ILLINOIS INSTITUTE OF TECHNOLOGY

CS520
Data Integration, Warehousing, and Provenance

Course Info

IIT DBGroup

Boris Glavic

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

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

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

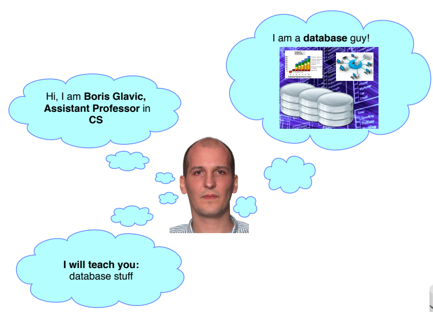
Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

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

About me

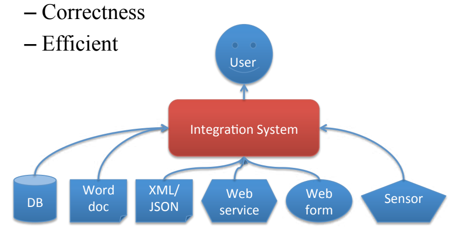
ILLINOIS INSTITUTE OF TECHNOLOGY



What is information integration?

ILLINOIS INSTITUTE OF TECHNOLOGY

- Combination of data and content from multiple sources into a common format
  - Completeness
  - Correctness
  - Efficient



Why Information Integration?

ILLINOIS INSTITUTE OF TECHNOLOGY

- Data is already available, right?
- …, but
- Heterogeneity
  - Structural
    - Data model (relational, XML, unstructured)
    - Schema (if there)
  - Semantic
    - Naming and identity conflicts
    - Data conflicts
  - Syntactic
    - Interfaces (web form, query language, binary file)

Why Information Integration?

ILLINOIS INSTITUTE OF TECHNOLOGY

- Autonomy
  - Sources may not give you unlimited access
    - Web form only support a fixed format of queries
    - Does not allow access to unlimited amounts of data
  - Source may not be available all the time
    - Naming and identity conflicts
    - Data conflicts
  - Data, schema, and interfaces of sources may change
    - Potentially without notice

"Real World" Examples?

ILLINOIS INSTITUTE OF TECHNOLOGY

- Portal websites
  - Flight websites (e.g., Expedia) gather data from multiple airlines, hotels
- Google News
  - Integrates information from a large number of news sources
- Science:
  - Biomedical data source
- Business
  - Warehouses: integrate transactional data

6 CS520 - Course Info

Example Integration Problem [1]

ILLINOIS INSTITUTE OF TECHNOLOGY

- Integrate stock ticker data from two web services A and B
  - **Service A**: Web form (Company name, year)
  - **Service B**: Web form (year)

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

7 CS520 - Course Info

Example Integration Problem [2]

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Service A**:

```
<Stock>
    <Company>IBM</Company>
    <DollarValue>155.8</DollarValue>
    <Month>12</Month>
</Stock>
```

- **Service B**:

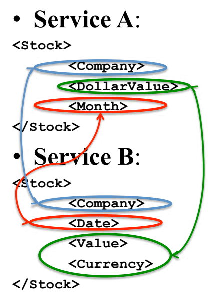
```
<Stock>
    <Company>International Business Machines</Company>
    <Date>2014-08-01</Date>
    <Value>106.8</Value>
    <Currency>Euro</Currency>
</Stock>
```

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

8 CS520 - Course Info

Example Integration Problem [2]

ILLINOIS INSTITUTE OF TECHNOLOGY

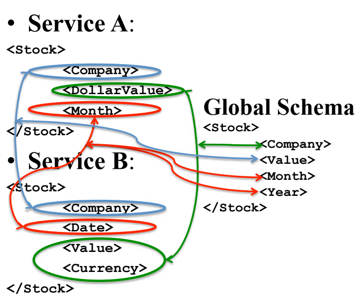


Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

9 CS520 - Course Info

Example Integration Problem [2]

