

- 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

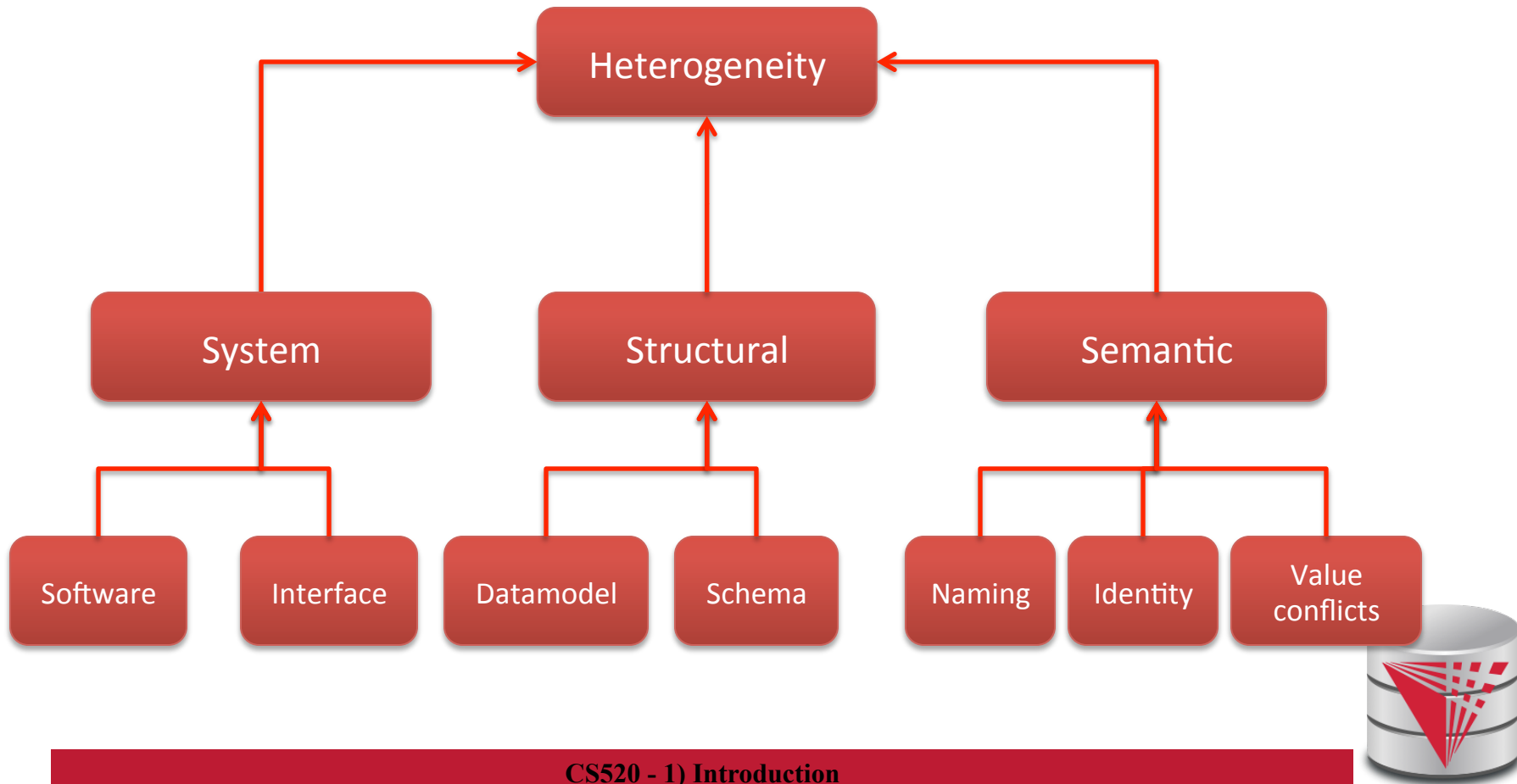


- Topics covered in this part
 - Heterogeneity and Autonomy
 - Data Integration Tasks
 - Data Integration Architectures (Methods)
 - Some Formal Background (sorry!)



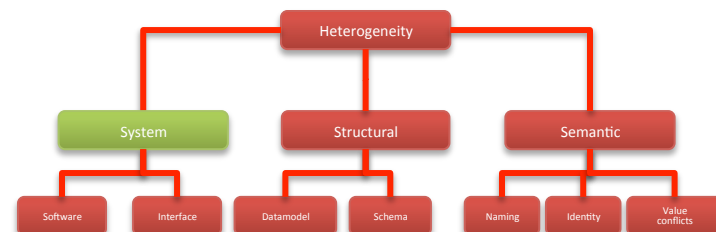
1.1 Heterogeneity + Autonomy

- Taxonomy of Heterogeneity



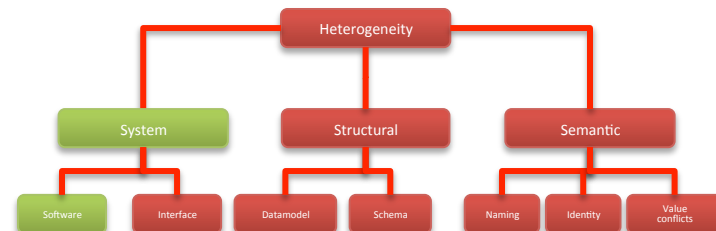
1.1 System Heterogeneity

- 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



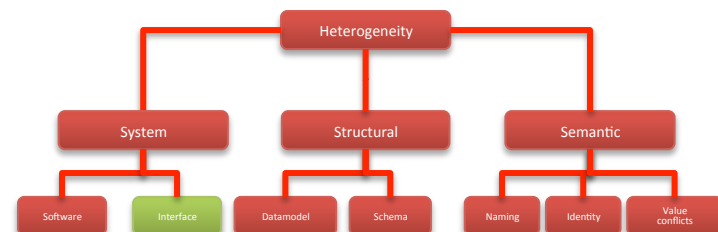
1.1 System Heterogeneity

- 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, ..



