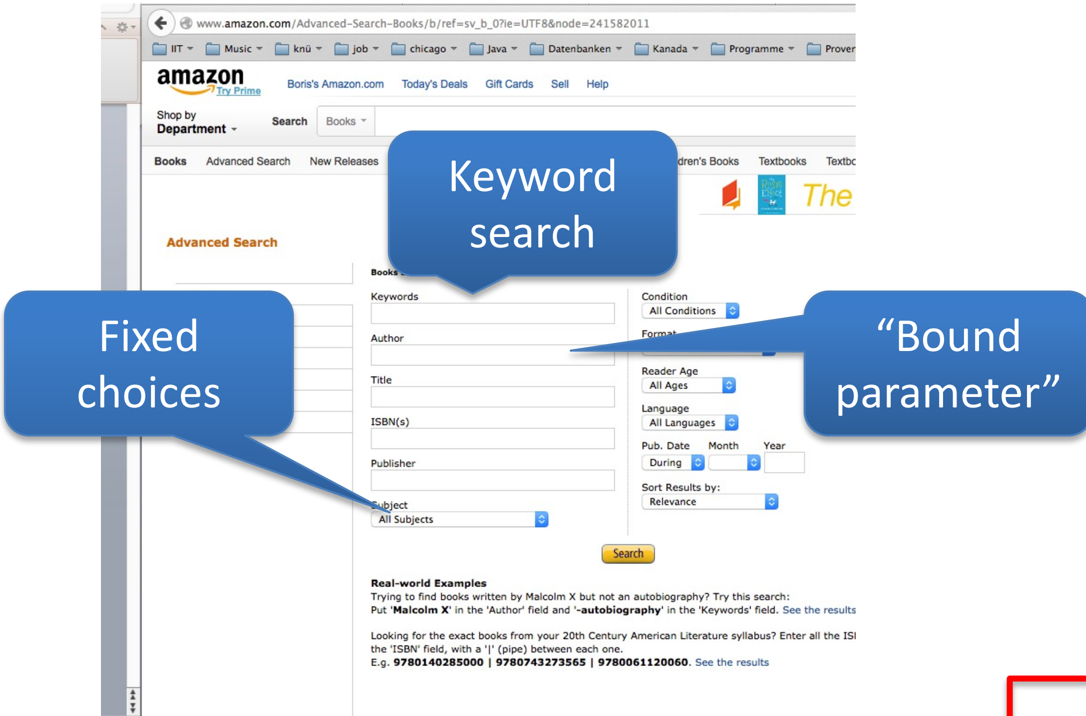


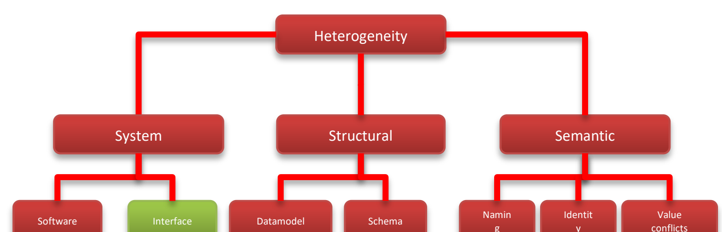
- Interface Heterogeneity – Examples
  - SQL





- Interface Heterogeneity – Examples
  - Web-form (with DB backend?)



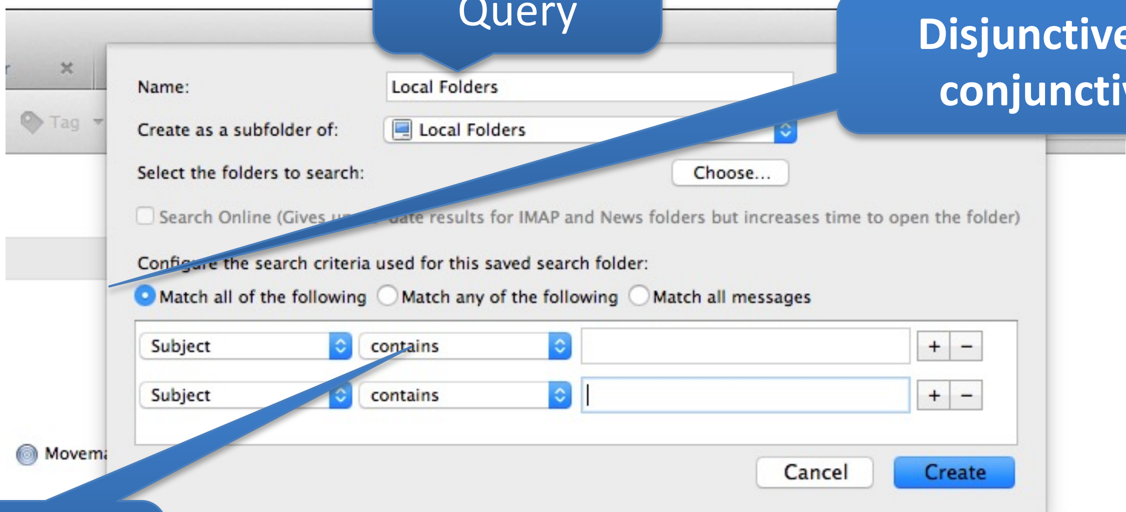


ILLINOIS INSTITUTE OF TECHNOLOGY

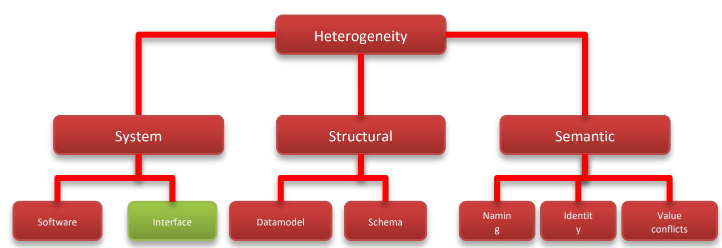
- Interface Heterogeneity – Examples
  - Email-client

Name Query

Disjunctive or conjunctive



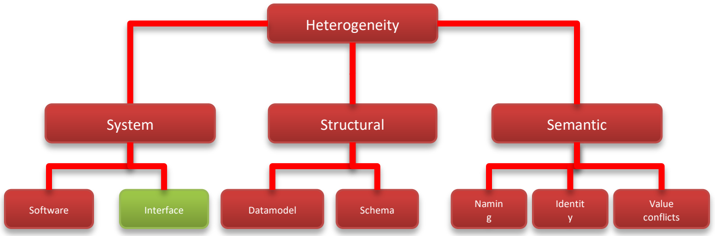
Comparison operator



1.1 System Heterogeneity

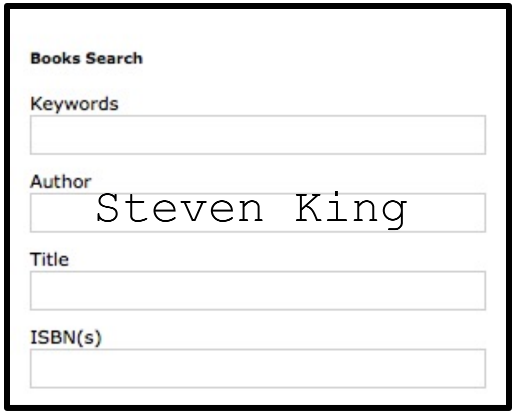
ILLINOIS INSTITUTE OF TECHNOLOGY

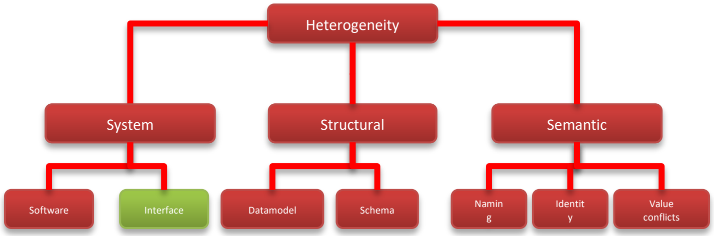
- Problems with interface heterogeneity
  - Global query language is more powerful
    - User queries may not be executable
    - Integration system has to evaluate part of the query
  - Bound parameters are incompatible with query
    - User query may not be executable



1.1 System Heterogeneity

- Example: more expressive global language
  - SQL with one table
    - books (title, author, year, isbn, genre)
  - Web form for books about history shown below
  - What problems do may arise translating user queries?