ILLINOIS INSTITUTE OF TECHNOLOGY



Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

10 CS520 - Course Info

Example Integration Problem [3]

ILLINOIS INSTITUTE OF TECHNOLOGY

- SQL interface for integrated service

```
SELECT month, value
FROM ticker
WHERE year = 2014
      AND cmp = 'IBM'
```

- Service A: **(IBM, 2014)**
- Service B: **(2014)**

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

11 CS520 - Course Info

Example Integration Problem [4]

- For web service A we can either
  - Get stocks for **IBM** in **all years**
  - Get stocks for **all companies** in **2014**
  - Get stocks for **IBM** in **2014**
- Trade-off between amount of processing that we have to do locally, amount of data that is shipped, …

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

12 CS520 - Course Info

Example Integration Problem [5]

- **Service A**: (IBM, 2014)
- **Service B**: (2014)

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

13 CS520 - Course Info

Example Integration Problem [6]

- **Service A**:

```
<Stock>
        <Company>IBM</Company>
        <DollarValue>155.8</DollarValue>
        <Month>12</Month>
…
```

- **Service B**:

```
<Stock>
        <Company>International Business Machines</Company>
        <Date>2014-12-01</Date>
        <Value>106.8</Value>
        <Currency>Euro</Currency>
…
```

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

14 CS520 - Course Info

Example Integration Problem [7]

- IBM vs. Integrated Business Machines

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

15 CS520 - Course Info

Example Integration Problem [8]

- Granularity of time attribute
  - Month vs. data
- What if both services return different values (after adapting granularity)
  - Average?
  - Median?
  - Trust-based?

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

16 CS520 - Course Info

Example Integration Problem [9]

- Return final results:

```
<Stock>
        <Month>01</Month>
        <Value>105</Value>
</Stock>
…
<Stock>
        <Month>12</Month>
        <Value>107</Value>
</Stock>
```

Steps
1) Interfaces
2) Schema integration
3) Translate queries
4) Optimization
5) Send queries to sources
6) Gather query results
7) Entity resolution
8) Fusion
9) Return final results

17 CS520 - Course Info

Why hard?

- System challenges
  - Different platforms (OS/Software)
  - Efficient query processing over multiple heterogeneous systems
- Social challenges
  - Find relevant data
  - Convince people to share their data
- Heterogeneity of data and schemas
  - A problem that even exists if we use same system

Why hard? Cont.

- Often called **AI-complete**
  - Meaning: "It requires human intelligence to solve the problem"
  - Unlikely that general completely automated solutions will exit
- So why do we still sit here
  - There exist automated solutions for relevant less general problems
  - Semi-automated solutions can reduce user effort (and may be less error prone)

AI completeness

- Yes, but still why is this problem really so hard?
  - **Lack of information**: e.g., the attributes of a database schema have only names and data types, but no computer interpretable information on what type of information is stored in the attribute
  - **Undecidable computational problems**: to decide whether a user query can be answered from a set of sources that provide different views on the data requires **query containment** checks which are undecidable for certain query types

Relevant less general problems

- **Data cleaning**:
  - Clean dirty data before integration
  - Conformance with a set of constraints
  - Deal with missing and outlier values
- **Entity resolution**
  - Determine which objects from multiple dataset represent the same real world entity
- **Data fusion**
  - Merge (potentially conflicting) data for the same entity

Relevant less general problems

- **Schema matching**
  - Given two schemas determine which elements store the same type of information
- **Schema mapping**
  - Describe the relationships between schemas
    - Allows us to rewrite queries written against one schema into queries of another schema
    - Allows us to translate data from one schema into

Relevant less general problems

- **Virtual data integration**
  - Answer queries written against a **global mediated schema** by running queries over **local sources**
- **Data exchange**
  - Map data from one schema into another
- **Warehousing: Extract, Transform, Load**
  - Clean, transform, fuse data and load it into a data warehouse to make it available for analysis