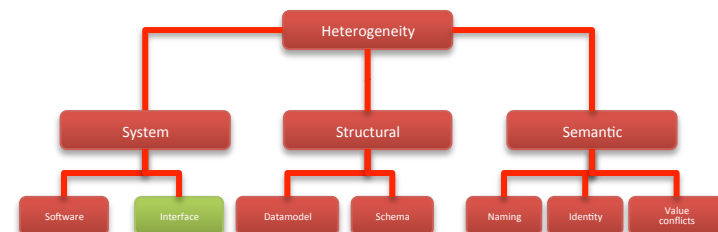
1.1 System Heterogeneity

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



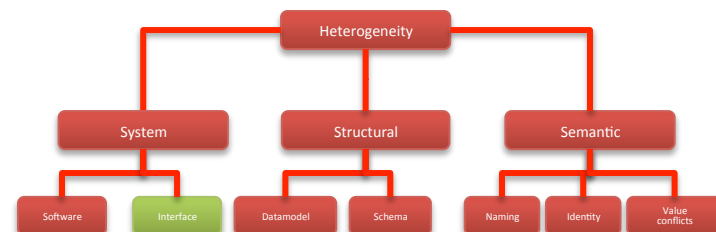
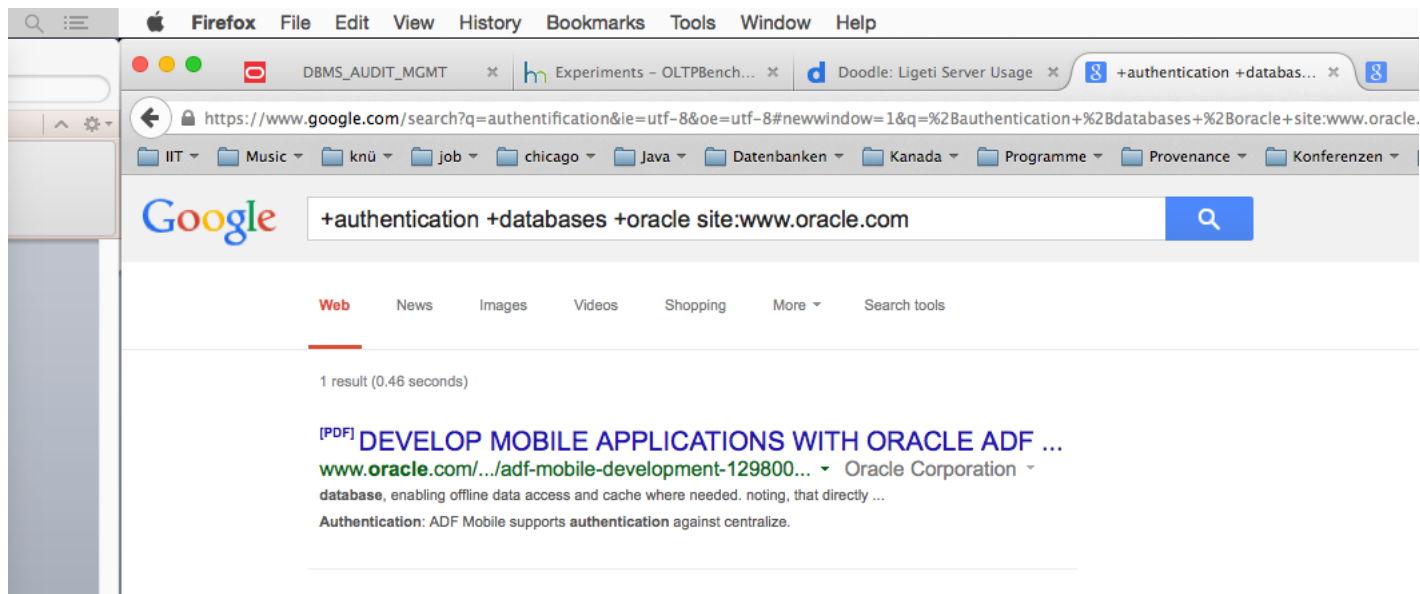
1.1 System Heterogeneity

- Interface Heterogeneity – Expressiveness
 - Keyword-search vs. query language
 - **Predicates:** equality ($=$), inequality ($<$, \neq)
 - **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

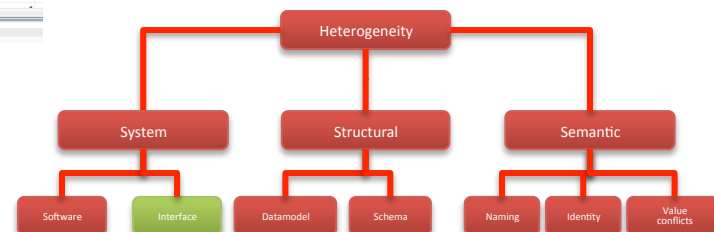
- Interface Heterogeneity – Examples
 - Google search (+/-, site:, intitle:, filetype:



1.1 System Heterogeneity

- Interface Heterogeneity – Examples – SQL

C_ID	C_D_ID	C_ID	C_DISCOUNT	C_CREDIT	C_LAST	C_FIRST	C_CREDIT_LIM	C_BALANCE	C_YTD_PAYMENT	C_PAYMENT_CNT
1	1	1	0.4156 GC	BARBLEANTI	ezdb (s lq l nzu)		50000	5003.2	4613.35	3
2	1	1	0.1041 GC	BARBLECALLY	okio t n t p r		50000	-10	10	1
3	1	1	0.356 GC	BARBLEATION	krpbhdydrmp		50000	35616.4	10	1
4	1	1	0.1387 GC	BARBLEEING	anedkycxlalr		50000	67503.81	10	1
5	1	1	0.3913 GC	BARPRIBAR	rbrtbklclu		50000	59286.28	10	1
6	1	1	0.3428 GC	BARPRIDIGHT	s f u s a d r c b y x		50000	-3951.01	3951.01	2
7	1	1	0.1980 GC	BARPRIABLE	tpkqhvqv lvt		50000	-1149.05	1149.05	2
8	1	1	0.432 GC	BARPRIPRI	sxeoyz		50000	-1209.53	1209.53	2
9	1	1	0.261 GC	BARPRIPRES	yasohdstgfencrr		50000	31377.55	10	1
10	1	1	0.0771 BC	BARPRIESE	hmvdraezchaxj		50000	38069.05	10	1
11	1	1	0.2667 GC	BARPRIANTI	r n v d f n q u l		50000	53852.94	10	1
12	1	1	0.0507 GC	BARPRICALLY	stjztkvtho		50000	-10	10	1
13	1	1	0.3148 GC	BARPRIATION	tezmxy		50000	52543.75	4787.26	4
14	1	1	0.4612 GC	BARPRIEING	tzoksqcncrsu		50000	85143.57	10	1
15	1	1	0.4508 GC	BARPRESBAR	rxwectf		50000	17865.26	3054.87	2
16	1	1	0.3242 GC	BARPRESOUGHT	ocTtuoy		50000	4273.91	10	1
17	1	1	0.1186 GC	BARPRESABLE	jbdnwf		50000	24632.91	15120.601	6



1.1 System Heterogeneity

- Interface Heterogeneity – Examples – SQL

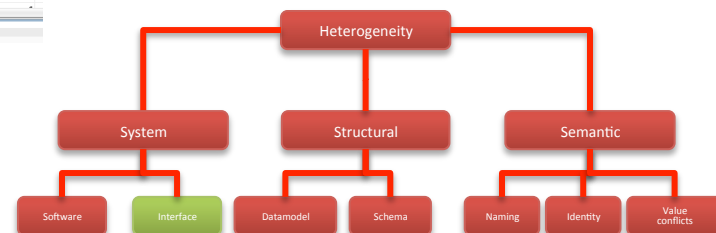
Oracle SQL Developer: ilgeti.cs.ill.edu - tpcc

Worksheet

```
SELECT * FROM customer;
```

Query Result: SQL | Fetched 50 rows in 0.008 seconds

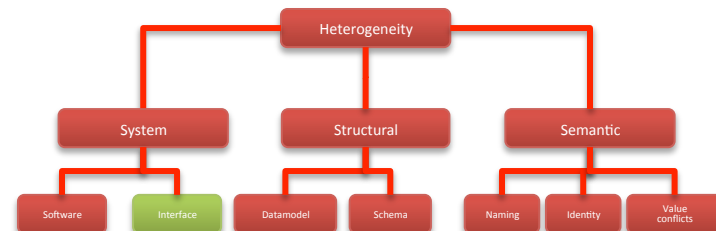
C_ID	C_D_ID	C_ID	C_DISCOUNT	C_CREDIT	C_LAST	C_FIRST	C_CREDIT_LIM	C_BALANCE	C_YTD_PAYMENT	C_PAYMENT_CNT
1	1	1	27	0.4156 GC	BARBLEANTI	ezdb (s lq l nzu)	50000	5003.2	4613.35	3
2	1	1	28	0.1041 GC	BARBLECALLY	okio t n t p r	50000	-10	10	1
3	1	1	29	0.356 GC	BARBLEATION	krpbhdydrmp	50000	35616.4	10	1
4	1	1	30	0.1387 GC	BARBLEEING	anedkycxlalr	50000	67503.81	10	1
5	1	1	31	0.3913 GC	BARPRIBAR	rbrtbklclu	50000	59286.28	10	1
6	1	1	32	0.3428 GC	BARPRIDIGHT	s f u s a d r c b y x	50000	-3951.01	3951.01	2
7	1	1	33	0.1980 GC	BARPRIABLE	tpkqhvqv lvt	50000	-1149.05	1149.05	2
8	1	1	34	0.432 GC	BARPRIPRI	sxeoyz	50000	-1209.53	1209.53	2
9	1	1	35	0.261 GC	BARPRIPRES	yasohdstgfencrr	50000	31377.55	10	1
10	1	1	36	0.0771 BC	BARPRIESE	hmvdraezchaxj	50000	38069.05	10	1
11	1	1	37	0.2667 GC	BARPRIANTI	r n v d f n q u l	50000	53852.94	10	1
12	1	1	38	0.0507 GC	BARPRICALLY	stjztkvtho	50000	-10	10	1
13	1	1	39	0.3148 GC	BARPRIATION	tezmxy	50000	52543.75	4787.26	4
14	1	1	40	0.4612 GC	BARPRIEING	tzoksqcncrsu	50000	85143.57	10	1
15	1	1	41	0.4508 GC	BARPRESBAR	rxwectf	50000	17865.26	3054.87	2
16	1	1	42	0.3242 GC	BARPRESOUGHT	ocTtuoy	50000	4273.91	10	1
17	1	1	43	0.1186 GC	BARPRESABLE	jbdnwf	50000	24632.91	15120.601	6