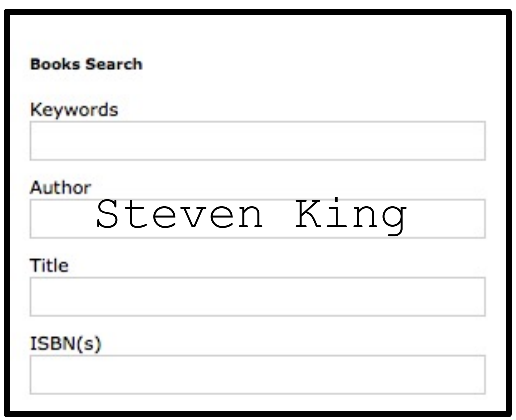
1.1 System Heterogeneity

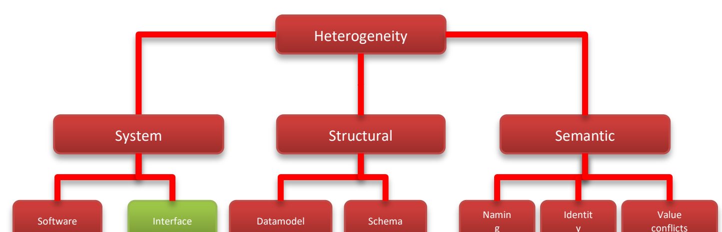
- Integration system has to process part of the query

```
SELECT title
FROM books
WHERE author = 'Steven King'
      AND year = 2012;
```

Stephen King, 2012, Misery

Stephen King, 2012, Misery
Stephen King, 2014, …
Stephen Kine, 1990, …

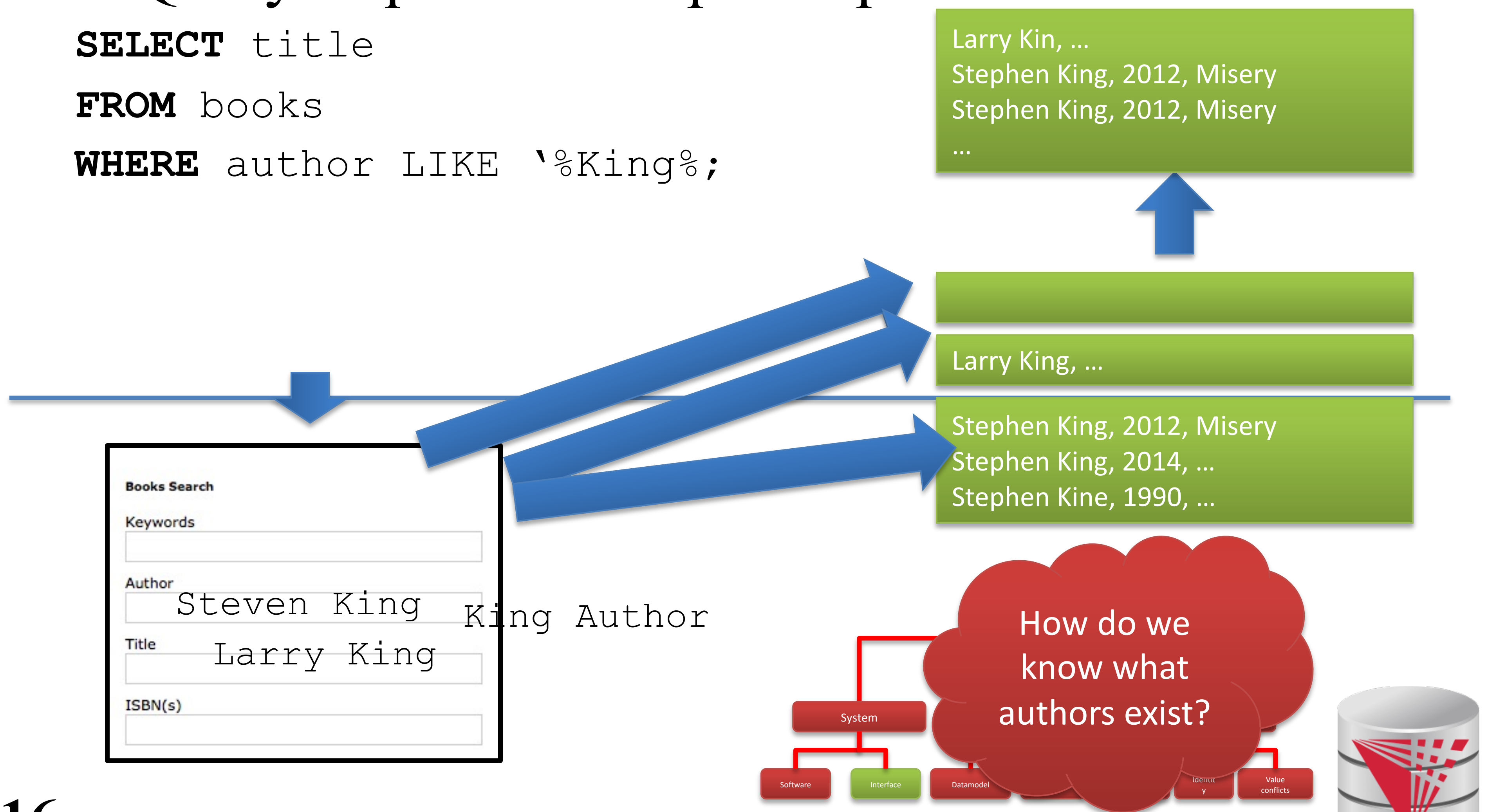




1.1 System Heterogeneity

- Query requires multiple requests

```
SELECT title
FROM books
WHERE author LIKE '%King%;
```

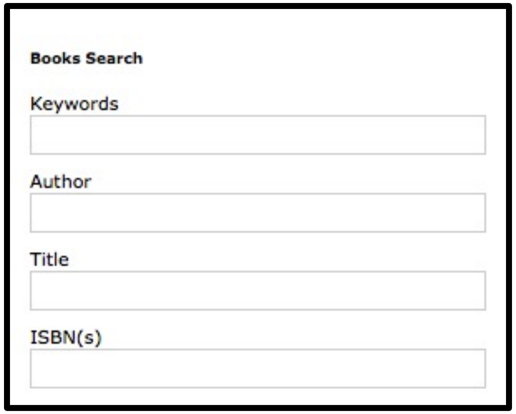


1.1 System Heterogeneity

- Query cannot be answered

```
SELECT title
FROM books
WHERE genre = 'SciFi';
```

Web form is for history book only!

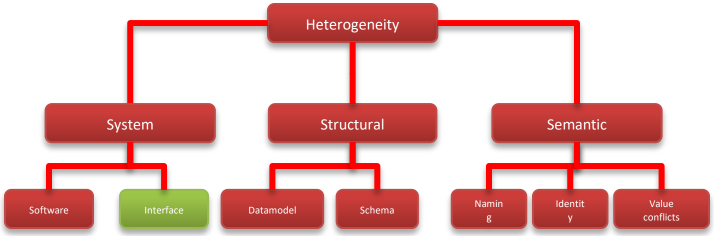

Books Search

Keywords

Author

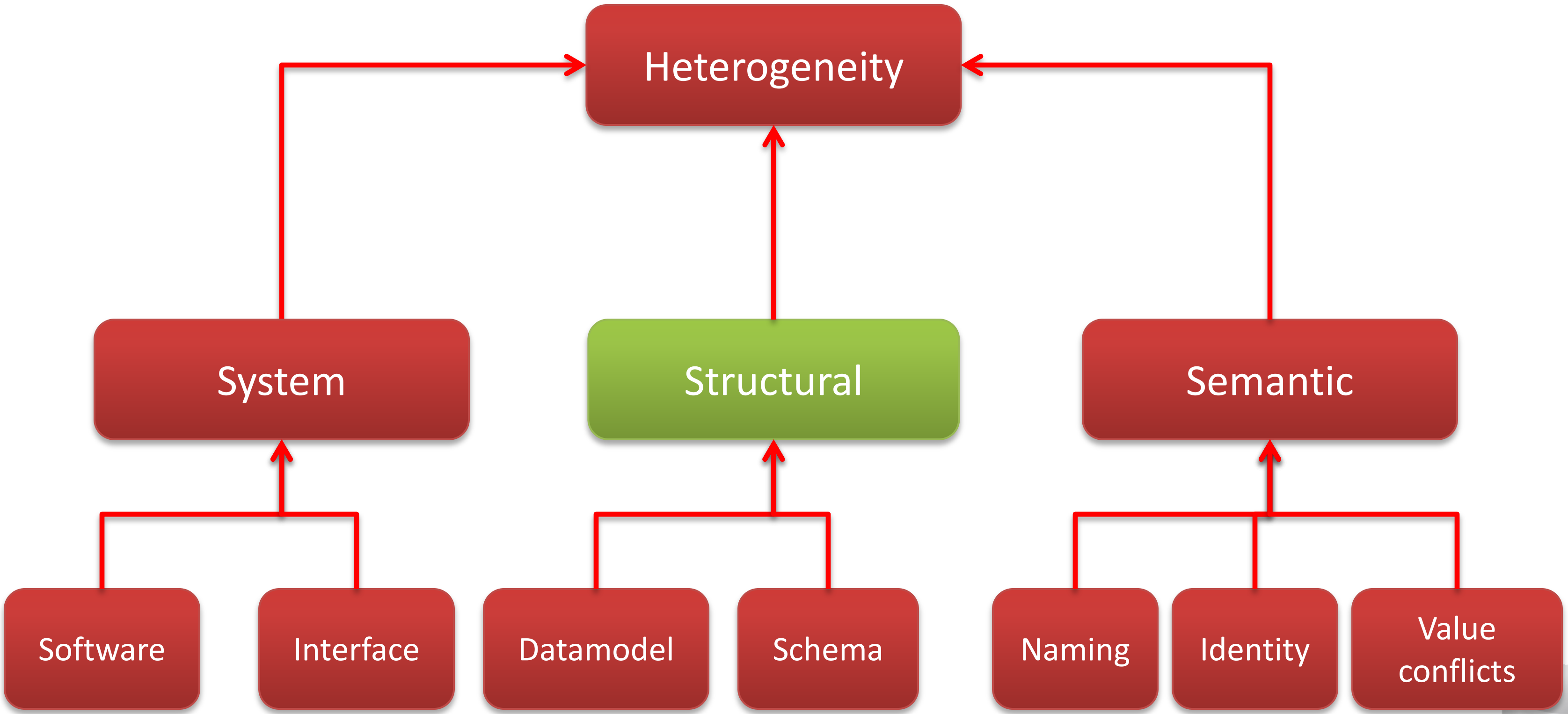
Title

ISBN(s)