Relevant less general problems

- **Integration in Big Data Analytics**
  - Often "pay-as-you-go":
    - No or limited schema
    - Engines support wide variety of data formats
- **Provenance**
  - Information about the origin and creation process of data
  - Very important for integrated data
    - E.g., "from which data source is this part of my query result"

Webpage and Faculty

- **Course Info**
  - **Course Webpage**: http://cs.iit.edu/~cs520
  - **Google Group**: https://groups.google.com/d/forum/cs520-2015-spring-group
    - Used for announcements
    - Use it to discuss with me, TA, and fellow students
  - **Syllabus:** http://cs.iit.edu/~cs520/files/syllabus.pdf
- **Faculty**
  - **Boris Glavic** (http://cs.iit.edu/~glavic)
  - **Email:** bglavic@iit.edu
  - **Phone**: 312.567.5205
  - **Office**: Stuart Building, room 226C
  - **Office Hours**: Mondays, 12pm-1pm (and by appointment)

TAs

- **TAs**
  - **TBA**

Workload and Grading

- **Exams (60%)**
  - Final
- **Homework Assignments** (preparation for exams!)
  - Practice theory for final exam
  - Practice the tools we discuss in class
- **Literature Review (40%)**
  - In groups of 2 students
  - Topics will be announced soon
  - You have to read a research paper
  - Papers will be assigned in the first few weeks of the course
  - You will give a short presentation (15min) on the topic in class
  - You will write a report summarizing and criticizing the paper (up to 4 pages)

Course Objectives

- Understand the problems that arise with querying heterogeneous and autonomous data sources
- Understand the differences and similarities between the data integration/exchange, data warehouse, and Big Data analytics approaches
- Be able to build parts of a small data integration pipeline by "glueing" existing systems with new code

Course Objectives cont.

- Have learned formal languages for expressing schema mappings
- Understand the difference between virtual and materialized integration (data integration vs. data exchange)
- Understand the concept of data provenance and know how to compute provenance

Fraud Policies

ILLINOIS INSTITUTE OF TECHNOLOGY

- All work has to be original!
  - Cheating = 0 points for review/exam
  - Possibly E in course and further administrative sanctions
  - Every dishonesty will be reported to office of academic honesty
- Late policy:
  - -20% per day
  - You have to give your presentation to pass the course!
  - No exceptions!

Fraud Policies cont.

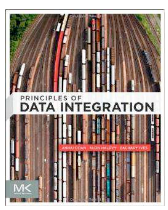
ILLINOIS INSTITUTE OF TECHNOLOGY

- Literature Review:
  - Every student has to contribute in both the presentation and report!
  - **Don't let others freeload on you hard work!**
    - Inform me or TA immediately

Reading and Prerequisites

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Textbook:** Doan, Halevy, and Ives.
  - **Principles of Data Integration**, 1st Edition
  - Morgan Kaufmann
  - Publication date: 2012
  - ISBN-13: 978-0124160446
  - Prerequisites:
    - CS 425



Additional Reading

ILLINOIS INSTITUTE OF TECHNOLOGY

- Papers assigned for literature review
- Optional: Standard database textbook

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

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

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/~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

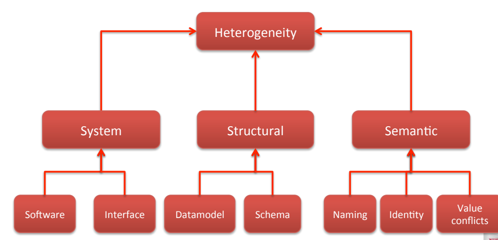
ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

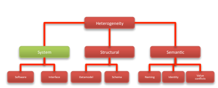
- Taxonomy of Heterogeneity



1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

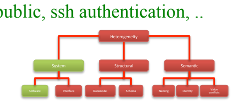
- Hardware/Software
  - Different hardware capabilities of sources
  - Different protocols, binary file formats, …
  - Different access control mechanism
- Interface Heterogeneity
  - Different interfaces for accessing data from a source
    - HTML forms
    - XML-Webservices
    - Declarative language