1.1 System Heterogeneity

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

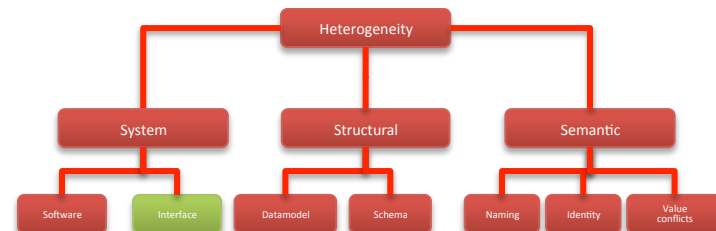
The image shows a screenshot of the Amazon website's Advanced Search page. The browser address bar shows the URL: www.amazon.com/Advanced-Search-Books/b/ref=sv_b_07ie=UTF8&node=241582011. The page features a search bar with a dropdown menu set to 'Books'. Below the search bar, there are several input fields for 'Keywords', 'Author', 'Title', 'ISBN(s)', and 'Publisher'. To the right, there are dropdown menus for 'Condition' (set to 'All Conditions'), 'Format', 'Reader Age' (set to 'All Ages'), 'Language' (set to 'All Languages'), and 'Pub. Date' (with 'Month' and 'Year' dropdowns). A 'Search' button is located at the bottom of the form. Three blue callout boxes are overlaid on the image: one pointing to the search bar labeled 'Keyword search', one pointing to the 'Subject' dropdown menu labeled 'Fixed choices', and one pointing to the 'Condition' dropdown menu labeled '"Bound parameter"'. Below the search form, there is a section titled 'Real-world Examples' with text: 'Trying to find books written by Malcolm X but not an autobiography? Try this search: Put **Malcolm X** in the 'Author' field and **~autobiography** in the 'Keywords' field. See the results. Looking for the exact books from your 20th Century American Literature syllabus? Enter all the ISI the 'ISBN' field, with a '|' (pipe) between each one. E.g. **9780140285000 | 9780743273565 | 9780061120060**. See the results



1.1 System Heterogeneity

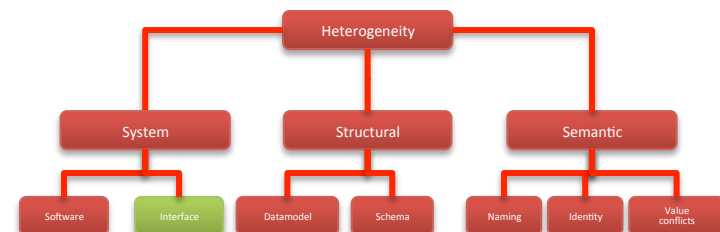
- Interface Heterogeneity – Examples
– Email-client

The screenshot shows a search criteria dialog box for an email client. The dialog has several sections: 'Name' (Local Folders), 'Create as a subfolder of' (Local Folders), 'Select the folders to search' (Choose...), and 'Configure the search criteria used for this saved search folder'. The search criteria section has three radio buttons: 'Match all of the following' (selected), 'Match any of the following', and 'Match all messages'. Below these are two rows of search criteria, each with a dropdown menu for the field (both set to 'Subject'), a dropdown menu for the comparison operator (both set to 'contains'), and an empty text input field. There are '+' and '-' buttons next to each row. At the bottom are 'Cancel' and 'Create' buttons. Three blue callout boxes with white text are overlaid on the image: 'Name Query' points to the 'Name' field; 'Disjunctive or conjunctive' points to the radio buttons; and 'Comparison operator' points to the 'contains' dropdown in the first row.



1.1 System Heterogeneity

- 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?

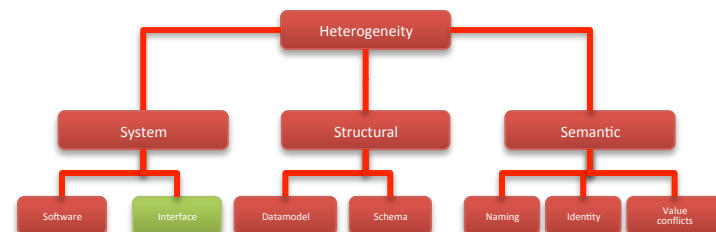
Books Search

Keywords

Author

Title

ISBN(s)



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, ...

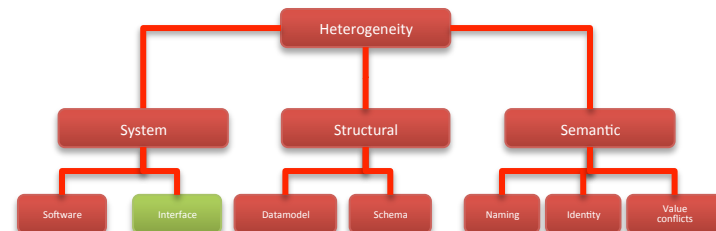
Books Search

Keywords

Author

Title

ISBN(s)