1.1 Heterogeneity +Autonomy

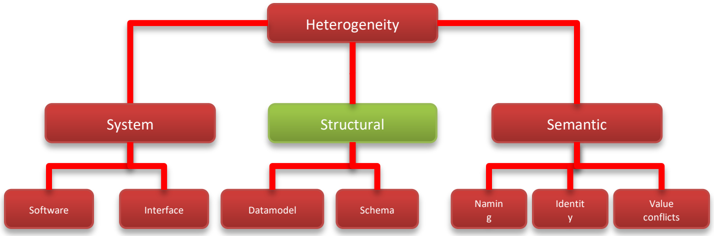
- Taxonomy of Heterogeneity



1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

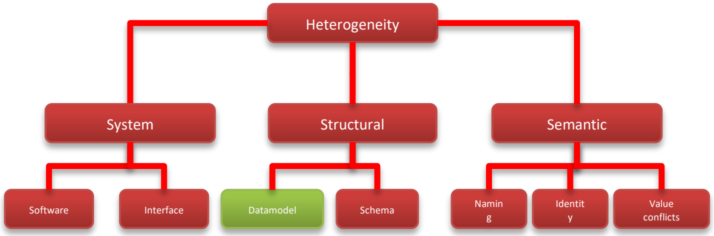
- **Data model**
  - Different semantic/expressiveness
  - Different structure
- **Schema**
  - **Integrity constraints, keys**
  - **Schema elements**:
    - use attribute or separate relations)
  - **Structure**:
    - e.g., normalized vs. denormalized relational schema



1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

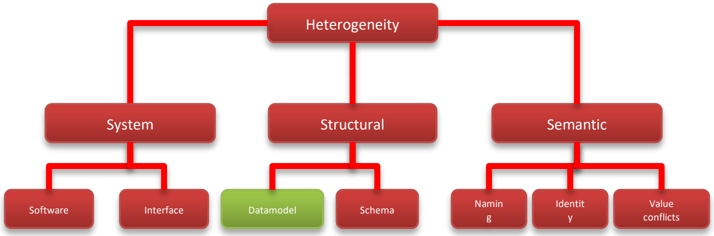
- **Data model**
  - Relational model
  - XML model
  - Object-oriented model
  - Ontological model
  - JSON
  - …



1.1 Structural Heterogeneity

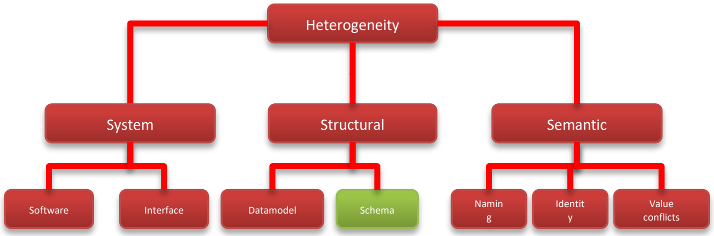
- Example: data model
  - Relational model
  - XML model
  - JSON
  - OO

- Person and their addresses



1.1 Structural Heterogeneity

- **Schema**
  - – Modeling choices
    - Relation vs. attribute
    - Attribute vs. value
    - Relation vs. value
  - – Naming
  - – Normalized vs. denormalized (relational concept)
  - – Nesting vs. reference



1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Modeling choices

```
Male(Id, firstname, lastname)
Female(id, firstname, lastname)
```

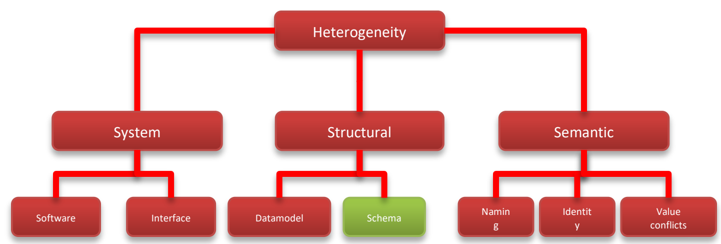
Relation vs. Attribute

Relation vs. Value

```
Person(Id, firstname, lastname, male, female)
```

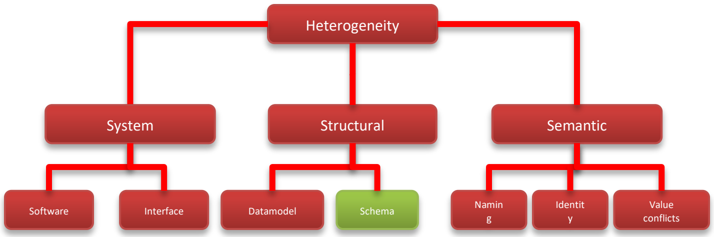
```
Person(Id, firstname, lastname, gender)
```

Value vs. Attribute



1.1 Structural Heterogeneity

- **Relation-relation conflicts**
  - Naming conflicts
    - Relations with different name representing the same data (**synonym**)
    - Relations with same name representing different information (**homonym**)
  - Structural conflicts
    - Missing attributes
    - Many-to-one
    - Missing, but derivable attributes
  - Integrity constraint conflicts



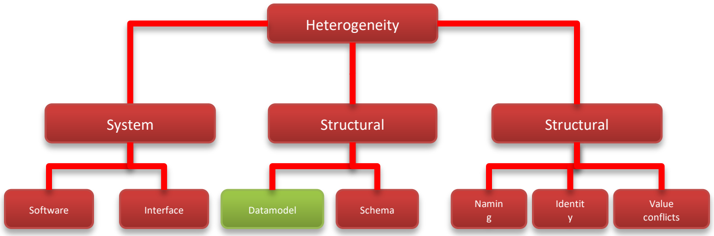
1.1 Structural Heterogeneity

Example: Conflicts between relations

Person(Id, firstname, lastname, male, female)

Person(Id, name, gender, birthday)

Manager(Id, name, gender, age)



1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

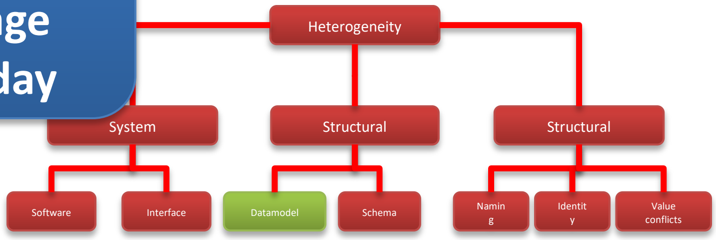
Mutliple attribtue vs one attribute

Example: Conflict between relations

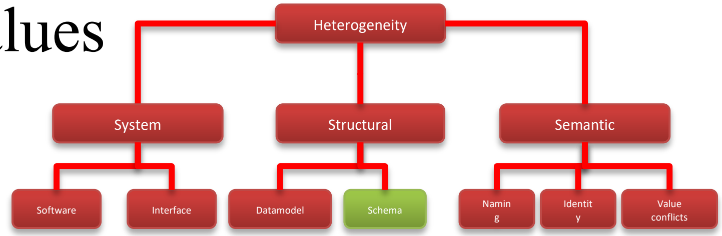
```
Person(Id, firstname, lastname, male, female)

Person(Id, name, gender, birthday)

Manager(Id, name, gender, age)
```

Missing derivable attribute: Role

Derivable attribute: Compute age from birthday



- **Attribute-attribute conflicts**
  - Naming conflicts
    - Attributes with different name representing the same data (**synonym**)
    - Attributes with same name representing different information (**homonym**)
  - Default value conflict
  - Integrity constraint conflicts
    - Datatype
    - Constraints restricting values

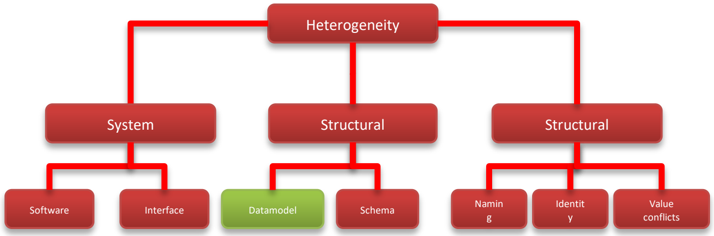


1.1 Structural Heterogeneity

Example: Conflicts between attributes and attributes

SSN	FirstName VARCHAR(40)	LastName	Age CHECK(Age > 18)
333-333-3333	Peter	Schmeter	30
333-333-9999	Hans	Glanz	NULL