1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

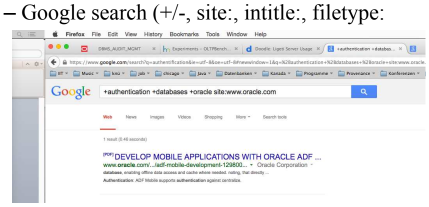
ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

ILLINOIS INSTITUTE OF TECHNOLOGY

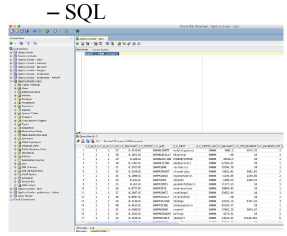
- Interface Heterogeneity – Examples
  - Google search (+/-, site:, intitle:, filetype:



1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

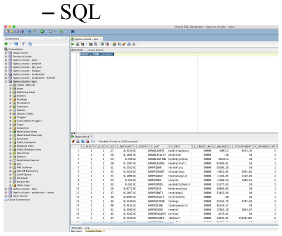
- Interface Heterogeneity – Examples
  - SQL



1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

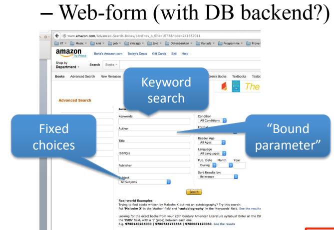
- Interface Heterogeneity – Examples
  - SQL



1.1 System Heterogeneity

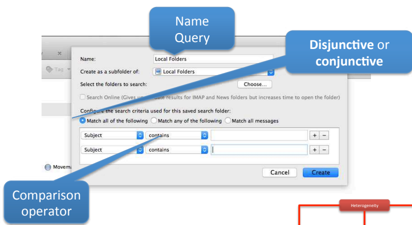
ILLINOIS INSTITUTE OF TECHNOLOGY

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



1.1 System Heterogeneity

- Interface Heterogeneity – Examples
  - Email-client



12 CS520 - 1) Introduction

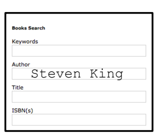
1.1 System Heterogeneity

- 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

13 CS520 - 1) Introduction

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?

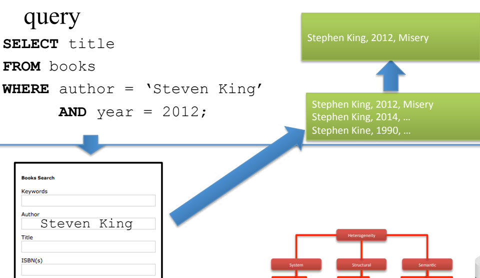


14 CS520 - 1) Introduction

1.1 System Heterogeneity

- Integration system has to process part of the query

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

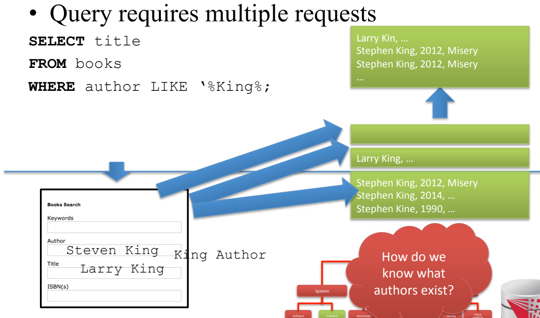


15 CS520 - 1) Introduction

1.1 System Heterogeneity

- Query requires multiple requests

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

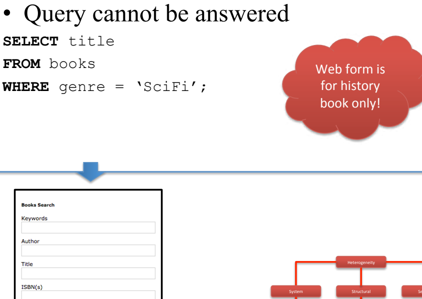


16 CS520 - 1) Introduction

1.1 System Heterogeneity

- Query cannot be answered

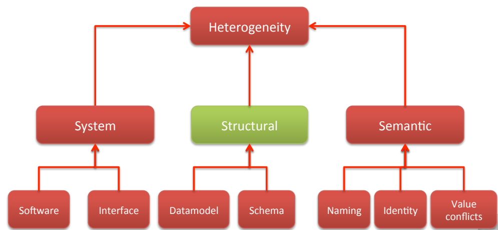
```
SELECT title
FROM books
WHERE genre = 'SciFi';
```