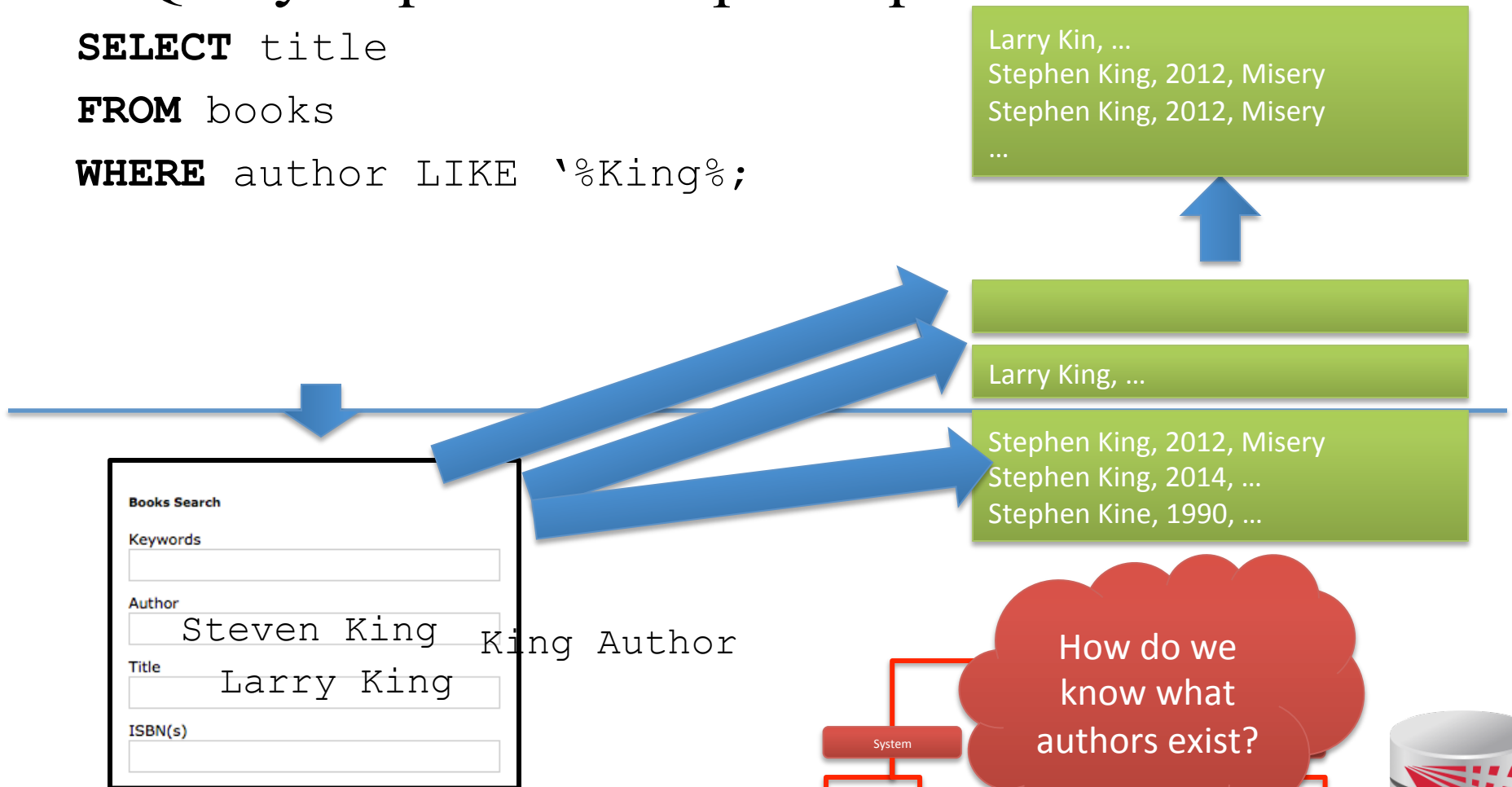
1.1 System Heterogeneity

- Query requires multiple requests

```
SELECT title
```

```
FROM books
```

```
WHERE author LIKE '%King%';
```



Larry King, ...
Stephen King, 2012, Misery
Stephen King, 2012, Misery
...

Larry King, ...

Larry King, ...

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

How do we know what authors exist?



1.1 System Heterogeneity

- Query cannot be answered

```
SELECT title
```

```
FROM books
```

```
WHERE genre = 'SciFi';
```

Web form is
for history
book only!

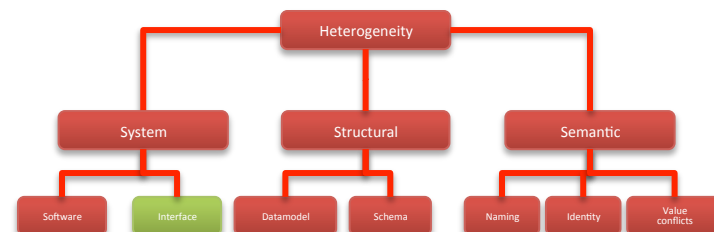
Books Search

Keywords

Author

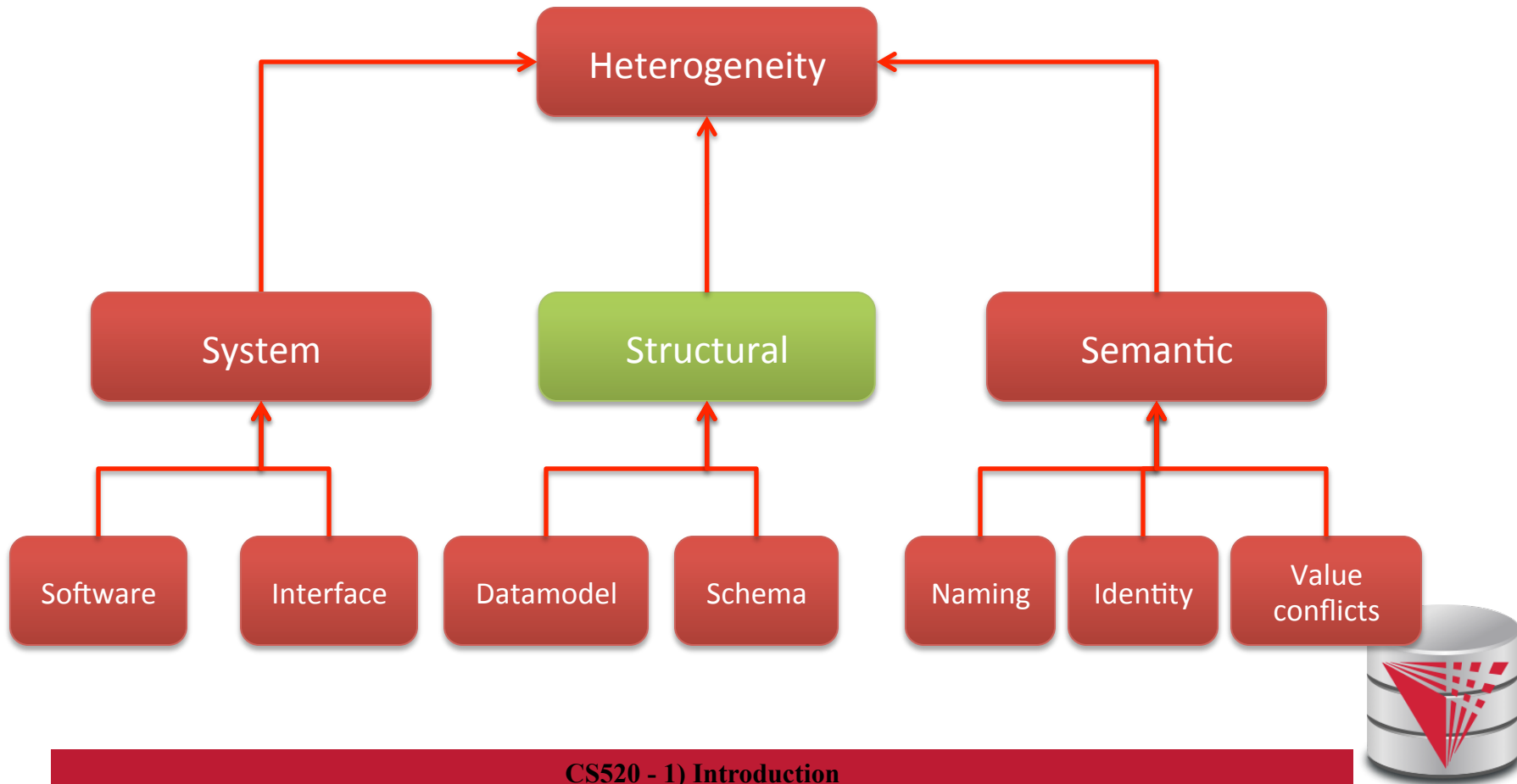
Title

ISBN(s)



1.1 Heterogeneity + Autonomy

- Taxonomy of Heterogeneity



1.1 Structural Heterogeneity

- **Data model**

- Different semantic/expressiveness
- Different structure

- **Schema**

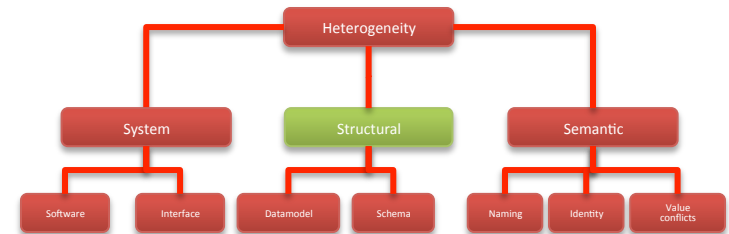
- **Integrity constraints, keys**

- **Schema elements:**

- use attribute or separate relations)