SSN	FirstName VARCHAR(25)	SurName	Age
3333333333	Peter	Schmeter	30
3333339999	Hans	Glanz	-1



1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Conflicts between attributes and attributes

SSN	FirstName VARCHAR(40)	LastName	Age CHECK(Age > 18)
333-333-3333	Peter	Schmeter	30
333-333-9999	Hans	Glanz	NULL

SSN	FirstName VARCHAR(25)	SurName	Age
3333333333	Peter	Schmeter	30
3333339999	Hans	Glanz	-1

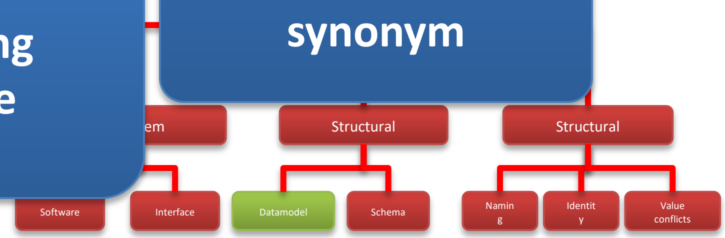
Conflicting constraint

Conflicting default value

Conflicting format

Conflicting datatype

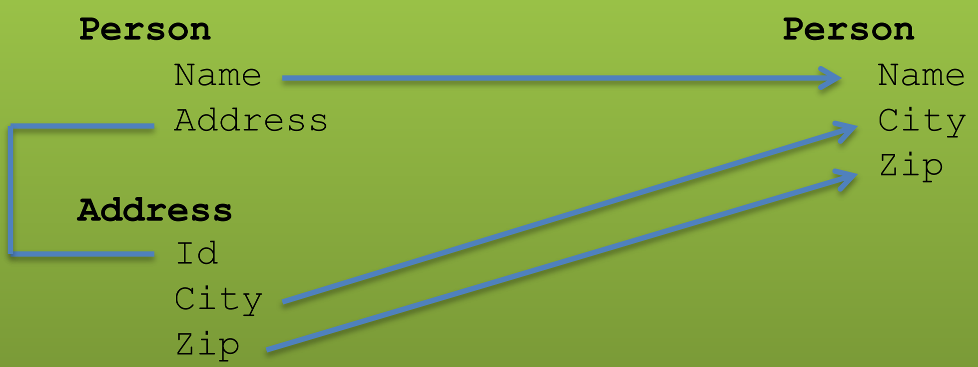
synonym

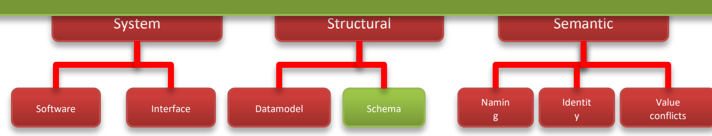


1.1 Structural Heterogeneity

- **Normalized vs. denormalized**
  - E.g., relational model: Association between entities can be represented using multiple relations and foreign keys or one relation

Example

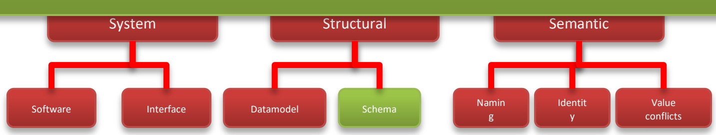




1.1 Structural Heterogeneity

- **Nested vs. flat**
  - Association between entities can be represented using nesting or references (previous slides)

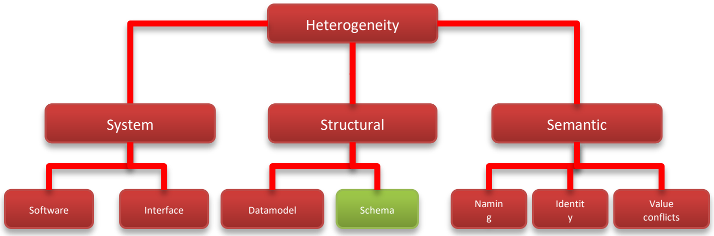
Example



1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

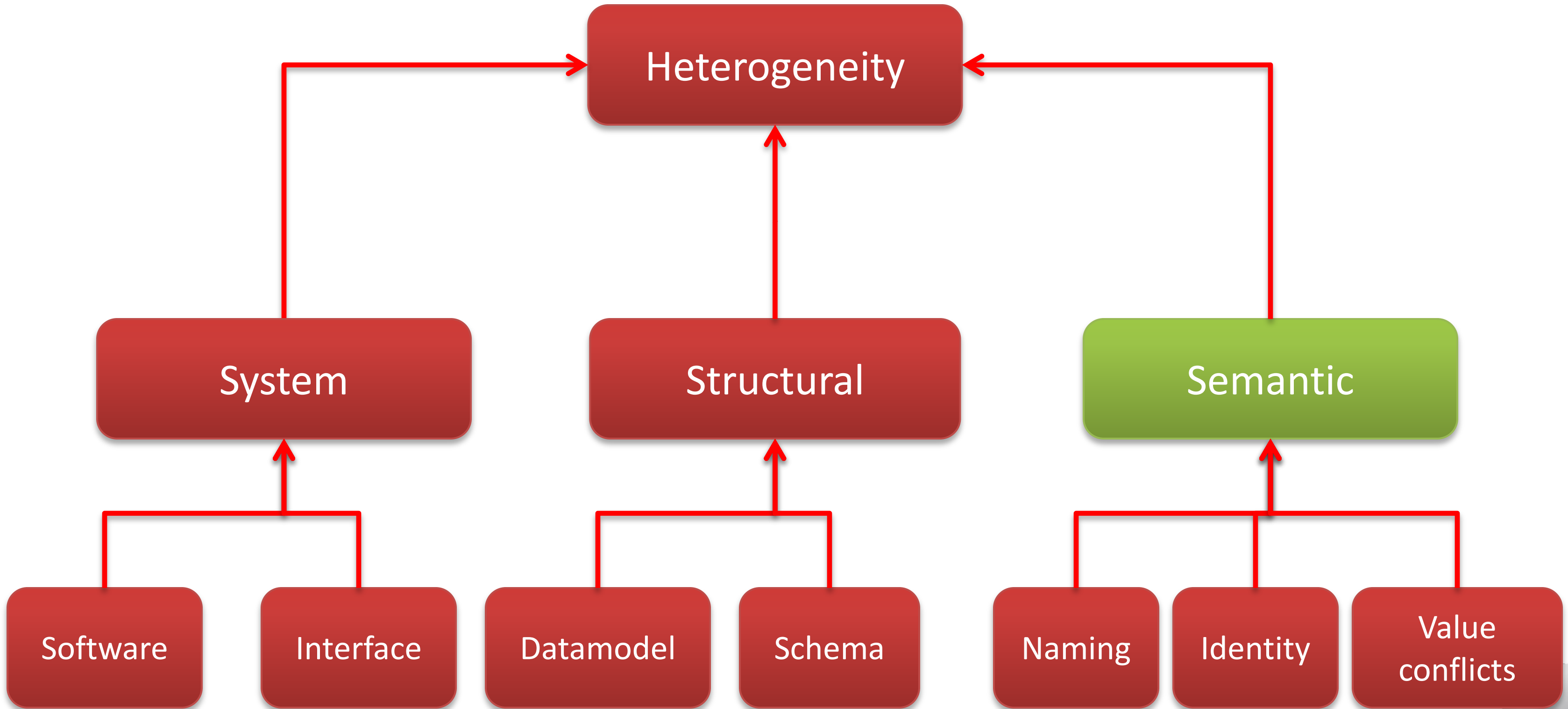
- **Problems caused by schema heterogeneity**
  - Unified access to multiple schemas or integrate schemas into new schema
    - **Schema level**: schema mapping, model management operators, schema languages
    - **Data Level:** virtual data integration, data exchange, warehousing (ETL)



1.1 Heterogeneity +Autonomy

ILLINOIS INSTITUTE OF TECHNOLOGY

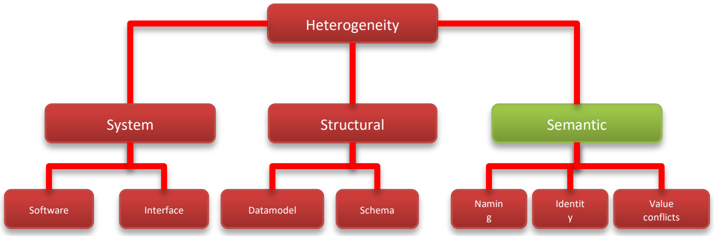
- Taxonomy of Heterogeneity



1.1 Semantic Heterogeneity

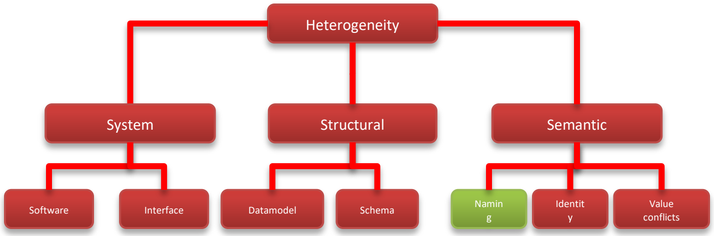
ILLINOIS INSTITUTE OF TECHNOLOGY

- **Semantic Heterogeneity**
  - Naming Conflicts
  - Identity Conflicts (Entity resolution)
  - Value Conflicts (Data Fusion)