17 CS520 - 1) Introduction

1.1 Heterogeneity +Autonomy

- Taxonomy of Heterogeneity



18 CS520 - 1) Introduction

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

19 CS520 - 1) Introduction

1.1 Structural Heterogeneity

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

20 CS520 - 1) Introduction

1.1 Structural Heterogeneity

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

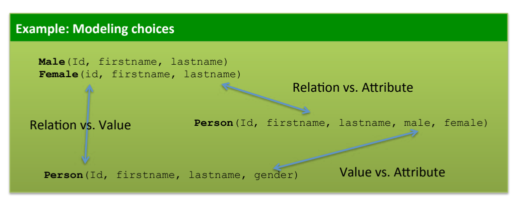
21 CS520 - 1) Introduction

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

22 CS520 - 1) Introduction

1.1 Structural Heterogeneity



23 CS520 - 1) Introduction

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

ILLINOIS INSTITUTE OF TECHNOLOGY

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: Conflicts 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

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

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

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

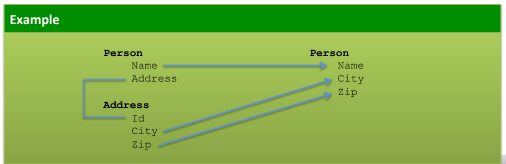
Conflicting datatype

synonym

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

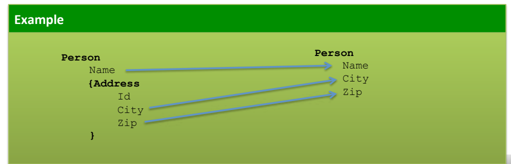
- **Normalized vs. denormalized**
  - E.g., relational model: Association between entities can be represented using multiple relations and foreign keys or one relation



1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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



1.1 Structural Heterogeneity

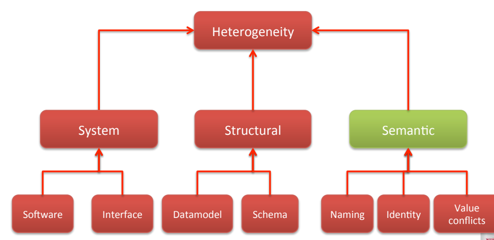
ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

36 CS520 - 1) Introduction

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

Example

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

37 CS520 - 1) Introduction

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

Example

```
Person(Name,Age)
Human(LastName,Age)
```

38 CS520 - 1) Introduction

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

Example

```
Person(Title,Name)
Movie(Title,Year)
```

39 CS520 - 1) Introduction

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Naming concepts (units)**

Example

```
Person(Title,Name,Salary)   $
Person(Title,Name,Salary)   CAD
```

40 CS520 - 1) Introduction

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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**
    - …

41 CS520 - 1) Introduction

1.1 Semantic Heterogeneity

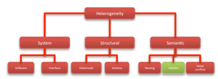
ILLINOIS INSTITUTE OF TECHNOLOGY

- Identity Conflicts

Example

(IBM,300000000,USA)

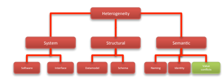
(International Business Machines Corporation,50000)



1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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 require **Data Fusion**
    - Pick value from conflicting values
    - Numerical methods: e.g., average
    - Preferred value
    - …



1.1 Autonomy

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

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

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

`Primary Key R(A,B):`

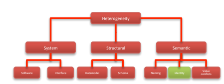
$$\forall x, y, z : R(x, y) \wedge R(x, z) \rightarrow y = z$$

`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(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

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

ILLINOIS INSTITUTE OF TECHNOLOGY

- A rules stands for: for each binding of variables to values from the database so that all the relational atoms R1 to Rn 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)

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

54 CS520 - 1) Introduction

1.4 Datalog

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

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

55 CS520 - 1) Introduction

1.4 Datalog

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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., <

56 CS520 - 1) Introduction

1.4 Datalog

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

57 CS520 - 1) Introduction

1.4 Datalog

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).