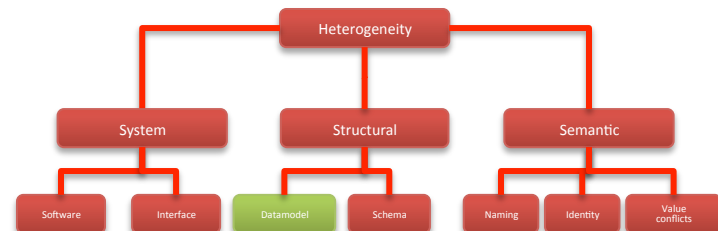
- **Structure:**

- e.g., normalized vs. denormalized relational schema



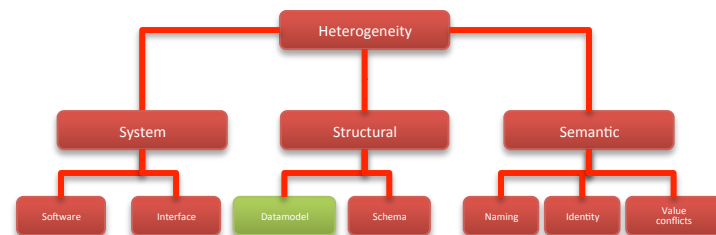
1.1 Structural Heterogeneity

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



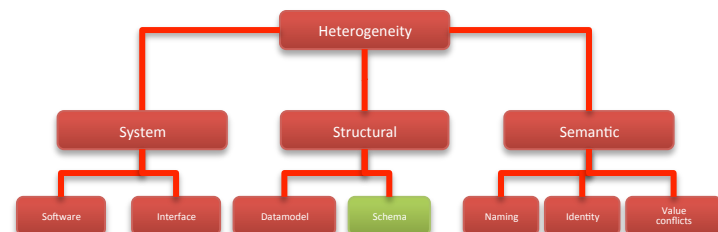
1.1 Structural Heterogeneity

- Example: data model
 - Relational model
 - XML model
 - JSON
 - OO
- Person and their addresses



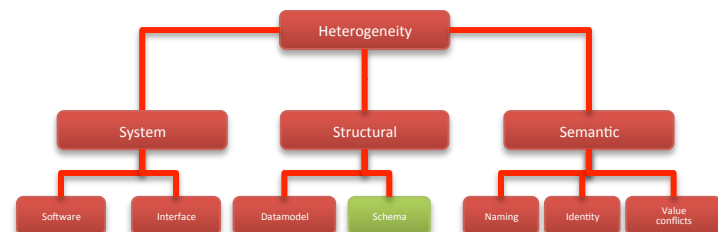
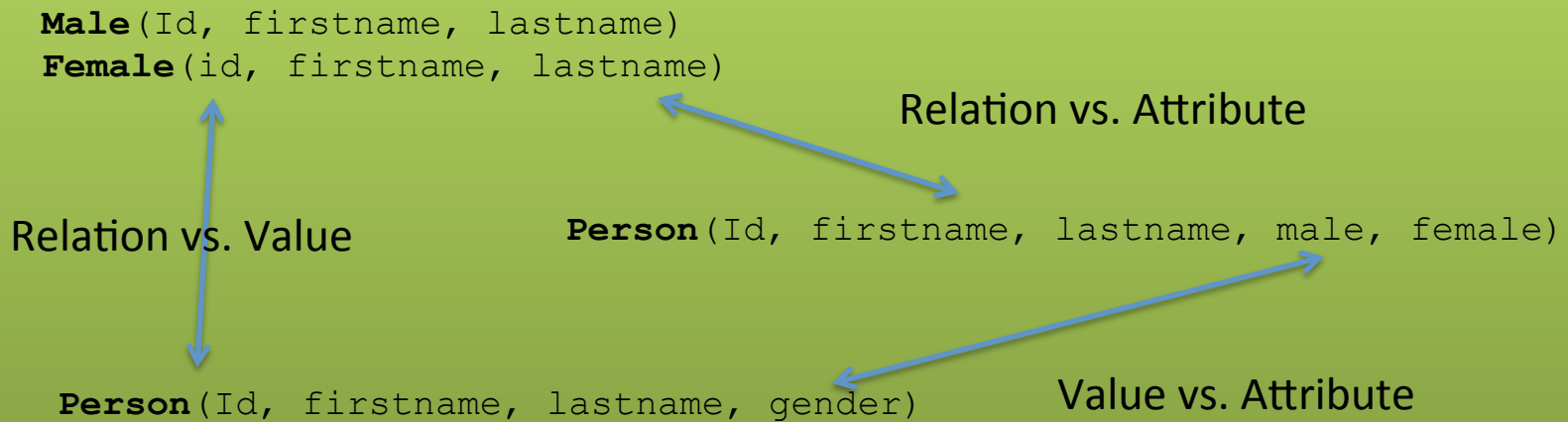
1.1 Structural Heterogeneity

- **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

Example: Modeling choices



1.1 Structural Heterogeneity

- **Relation-relation conflicts**

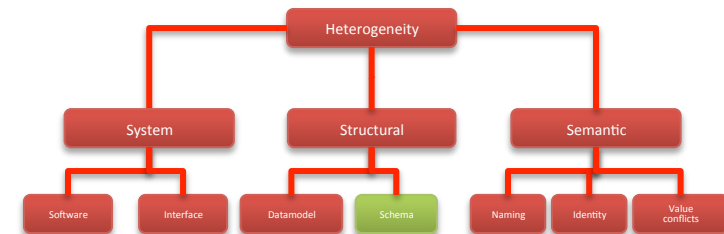
- Naming conflicts

- 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



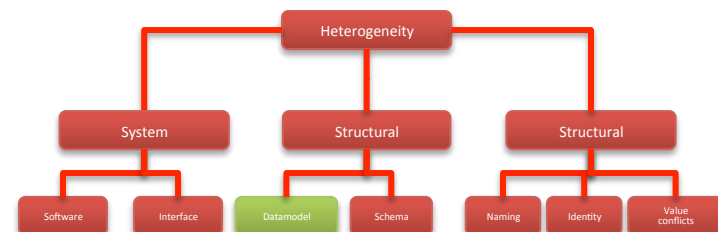
1.1 Structural Heterogeneity

Example: Conflicts between relations

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

```
Person(Id, name, gender, birthday)
```

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



1.1 Structural Heterogeneity

Multiple attribute
vs one attribute

Example: Conflict between relations

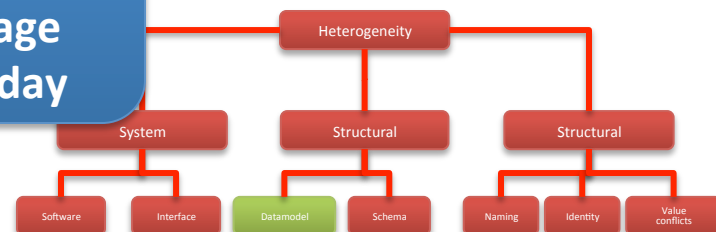
Person(Id, firstname, lastname, male, female)

Person(Id, name, gender, birthday)

Manager(Id, name, gender, age)