1.1 Semantic Heterogeneity

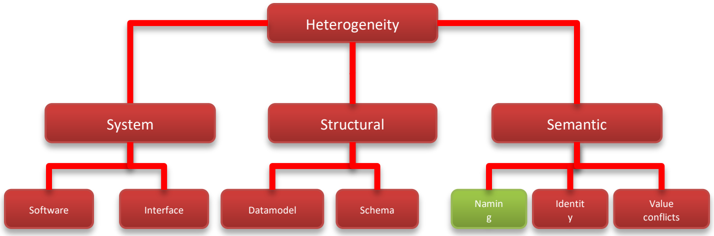
- **Naming Conflicts**
  - Ontological (concepts)
    - Birds vs. Animals
  - Synonyms
    - Surname vs. last name
  - Homonyms
  - Units
    - Gallon vs. liter
  - Values
    - Manager vs. Boss



1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

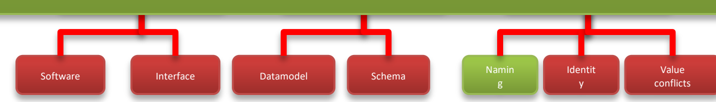
- **Ontological concepts**
  - Relationships between concepts
    - **A = B** - Equivalence
    - **A ⊆B** - Inclusion
    - **A ∩ B** - Overlap
    - **A ≠ B** - Disjunction



- **Ontological concepts**
  – Relationships between concepts
    - **A = B** - Equivalence
    - **A ⊆B** - Inclusion
    - **A ∩ B** - Overlap
    - **A ≠ B** - Disjunction

Example

Equivalence: Human vs Homo sapiens
Inclusion: Bird vs Animal
Overlap: Animal vs aquatic lifeform
Disjunction: Fish vs Mamal



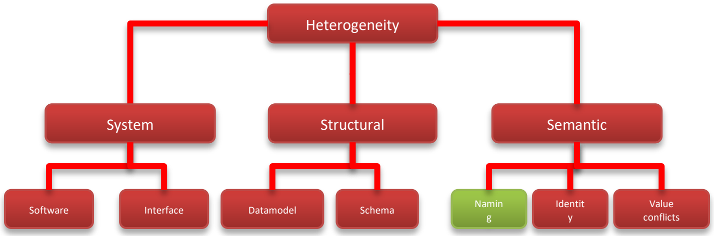
ILLINOIS INSTITUTE OF TECHNOLOGY

- **Naming concepts (synonyms)**
  - **Different** words with **same meaning**

Example

Person(Name, Age)

Human(LastName, Age)



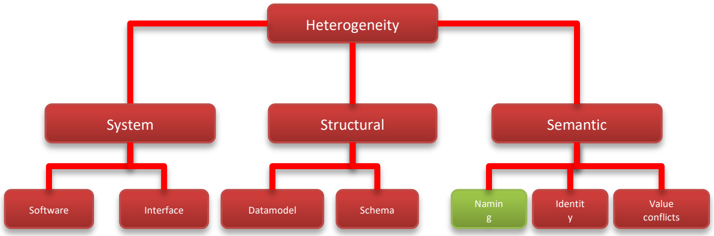
1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Naming concepts (homonyms)**
  - **Same** words with **different meaning**

Example

```
Person (Title, Name)

Movie (Title, Year)
```



1.1 Semantic Heterogeneity

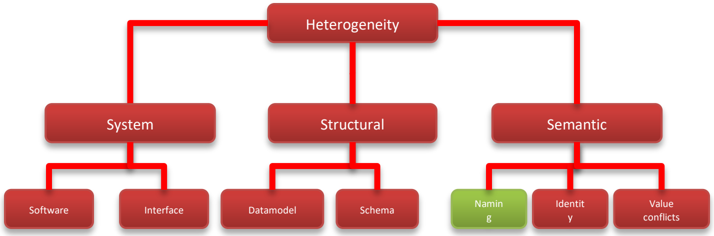
ILLINOIS INSTITUTE OF TECHNOLOGY

- **Naming concepts (units)**

Example

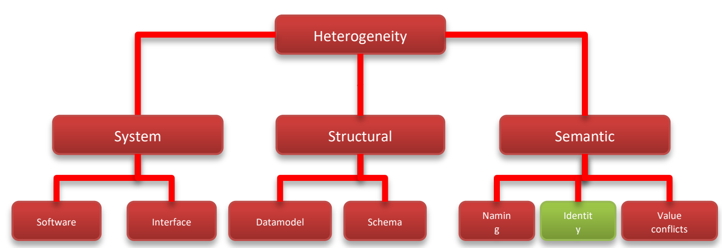
Person(Title,Name,Salary) → $

Person(Title,Name,Salary) → CAD



1.1 Semantic Heterogeneity

- Identity Conflicts
  - What is an object?
    - E.g., multiple tuples in relational model
  - Central question:
    - Does object A represent the same entity as B
  - This problem has been called
    - **Entity resolution**
    - **Record linkage**
    - **Deduplication**
    - …



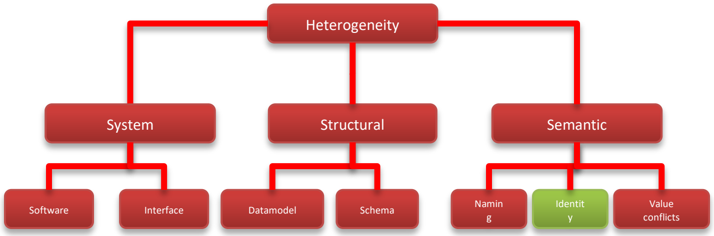
1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- Identity Conflicts

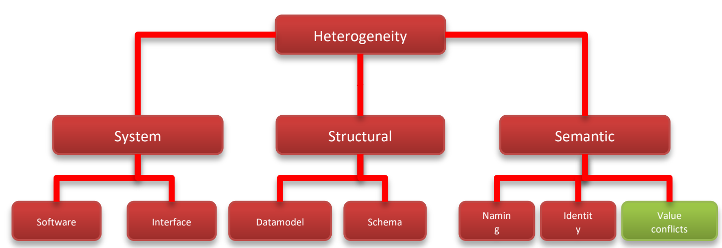
Example

```
(IBM,300000000,USA)

(International Business Machines Corporation,50000)
```



1.1 Semantic Heterogeneity

- **Value Conflicts**
  - Objects representing the same entities have conflicting values for semantically equivalent attributes
    - We have to identified that these objects are represent the same entitity first!
  - Resolving such conflicts requires **Data Fusion**
    - Pick value from conflicting values
    - Numerical methods: e.g., average
    - Preferred value
    - …



1.1 Autonomy

- **How autonomous are data sources**
  - One company
    - Can enforce, e.g., schema and software
  - …
  - The web
    - Website decides
      - Interface
      - Determines access restrictions and limits
      - Availability
      - Format
      - Query restrictions
      - …

1.2 Data integration tasks

- **Cleaning and prepreparation**
- **Entity resolution**
- **Data Fusion**
- **Schema matching**
- **Schema mapping**
- **Query rewrite**
- **Data translation**

1.3 Data integration architectures

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Virtual data integration**
- **Data Exchange**
- **Peer-to-peer data integration**
- **Datawarehousing**
- **Big Data analytics**

1.4 Formal Background

- **Query Equivalence**
  - Complexity for different query classes
- **Query Containment**
  - Complexity for different query classes
- **Datalog**
  - Recursion + Negation
- **Integrity Constraints**
  - Logical encoding of integrity constraints
- **Similarity Measures/Metrics**

1.4 Boolean Logic

- **Boolean Logic (syntax)**
  - **Atomic formulas:**
    - Boolean constants (**true, false)**
    - Boolean Variables (can take Boolean constants as values)
  - **Formulas:**
    - Any atomic formula is also a formula
    - If ϕ, ψ are formulas then the following are also valid formulas:

$\neg \phi$ $\qquad$ $\phi \wedge \psi$

$\phi \vee \psi$ $\qquad$ $\phi \rightarrow \psi$

1.4 Boolean Logic

- **Boolean Logic (semantics)**
  - **Valuation:**
    - Assign truth values to the variables of a formula
    - Under a valuation a formula evaluates to a Boolean value (true or false)
    - If there exists a valuation that makes the formula ψ true then the formula ψ is called **satisfiable**
  - **Semantics:**
    - Expected semantics of Boolean operators:

$$\top \wedge \bot = \bot$$
$$\top \wedge \top = \top$$
$$\bot \vee \top = \top$$
$$\ldots$$

1.4 Boolean Logic

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Formula:

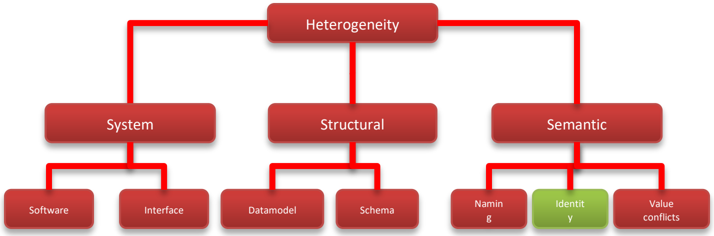
$$(x \vee y) \wedge \neg z$$

A possible valuation:

$$\nu : x = \top, y = \bot, z = \top$$

Evaluating the formula:

$$(\top \vee \bot) \wedge \neg \top = \top \wedge \bot = \bot$$



1.4 First-order logic (FO)

– Concepts

- **Domain of discourse** $\mathbb{D}$
  - These are the values that we can bind variables to
  - Values from the domain can also be used as constants in formulas
- **A set of predicate symbols (each with an arity)**

$$R_1, \ldots, R_n$$

  - These represent relations (in the mathematical sense)
- **An infinite set of variables** $\mathcal{X}$

1.4 FO Syntax

- **Terms**
  - **Variables:** any variable from $\mathcal{X}$ is a term
  - **Constants:** any constant from $\mathbb{D}$ is a term
- **Atomic formulas:**
  - For any n-ary predicate *R* and terms $t_1, \ldots, t_n$ $R(t_1, \ldots, t_n)$ is an atomic formula
- **Formulas:**
  - If ϕ, ψ are formulas then the following are also valid formulas:

$\psi \wedge \phi$ $\psi \vee \phi$ $\neg \phi$

$\psi \rightarrow \phi$ $\exists x : \psi$ $\forall x : \psi$

1.4 Free / Bound Variables

– Free variables of a formula

- All variables not bound by quantifiers

$$free(\neg\psi) = free(\psi)$$
$$free(\psi \wedge \phi) = free(\psi) \cup free(\phi)$$
$$free(\psi \vee \phi) = free(\psi) \cup free(\phi)$$
$$free(\forall x : \psi) = free(\psi) - \{x\}$$
$$free(\exists x : \psi) = free(\psi) - \{x\}$$
$$free(R(t_1, \ldots, t_n)) = free(t_1) \cup \ldots \cup free(t_n)$$
$$free(x) = \{x\}$$
$$free(c) = \emptyset$$

1.4 FO Semantics

- **Model** $\mathcal{M}$
  - an interpretation of the predicates, i.e., we assign each predicate to a concrete relation
  - We select a domain of discourse $\mathbb{D}$
- **Valuations** μ **for a formula** ψ
  - Assigns free variables of ψ to values from $\mathbb{D}$
- **Substitutions**
  - Replace all free occurrences of variable x with c

$$\psi[x \leftarrow c]$$

1.4 FO Semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Given a model $\mathcal{M}$ and valuation μ**
  - The "result" of a formula $[\![\psi]\!]_{\mathcal{M},\mu}$

$$[\![c]\!]_{\mathcal{M},\mu} = c$$

$$[\![x]\!]_{\mathcal{M},\mu} = \mu(x)$$

$$[\![R(t_1,\ldots,t_n)]\!]_{\mathcal{M},\mu} = \begin{cases} \top & \text{if } ([\![t_1]\!]_{\mathcal{M},\mu},\ldots,[\![t_n]\!]_{\mathcal{M},\mu}) \in R \\ \bot & \text{otherwise} \end{cases}$$

$$[\![\psi \wedge \phi]\!]_{\mathcal{M},\mu} = [\![\psi]\!]_{\mathcal{M},\mu} \wedge [\![\phi]\!]_{\mathcal{M},\mu}$$

$$[\![\psi \vee \phi]\!]_{\mathcal{M},\mu} = [\![\psi]\!]_{\mathcal{M},\mu} \vee [\![\phi]\!]_{\mathcal{M},\mu}$$

$$[\![\neg\,\psi]\!]_{\mathcal{M},\mu} = \neg\,[\![\psi]\!]_{\mathcal{M},\mu}$$

$$[\![\exists x : \psi]\!]_{\mathcal{M},\mu} = \bigvee_{c \in \mathbb{D}} : [\![\psi[x \leftarrow c]]\!]_{\mathcal{M},\mu}$$

$$[\![\forall x : \psi]\!]_{\mathcal{M},\mu} = \bigwedge_{c \in \mathbb{D}} : [\![\psi[x \leftarrow c]]\!]_{\mathcal{M},\mu}$$

Example

Formula: $\psi = \forall y : R(x, y)$

Model: $\mathcal{M} = \{R = \{(1,1), (1,2), (1,3)\}$

$\mathbb{D} = \{1, 2, 3\}\}$

Valuation: $\mu(x) = 1$

Result:

$$
\begin{aligned}
&[\![\forall y : R(x, y)]\!]_{\mathcal{M},\mu} \\
=&[\![R(x, 1)]\!]_{\mathcal{M},\mu} \wedge [\![R(x, 2)]\!]_{\mathcal{M},\mu} \wedge [\![R(x, 3)]\!]_{\mathcal{M},\mu} \\
=&[\![(x, 1)]\!]_{\mathcal{M},\mu} \in R \wedge [\![(x, 2)]\!]_{\mathcal{M},\mu} \in R \wedge [\![(x, 3)]\!]_{\mathcal{M},\mu} \in R \\
=&(\mu(x), 1) \in R \wedge (\mu(x), 2) \in R \wedge (\mu(x), 3) \in R \\
=&(1, 1) \in R \wedge (1, 2) \in R \wedge (1, 3) \in R \\
=&\top \wedge \top \wedge \top \\
=&\top
\end{aligned}
$$

1.4 FO Problems

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Model checking**
  - Given a model $\mathcal{M}$ and formula ψ without free variables
  - Is $[\![\psi]\!]_{\mathcal{M},\mu}$ true?
- **Satisfiability**
  - Given a formula ψ does there exist a model $\mathcal{M}$ and valuation μ such that $[\![\psi]\!]_{\mathcal{M},\mu}$ is true?

1.4 Integrity constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

- **You know some types of integrity constraints already**
  - **Functional dependencies**
    - **Keys are a special case**
  - **Foreign keys**
    - **We have not really formalized that**

1.4 Integrity constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

- Other types are
  - Conditional functional dependencies
    - **E.g., used in cleaning**
  - Equality-generating dependencies
  - Multi-valued dependencies
  - Tuple-generating dependencies
  - Join dependencies
  - Denial constraints
  - …

1.4 Integrity constraints

- How to manage all these different types of constraints?
  - Has been shown that these constraints can be expressed in a logical formalism.
  - Formulas which consist of relational and comparison atoms. Variables represent values
    - R(x,y,z)
    - x = y

1.4 Integrity Constraints

Example

Primary Key R(<u>A</u>,B):

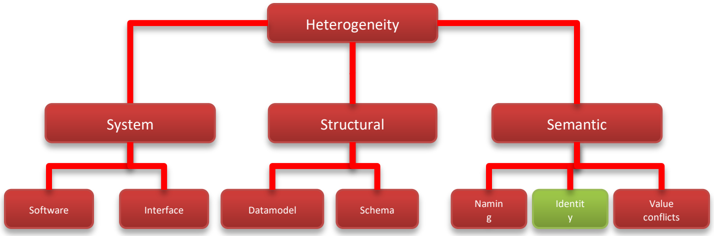
$$\forall x, y, z : R(x,y) \wedge R(x,z) \rightarrow y = z$$

Functional Dependency R(A,B) with A->B:

$$\forall x, y, z, a : R(x,y) \wedge R(z,a) \wedge x = z \rightarrow y = a$$

Foreign Key R(<u>A</u>,B), S(C,D) where D is FK to R:

$$\forall x, y : S(x,y) \rightarrow \exists z : R(y,z)$$



1.4 Integrity constraints

- Types of constraints we will use a lot
  - Tuple-generating dependencies (**tgds**)
    - Implication with conjunction of relational atoms
    - Foreign keys and schema mappings (later)

$$\forall \vec{x} : \phi(\vec{x}) \rightarrow \exists \vec{y} : \psi(\vec{x}, \vec{y})$$

  - Equality-generating dependencies (**egds**)
    - Generalizes keys, FDs

$$\forall \vec{x} : \phi(\vec{x}) \rightarrow \wedge_{k=1}^{n} x_{i_k} = x_{j_k}$$

1.4 Datalog

- What is Datalog?
  - Prolog for databases (syntax very similar)
  - A logic-based query language
- Queries (Program) expressed as set of rules

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

- One Q is specified as the answer relation (the relation returned by the query)

1.4 Datalog - Intuition

ILLINOIS INSTITUTE OF TECHNOLOGY

- **A Datalog rule**

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

- **Procedural Interpretation:** For all bindings of variables that makes the RHS true (conjunction) return bindings of $\vec{x}$

Example

```
Q(Name):- Person(Name,Age).
Return names of persons
```

1.4 Datalog - Syntax

- A **Datalog program** is a set of Datalog rules
  - Optionally a distinguished answer predicate
- A **Datalog rule** is

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

- **X's** are lists of variables and constants
- **Ri's** are relation names
- **Q** is a relation name