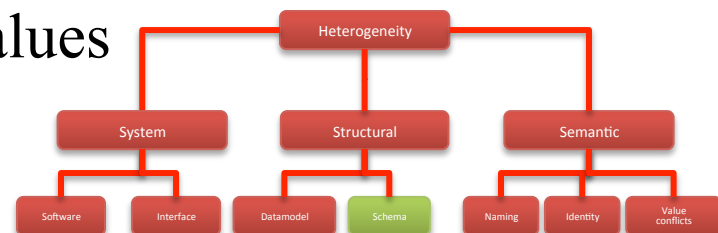
Missing derivable
attribute:
Role

Derivable
attribute:
Compute age
from birthday



1.1 Structural Heterogeneity

- **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

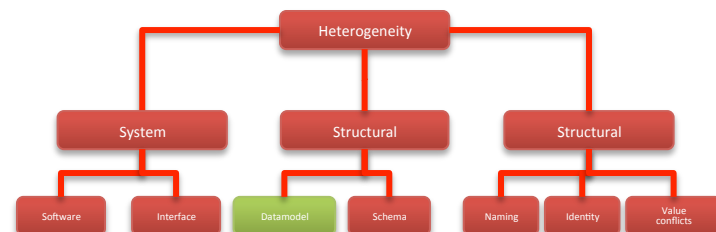


1.1 Structural Heterogeneity

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

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

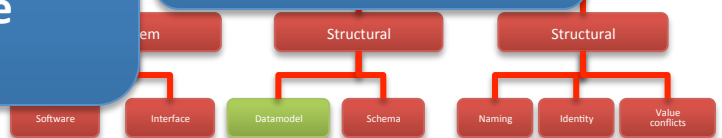
Conflicting format

Conflicting datatype

synonym

Conflicting constraint

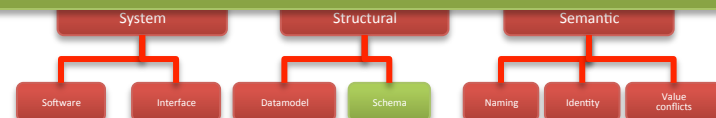
Conflicting default value



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

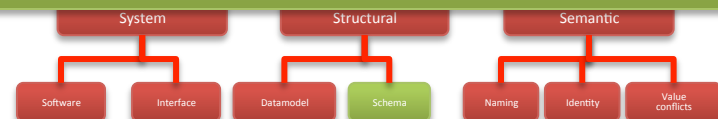
Example



1.1 Structural Heterogeneity

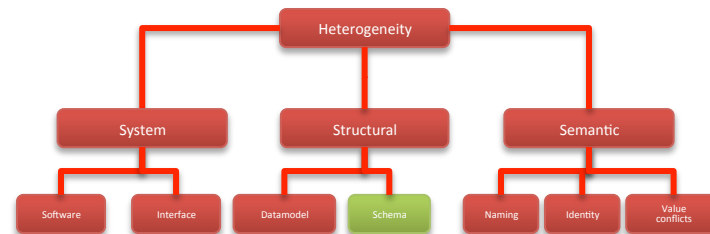
- **Nested vs. flat**
 - Association between entities can be represented using nesting or references (previous slides)

Example



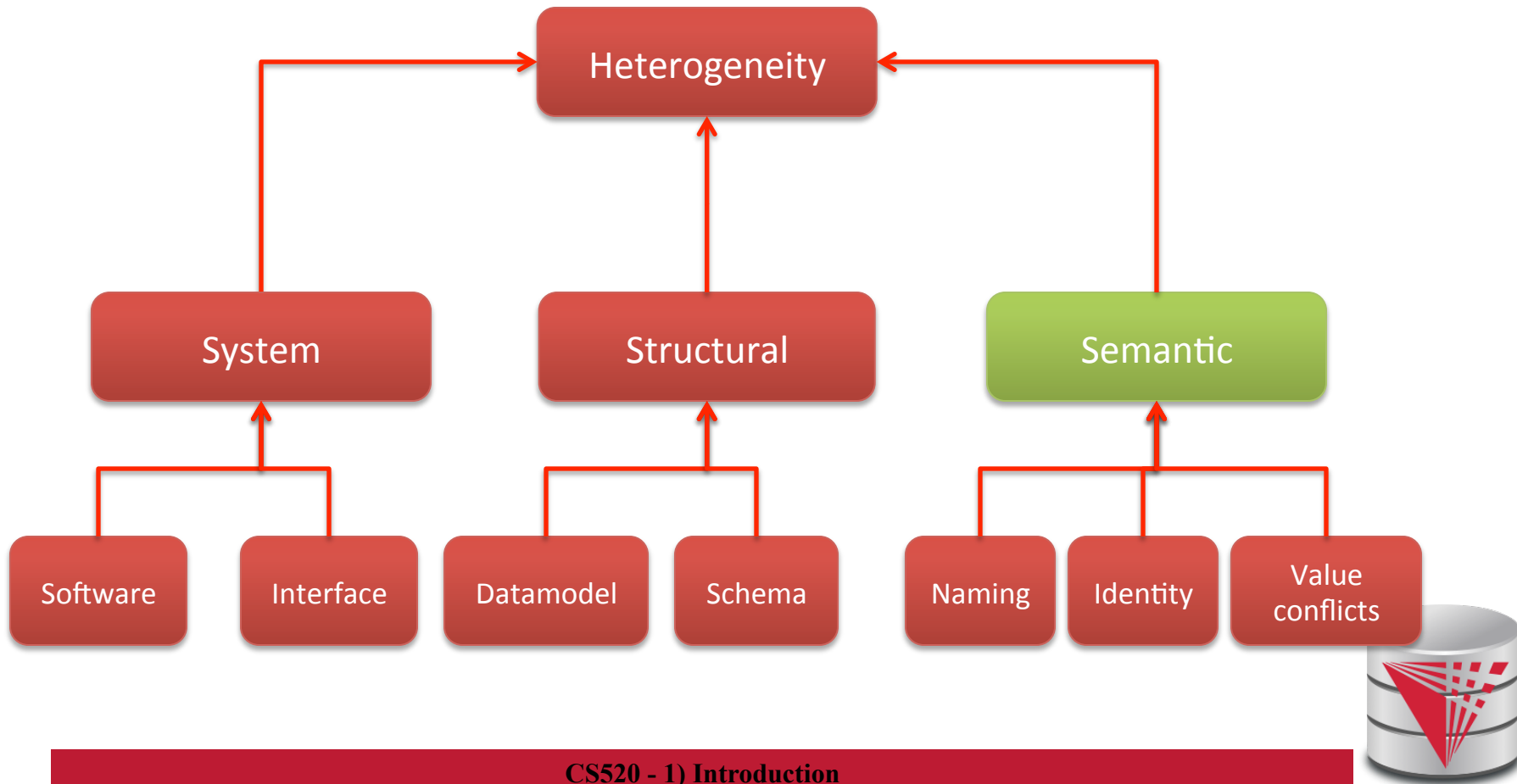
1.1 Structural Heterogeneity

- **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)



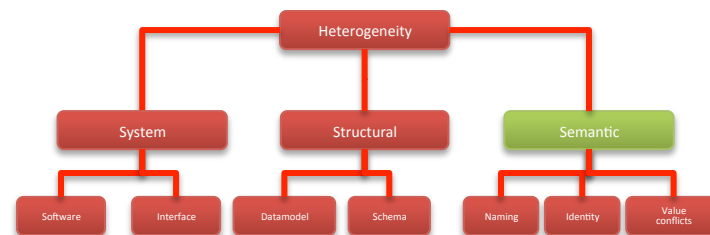
1.1 Heterogeneity + Autonomy

- Taxonomy of Heterogeneity



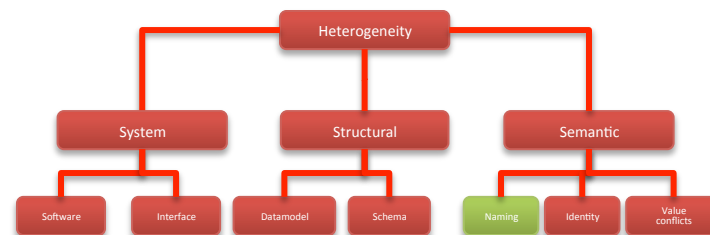
1.1 Semantic Heterogeneity

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



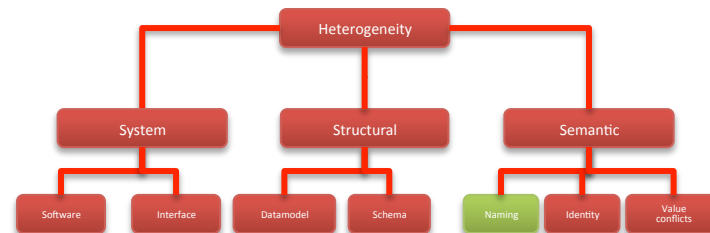
1.1 Semantic Heterogeneity

- **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

- **Ontological concepts**
 - Relationships between concepts
 - $A = B$ - Equivalence
 - $A \subseteq B$ - Inclusion
 - $A \cap B$ - Overlap
 - $A \neq B$ - Disjunction

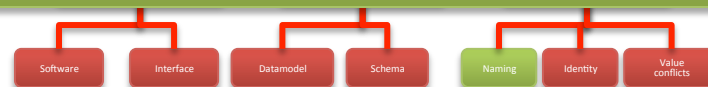


1.1 Semantic Heterogeneity

- **Ontological concepts**
 - Relationships between concepts
 - $A = B$ - Equivalence
 - $A \subseteq B$ - Inclusion
 - $A \cap B$ - Overlap
 - $A \neq B$ - Disjunction

Example

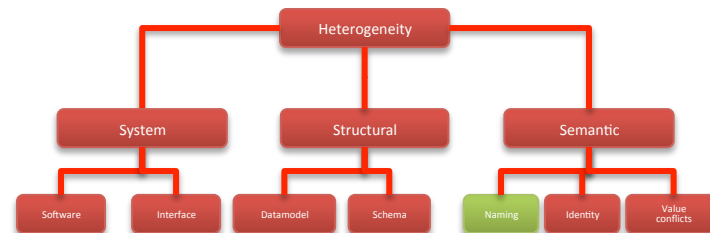
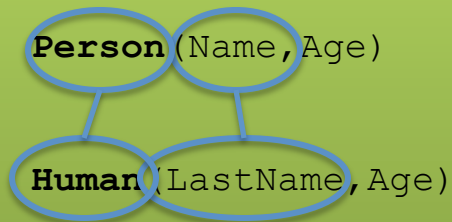
Equivalence: Human vs Homo sapiens
Inclusion: Bird vs Animal
Overlap: Animal vs aquatic lifeform
Disjunction: Fish vs Mammal



1.1 Semantic Heterogeneity

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

Example



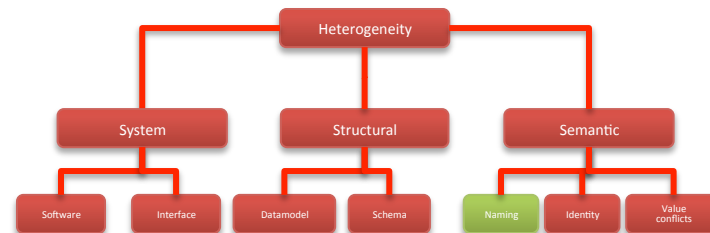
1.1 Semantic Heterogeneity

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

Example

Person (Title, Name)

Movie (Title, Year)



1.1 Semantic Heterogeneity

- **Naming concepts (units)**

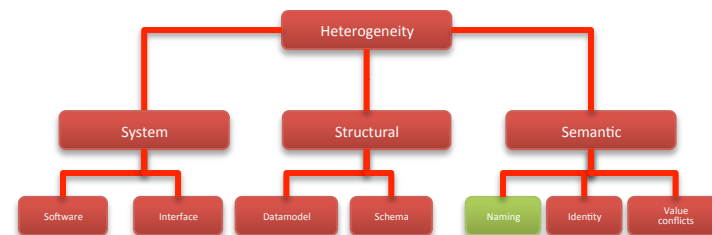
Example

Person (Title, Name, Salary)

\$

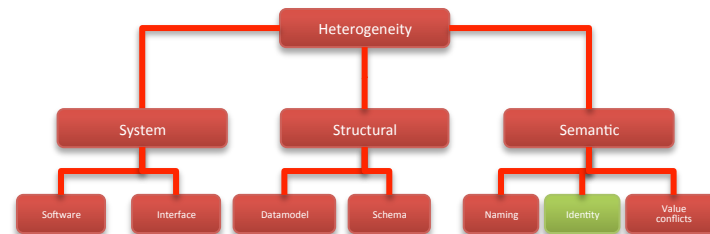
Person (Title, Name, Salary)

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**
 - ...



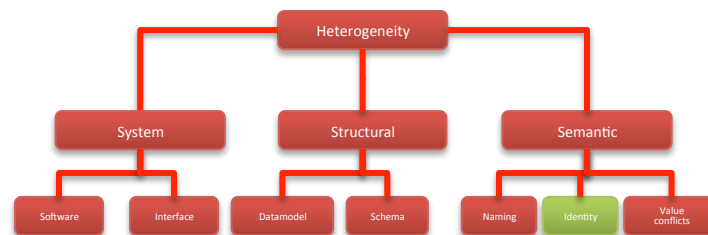
1.1 Semantic Heterogeneity

- Identity Conflicts

Example

(IBM, 300000000, USA)

(International Business Machines Corporation, 50000)



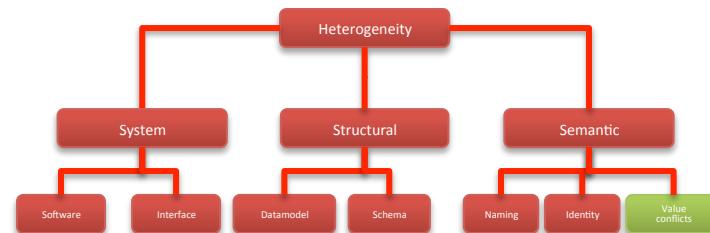
- **Value Conflicts**

- Objects representing the same entities have conflicting values for semantically equivalent attributes

- We have to identified that these objects are represent the same entity first!

- Resolving such conflicts require **Data Fusion**

- Pick value from conflicting values
- Numerical methods: e.g., average
- Preferred value
- ...



- **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

- **Virtual data integration**
- **Data Exchange**
- **Peer-to-peer data integration**
- **Data warehousing**
- **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 Integrity constraints

- **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

- 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 are represent values
 - $R(x,y,z)$
 - $x = y$



1.4 Integrity Constraints

Example

Primary Key $R(\underline{A}, B)$:

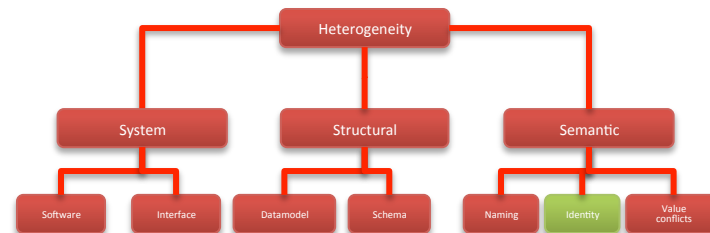
$$\forall x, y, z : R(x, y) \wedge R(x, z) \rightarrow y = z$$

Functional Dependency $R(A, B)$ with $A \rightarrow B$:

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

Foreign Key $R(\underline{A}, B)$, $S(C, D)$ 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