1.4 Datalog - Terminology

- Left-hand side of a rule is called it's **head**
- Right-hand side of a rule is called it's **body**
- Relation are called **predicates**
- $R(\vec{x})$ is called an **atom**
- An **instance** I of a database is the data
- The **active domain** adom(I) of an instance I is the set of all constants that occur in I

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

1.4 Datalog - Terminology

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

```
Q(N) :- Person(N,A).
```

N, A are **variables**

Q(N), Person(N,A) are **atoms**

Person and Q are **predicates**

Name	Age
peter	34
bob	45

Activate domain

adom(I) = {peter,bob,34,45}

1.4 Datalog - Terminology

- **Intensional** vs. **extensional**
  - Extensional database (**edb**)
    - What we usually call database
  - Intensional database (**idb**)
    - Relations that occur in the head of rules (are populated by the query)
  - Usually we assume that these do not overlap

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

1.4 Datalog - Safety

- A Datalog program is safe if all its rules are **safe**
- A rule is **safe** if all variables in $\vec{x}$ occur in at least one $\vec{x_i}$

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

Example

```
Q(Name):- Person(Name,Age).   (safe)
Q(Name,Sal):-Peron(Name,Age).(unsafe)
```

1.4 Datalog - Semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

- The instance of an idb predicate Q in a datalog program for an edb instance I contains all facts that can be derived by applying rules with Q in the head
- A rule derives a fact Q(c) if we can find a binding of variables of the rule to constants from adom(I) such that x is bound to c and the body is true

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

1.4 Datalog - Semantics

Example

```
Q(N) :- Person(N,A).
```

N
peter
bob

```
N=peter,A=peter: Q(peter):- Person(peter,peter).
N=peter,A=bob: Q(peter):- Person(peter,bob).
```
`N=peter,A=34: Q(peter):- Person(peter,34).`
```
N=bob,A=peter: Q(bob):- Person(peter,peter).
N=bob,A=bob: Q(bob):- Person(peter,bob).
```
`N=bob,A=34: Q(bob):- Person(bob,34).`
```
N=34,A=peter: Q(34):- Person(34,peter).
N=34,A=bob: Q(34):- Person(34,bob).
N=34,A=34: Q(34):- Person(34,34).
```

Active domain

adom(I) = {peter,bob,34}

Name	Age
peter	34
bob	34

1.4 Datalog

- Different flavors of datalog
  - **Conjunctive query**
    - Only one rule
    - Expressible as Select-project-join (SPJ) query in relational algebra (only equality and AND in selection)
  - **Union of conjunctive queries**
    - Also allow union
    - SPJ + set union in relational algebra
    - Rules with the same head in Datalog
  - **Conjunctive queries with inequalities**
    - Also allow inequivalities, e.g., $<$

1.4 Datalog

- Different flavors of datalog
  - **Recursion**
    - Rules may have recursion:
      - E.g., head predicate in the body
    - Fixpoint semantics based on immediate consequence operator
  - **Negation (first-order queries)**
    - Negated relational atoms allowed
    - Require that every variable used in a negated atom also occurs in at least on positive atom (**safety**)
  - **Combined Negation + recursion**
    - Stronger requirements (e.g., stratification)

1.4 Datalog – Semantics (Negation)

ILLINOIS INSTITUTE OF TECHNOLOGY

- A rule derives a fact Q(c) if we can find a binding of variables of the rule to constants from adom(I) such that x is bound to c and the body is true
- A negated atom not R(X) is true if R(X) is not part of the instance

$$Q(\vec{x}) :- R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}).$$

1.4 Datalog - Semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

```
Q(N) :- Person(N,A), not Lives(N).
```

```
N=peter,A=peter: Q(peter):- Person(peter,peter),
                              not Lives(peter).
N=peter,A=bob: Q(peter):- Person(peter,bob),
                              not Lives(peter).
...
```

N=bob,A=34: Q(bob):- Person(bob,34),
not Lives(bob).

...

Active domain

adom(I) = {peter,bob,34}

Result

N
bob

Lives

Name
peter

Person

Name	Age
peter	34
bob	34

1.4 Datalog

Example

Relation **hop(A,B)** storing edges of a graph.

$Q_{2hop}(x,z)$: hop(x,y),hop(y,z).

$Q_{reach}(x,y)$: hop(x,y).
$Q_{reach}(x,z)$: $Q_{reach}(x,y)$,$Q_{reach}(y,z)$.

$Q_{node}(x)$: hop(x,y).
$Q_{node}(x)$: hop(y,x).

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Relation **hop(A,B)** storing edges of a graph.

Q_{node}(x): hop(x,y).
Q_{node}(x): hop(y,x).

$Q_{notReach}$(x,y): Q_{node}(x), Q_{node}(y),
not Q_{reach}(x,y).

1.4 Datalog versus FO

ILLINOIS INSTITUTE OF TECHNOLOGY

- A Datalog rule is a FO implication:

$$Q(X, Y) :- R(X, Z), R(Z, Y).$$

Means

$$\forall x, y : \exists z : R(x, z) \wedge R(z, y) \rightarrow Q(x, y)$$

- Databases can be expressed as rules!

$$R = \{(Peter, Bob), (Bob, Alice)\}$$

$$R(Peter, Bob) :-$$

$$R(Bob, Alice) :-$$

1.4 Model-theoretic semantics

- The result of a Datalog program P is the smallest model $\mathcal{M}$ for the program if interpreted as a logical formula
  - Only facts that are justified by the program are included in the query result!

1.4 Free Datalog Systems

- Datalog Education System (DES)
  - http://des.sourceforge.net/
- DLV
  - http://www.dlvsystem.com/dlv/

1.4 Containment and Equivalence

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Query Equivalence

Query Q is equivalent to Q' iff for every database instance I both queries return the same result

$$Q \equiv Q' \Leftrightarrow \forall I : Q(I) = Q'(I)$$

Definition: Query Containment

Query Q is contained in query Q' iff for every database instance I the result of Q is contained in the result of Q'

$$Q \sqsubseteq Q' \Leftrightarrow \forall I : Q(I) \subseteq Q'(I)$$

1.4 Equivalence

- The problem of checking query equivalence is of different complexity depending on the **query language** and whether we consider **set or bag semantics**

1.4 Containment and Equiv.

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Q_1(x,y): R(x,y), R(x,z).
Q_2(x,y): R(x,y).
Q_3(x,x): R(x,x).
Q_4(x,y): R(x,y).
Q_5(x,x): R(x,y), R(x,x).
Q_6(x,z): R(x,y), R(y,z).

1.4 Containment and Equiv.

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Relation **hop(A,B)** storing edges of a graph.

Q_{2hop}(x,z): hop(x,y),hop(x,z).

Q_{up2Hop}(x,z): hop(x,y),hop(x,z).
Q_{up2Hop}(x,z): hop(x,z).

Q_{sym}(x,y): hop(x,y).
Q_{sym}(x,y): hop(y,x).
$Q_{sym2Hop}$(x,y): Q_{sym}(x,y),Q_{sym}(y,z).

1.4 Complexity of Eq. and Cont.

Set semantics	Relational Algebra	Conjunctive Queries (CQ)	Union of Conjunctive Queries (UCQ)	Monotone Queries/ CQ≠
Query Evaluation (Combined Complexity)	PSPACE-complete	NP-complete	NP-complete	NP-complete
Query Evaluation (Data Complexity)	LOGSPACE (that means in P)	LOGSPACE (that means in P)	LOGSPACE (that means in P)	LOGSPACE (that means in P)
Query Equivalence	Undecidable	NP-complete	NP-complete	Π_2^p-complete
Query Containment	Undecidable	NP-complete	NP-complete	Π_2^p-complete

1.4 Complexity of Eq. and Cont.

ILLINOIS INSTITUTE OF TECHNOLOGY

Bag semantics	Relational Algebra	Conjunctive Queries (CQ)	Union of Conjunctive Queries (UCQ)
Query Equivalence	Undecidable	Equivalent to graph isomorphism	Undecidable
Query Containment	Undecidable	**Open Problem**	Undecidable

1.4 Containment Mappings

- NP-completeness for set semantics CQ and UCQ for the containment, evaluation, and equivalence problems is based on reducing these problems to the same problem
  - **[Chandra & Merlin, 1977]**
- Notational Conventions:
  - **head(Q)** = variables in head of query Q
  - **body(Q)** = atoms in body of Q
  - **vars(Q)** = all variable in Q

1.4 Boolean Conjunctive Queries

- A conjunctive query is boolean if the head does not have any variables
  - **Q() :- hop(x,y), hop(y,z)**
  - We will use Q :- … as a convention for Q() :- …
  - What is the result of a Boolean query
    - Empty result {}, e.g., no **hop(x,y), hop(y,z)**
    - If there are tuples matching the body, then a tuple with zero attributes is returned {()}
  - -> We interpret {} as **false** and {()} as **true**
  - Boolean query is essentially an existential check

1.4 Boolean Conjunctive Queries

- BCQ in SQL

Example

```
Hop relation: Hop(A,B)

Q :- hop(x,y)

SELECT EXISTS (SELECT * FROM hop)

Note: in Oracle and DB2 we need a
from clause
```

1.4 Boolean Conjunctive Queries

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

```
SELECT
     CASE WHEN EXISTS (SELECT *
                       FROM hop)
     THEN 1 ELSE 0
     END AS x
FROM dual;

Notes:
- Oracle and DB2 FROM not optional
- Oracle has no boolean datatype
```

- BCQ in SQL

Example

```
Q :- hop(x,y), hop(y,z)

SELECT EXISTS
      (SELECT *
      FROM hop l, hop r
      WHERE l.B = r.A)
```

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

- How to check for containment of CQs (set)

Definition: Variable Mapping

A variable mapping ψ from query Q to query Q' maps the variables of Q to constants or variables from Q'

Definition: Containment Mapping

A containment mapping from query Q to Q' is a variable mapping ψ such that:

$$\Psi(head(Q)) = head(Q')$$

$$\forall R(\vec{x_i}) \in body(Q) : \Psi(R(\vec{x_i})) \in body(Q')$$

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Theorem: Containment Mappings and Query Containment

Query Q is contained in query Q' iff there exists a containment mapping ψ from Q' to Q

$$Q \sqsubseteq Q' \Leftrightarrow \exists \Psi : \Psi \text{ is a containment mapping } Q' \rightarrow Q$$

Example

```
Q1(u,z): R(u,z).
Q2(x,y): R(x,y).
```

Can we find a containment mapping?