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



1.4 Datalog

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

$$Q(\vec{x}) : -R_1(\vec{x}_1), \dots, R_n(\vec{x}_n).$$

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



1.4 Datalog

- A rule stands for: for each binding of variables to values from the database so that all the relational atoms R_1 to R_n evaluate to true, return the bindings of x
 - All variables in the **head** (left-hand side) have to occur also in the **body** (right-hand side)

$$Q(\vec{x}) : -R_1(\vec{x}_1), \dots, R_n(\vec{x}_n).$$



- **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

$$Q(\vec{x}) : -R_1(\vec{x}_1), \dots, R_n(\vec{x}_n).$$



1.4 Datalog

- Different flavors of datalog
 - Conjunctive query
 - Only one rule
 - Expressible as Select-project-join query in relational algebra
 - 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., $<$



- Different flavors of datalog
 - Recursion
 - Rules may have recursion:
 - E.g., head predicate in the body
 - Fix point semantics based on immediate consequence operator
 - Negation
 - Negated relational atoms allowed
 - Require that every variable used in a negated atom also occurs in at least on positive atom
 - Combined Negation + recursion



1.4 Datalog

Example

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

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

$Q_3(x, x) : R(x, x) .$

$Q_4(x, y) : R(x, y) .$

$Q_5(x, x) : R(x, y), R(x, x) .$

$Q_6(x, z) : R(x, y), R(y, z) .$



Example

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

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

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

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

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

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



1.4 Datalog

Example

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

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

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

$Q_{\text{notSameComp}}(x, y) : Q_{\text{node}}(x) , Q_{\text{node}}(y) ,$
 $\text{not}(Q_{\text{reach}}(x, y)) ,$
 $\text{not}(Q_{\text{reach}}(y, x)) .$



1.4 Containment and Equivalence

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.

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.

Example

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

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

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

$Q_{up2Hop}(x, z) : \text{hop}(x, z) .$

$Q_{sym}(x, y) : \text{hop}(x, y) .$

$Q_{sym}(x, y) : \text{hop}(y, x) .$

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



1.4 Complexity of Eq. and Cont.

Set semantics	Relational Algebra	Conjunctive Queries (CQ)	Union of Conjunctive Queries (UCQ)	Monotone Queries/CQ \neq
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.

Bag semantics	Relational Algebra	Conjunctive Queries (CQ)	Union of Conjunctive Queries (UCQ)	Monotone Queries/ CQ \neq
Query Equivalence	Undecidable	Equivalent to graph isomorphism		It is in PSPACE, lower-bound unknown
Query Containment	Undecidable	Open Problem	Undecidable	Π_2^P -complete



1.4 Containment Mappings

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



1.4 Boolean Conjunctive Queries

- A conjunctive query is boolean if the head does not have any variables
 - $Q() :- \text{hop}(x,y), \text{hop}(y,z)$
 - We will use $Q :- \dots$ as a convention for $Q() :- \dots$
 - What is the result of a boolean query
 - Empty result $\{\}$, e.g., no $\text{hop}(x,y), \text{hop}(y,z)$
 - If there are tuples matching the body, then a tuple zero attributes is returned $\{()\}$
 - \rightarrow 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)

$Q \text{ :- } \text{hop}(x, y)$

```
SELECT EXISTS (SELECT * FROM hop)
```

Note: in Oracle and DB2 we need a from clause



1.4 Boolean Conjunctive Queries

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



1.4 Boolean Conjunctive Queries

- BCQ in SQL

Example

$Q :- \text{hop}(x, y), \text{hop}(y, z)$

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



1.4 Containment Mappings

- 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 so that:

$$\Psi(\text{head}(Q)) = \text{head}(Q')$$

$$\forall R(\vec{x}_i) \in \text{body}(Q) : \Psi(\vec{x}_i) \in \text{body}(Q')$$



1.4 Containment Mappings

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

$$Q_1(u, z) : R(u, z) .$$

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

Can we find a containment mapping?



1.4 Containment Mappings

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

$$Q_1(u, z) : R(u, z) .$$

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

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

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



1.4 Containment Mappings

Example

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

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



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?

$Q_1 \rightarrow Q_2$: none exists

$Q_2 \rightarrow Q_1$: $\Psi(x) = a, \Psi(y) = b$



1.4 Containment Mappings

Example

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

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

$$Q_1 \rightarrow Q_2 : \Psi(a) = x, \Psi(b) = y, \Psi(c) = y$$

$$Q_2 \rightarrow Q_1 : \Psi(x) = a, \Psi(y) = b$$



1.4 Containment Background

- 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
 - $I = \{\text{hop}(a,b), \text{hop}(b,c)\} \rightarrow Q^I :- \text{hop}(a,b), \text{hop}(b,c)$
 - Canonical instance I^Q for query Q
 - Interpret each conjunct as a tuple
 - Interpret variables as constants
 - $Q :- \text{hop}(a,a) \rightarrow I^Q = \{\text{hop}(a,a)\}$



1.4 Containment Background

- Containment Mapping \leftrightarrow 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 $\text{vars}(Q_2)$ to D
 - Compose this with the containment mapping and prove that this is a result for Q_1



1.4 Containment Mappings

Example

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

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

$Q_2 \rightarrow Q_1 : \Psi(x) = a, \Psi(y) = b$

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

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

$\phi(a) = 1, \phi(b) = 2, \phi(c) = 1$

$\Psi \phi(x) = 1, \Psi \phi(y) = 2$



1.4 Containment Background

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



1.4 Containment Mappings

Example

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

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

$$Q_2 \rightarrow Q_1 : \Psi(x) = a, \Psi(y) = b$$

$$I^{Q_1} = \{ (a, b), (c, b) \}$$

$$Q_2(I^{Q_1}) = \{ () \}$$

$$\varphi(x) = a, \varphi(y) = b$$

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/cms277/Winter10/>



1.4 Containment Background

- 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

Example

$Q_1 () : \mathbf{R(a, b)}, R(c, b) .$

$Q_2 () : \mathbf{R(x, y)} .$

$Q_2 \rightarrow Q_1 : \Psi(x) = a, \Psi(y) = b$

If there exists tuples

$\mathbf{R(a, b)}$ and $\mathbf{R(c, b)}$

in R that make Q_1 true, then we take

$\mathbf{R(a, b)}$

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
 - \rightarrow same tuples returned, but again Q 's condition is more restrictive



1.4 Containment Mappings

Example

$Q_1(\mathbf{a}) : \mathbf{R(a, b)}, \mathbf{R(c, b)}$.

$Q_2(\mathbf{x}) : \mathbf{R(x, y)}$.

$Q_2 \rightarrow Q_1 : \Psi(x) = a, \Psi(y) = b$

For every

$\mathbf{R(a, b)}$ and $\mathbf{R(c, b)}$

Q_1 returns $\mathbf{(a)}$ and for every

$\mathbf{R(a, b)}$

Q_2 returns $\mathbf{(a)}$



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

Definition: Similarity Measure

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

- $d(p,p) = 0$
- $d(p,q) \geq 0$

- **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 $[0,1]$**



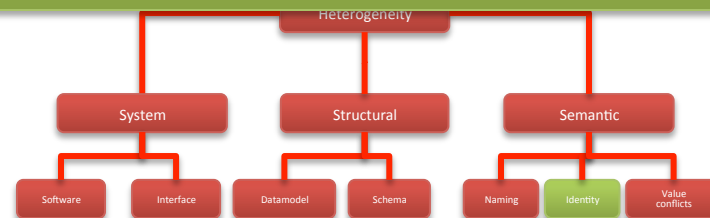
1.4 Similarity Measures

Example

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

Euclidian distance: $d(p, q) = \sqrt{\sum_{i=1}^n (p[i] - q[i])^2}$
N-dimensional space

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



1.4 Similarity Measures

Definition: Metric

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

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

- **Metric is a stricter definition**
- **Which of the previous similarity measure is a metric?**



1.4 Similarity Measures

Definition: Metric

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

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

- **Metric is a stricter definition**
- **Which of the previous similarity measure is a metric?**
 - **All of them!**



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 Neighborhood-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**



- 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



- 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