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Theorem: Containment Mapping and Query Containment

Query Q is contained in query Q’ iff there exists a containment mapping ψ from Q’ to Q

Example

$Q_1(u,z)$: R(u,z).
$Q_2(x,y)$: R(x,y).

$Q_1 \rightarrow Q_2 : \Psi(u)=x, \Psi(z)=y$
$Q_2 \rightarrow Q_1 : \Psi(x)=u, \Psi(y)=z$

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Q_1(a,b): R(a,b), R(b,c).

Q_2(x,y): R(x,y).

1.4 Containment Mappings

Example

Q_1(a,b): R(a,b), R(b,c).
Q_2(x,y): R(x,y).

Do containment mappings exist?

Q_1 -> Q_2: none exists
Q_2 -> Q_1: Ψ(x)=a, Ψ(y)=b

1.4 Containment Mappings

Example

Q_1(a,b): R(a,b), R(c,b).
Q_2(x,y): R(x,y).

Q_1 -> Q_2 :Ψ(a)=x, Ψ(b)=y, Ψ(c)=x
Q_2 -> Q_1 :Ψ(x)=a, Ψ(y)=b

1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

- It was shown that query evaluation, containment, equivalence as all reducible to homomorphism checking for CQ
  - Canonical conjunctive query Q^I for instance I
    - Interpret attribute values as variables
    - The query is a conjunction of all atoms for the tuples
    - **I = {hop(a,b), hop(b,c)} -> Q^I :- hop(a,b), hop(b,c)**
  - Canonical instance I^Q for query Q
    - Interpret each conjunct as a tuple
    - Interpret variables as constants
    - **Q :- hop(a,a) -> I^Q = {hop(a,a)}**

1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

- Containment Mapping <-> Containment
- Proof idea (boolean queries)
  - (if direction)
    - Assume we have a containment mapping Q_1 to Q_2
    - Consider database D
    - Q_2(D) is true then we can find a mapping from vars(Q_2) to D
    - Compose this with the containment mapping and prove that this is a result for Q_1

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Q_1(): R(a,b), R(c,b).

Q_2(): R(x,y).

Q_2 -> Q_1 :Ψ(x)=a, Ψ(y)=b

D={R(1,1), R(1,2)}

Q_1(D)={(1,1),(1,2)}

φ(a)=1, φ(b)=2, φ(c)=1

Ψφ(x)=1, Ψφ(y)=2

1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

- Containment Mapping <-> Containment
- Proof idea (boolean queries)
  - (only-if direction)
    - Assume Q_2 contained in Q_1
    - Consider canonical (frozen) database I^{Q2}
    - Evaluating Q_1 over I^{Q2} and taking a variable mapping that is produced as a side-effect gives us a containment mapping

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Q_1(): R(a,b), R(c,b).

Q_2(): R(x,y).

Q_2 -> Q_1 :Ψ(x)=a, Ψ(y)=b

I^{Q1} = {(a,b),(c,b)}

$Q_2(I^{Q1})$={()}

φ(x)=a, φ(y)=b

φ is our containment mapping Ψ

1.4 Containment Background

- If you are not scared and want to know more:
  - Look up Chandra and Merlins paper(s)
  - The text book provides a more detailed overview of the proof approach
  - Look at the slides from Phokion Kolaitis excellent lecture on database theory
    - https://classes.soe.ucsc.edu/cmps277/Winter10/

1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

- A more intuitive explanation why containment mappings work
  - Variable naming is irrelevant for query results
  - If there is a containment mapping Q to Q'
    - Then every condition enforced in Q is also enforced by Q'
    - Q' may enforce additional conditions

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Q_1() : **R(a,b)**, R(c,b).

Q_2() : **R(x,y)**.

Q_2 -> Q_1 :Ψ(x)=a, Ψ(y)=b

If there exists tuples

R(a,b) and R(c,b)

in R that make Q_1 true, then we take

R(a,b)

to fulfill Q_2

1.4 Containment Background

- From boolean to general conjunctive queries
  - Instead of returning true or false, return bindings of variables
  - Recall that containment mappings enforce that the head is mapped to the head
  - -> same tuples returned, but again Q’ s condition is more restrictive

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

```
Q1(a): R(a,b), R(c,b).
Q2(x): R(x,y).
Q2 -> Q1 :Ψ(x)=a, Ψ(y)=b
```

For every

R(a,b) and R(c,b)

Q_1 returns (a) and for every

R(a,b)

Q_2 returns (a)

1.4 Similarity Measures

- **Problem faced by multiple integration tasks**
  - Given two objects, how similar are they
  - **E.g., given two attribute names in schema matching, given two values in data fusion/entity resolution, …**

1.4 Similarity Measures

- **Object models**
  - **Multidimensional (feature vector model)**
    - Object is described as a vector of values - one for each dimension out of a given set of dimensions
    - E.g., Dimensions are gender (male/female), age (0-120), and salary (0-1,000,000). An example object is [male,80,70,000]
  - **Strings**
    - E.g., how similar is “Poeter” to “Peter”
  - **Graphs and Trees**
    - E.g., how similar are two XML models

1.4 Similarity Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Similarity Measure

Function d(p,q) where p and q are objects, that returns a real score with

- d(p,p) = 0
- d(p,q) >= 0

- **Interpretation: the lower the score the "more similar" the objects are**
- **We require d(p,p)=0, because nothing can be more similar to an object than itself**
- **Note: often scores are normalized to the range [0,1]**

1.4 Similarity Measures

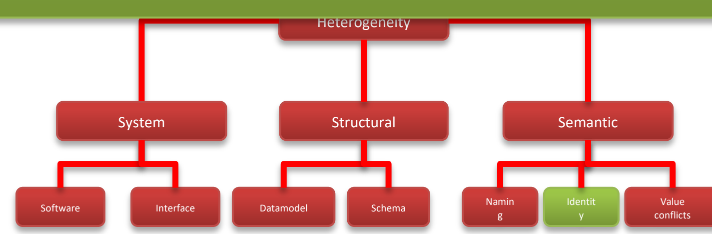
Example

String equality: d(p,q) = 0 if p=q
strings d(p,q) = 1 else

Euclidian distance: d(p,q) =
N-dimensional space

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p[i] - q[i])^2}$$

Edit distance: d(p,q) = minimum number of single character insertions, deletions, replacements to transform p into q
strings



1.4 Similarity Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Metric

Function d(p,q) where p and q are objects, that returns a real score with

- **Non-negative** d(p,q) >= 0
- **Symmetry** d(p,q) = d(q,p)
- **Identity of indiscernibles** d(p,q) = 0 iff p=q
- **Triangle inequality** d(p,q) + d(q,r) >= d(p,r)

1.4 Similarity Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Metric

Function d(p,q) where p and q are objects, that returns a real score with

• **Non-negative**	d(p,q) >= 0
• **Symmetry**	d(p,q) = d(q,p)
• **Identity of indiscernibles**	d(p,q) = 0 iff p=q
• **Triangle inequality**	d(p,q) + d(q,r) >= d(p,r)

1.4 Similarity Measures

- **Why do we care whether d is a metric?**
  - Some data mining algorithms only work for metrics
    - **E.g., some clustering algorithms such as k-means**
    - **E.g., clustering has been used in entity resolution**
  - Metric spaces allow optimizations of some methods
    - **E.g., Nearest Neighboorhood-search: find the most similar object to an object p. This problem can be efficiently solved using index structures that only apply to metric spaces**

Summary

- Heterogeneity
  - Types of heterogeneity
  - Why do they arise?
  - Hint at how to address them
- Autonomy
- Data Integration Tasks
- Data Integration Architectures
- Background
  - Datalog + Query equivalence/containment + Similarity + Integrity constraints

Outline

0) Course Info
1) Introduction
2) **Data Preparation and Cleaning**
3) Schema matching and mapping
4) Virtual Data Integration
5) Data Exchange
6) Data Warehousing
7) Big Data Analytics
8) Data Provenance