58 CS520 - 1) Introduction

1.4 Datalog

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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

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

59 CS520 - 1) Introduction

1.4 Datalog

Example

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

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

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

60 CS520 - 1) Introduction

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'

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

61 CS520 - 1) Introduction

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**

62 CS520 - 1) Introduction

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).

63 CS520 - 1) Introduction

1.4 Containment and Equiv.

Example

Relation **hops(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).

64 CS520 - 1) Introduction

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

65 CS520 - 1) Introduction

1.4 Complexity of Eq. and Cont.

ILLINOIS INSTITUTE OF TECHNOLOGY

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

66 CS520 - 1) Introduction

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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:
  - **head(Q)** = variables in head of query Q
  - **body(Q)** = atoms in body of Q
  - **vars(Q)** = all variable in Q

67 CS520 - 1) Introduction

1.4 Boolean Conjunctive Queries

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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 zero attributes is returned **{()}**
  - -> We interpret {} as **false** and {()} as **true**
  - Boolean query is essentially an existential check

68 CS520 - 1) Introduction

1.4 Boolean Conjunctive Queries

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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
```

69 CS520 - 1) Introduction

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
```

70 CS520 - 1) Introduction

1.4 Boolean Conjunctive Queries

ILLINOIS INSTITUTE OF TECHNOLOGY

- BCQ in SQL

Example

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

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

71 CS520 - 1) Introduction

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 so that:

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

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

72 CS520 - 1) Introduction

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).

Can we find a containment mapping?

73 CS520 - 1) Introduction

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 -> Q_2 :Ψ(u)=x, Ψ(z)=y
Q_2 -> Q_1 :Ψ(x)=u, Ψ(y)=z

74 CS520 - 1) Introduction

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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

75 CS520 - 1) Introduction

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

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

76 CS520 - 1) Introduction

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

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)=y
Q_2 -> Q_1 :Ψ(x)=a, Ψ(y)=b

77 CS520 - 1) Introduction

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

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

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

Q1(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

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

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

Q2(I^Q1)={()}
    φ(x)=a, φ(y)=b

φ is our containment mapping Ψ
```


1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

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

If there exists tuples
     R(a,b) and R(c,b)
in R that make Q1 true, then we take
     R(a,b)
to fulfill Q2
```


1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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 R(a,b)
  - -> 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)
Q1 returns (a) and for every
     R(a,b)
Q2 returns (a)
```


1.4 Similarity Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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]

90 CS520 - 1) Introduction

1.4 Similarity Measures

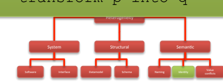
ILLINOIS INSTITUTE OF TECHNOLOGY

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 single character insertions, deletions, replacements to transform p into q
strings



91 CS520 - 1) Introduction

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)

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

92 CS520 - 1) Introduction

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)

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

93 CS520 - 1) Introduction

1.4 Similarity Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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**

94 CS520 - 1) Introduction

Summary

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

95 CS520 - 1) Introduction

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

ILLINOIS INSTITUTE OF TECHNOLOGY

CS520 Data Integration, Warehousing, and Provenance

2. Data Preparation and Cleaning

IIT DBGroup

Boris Glavic
http://www.cs.iit.edu/~glavic/
http://www.cs.iit.edu/~cs520/
http://www.cs.iit.edu/~dbgroup/

Outline

2. Overview

- Topics covered in this part
  - **Causes of Dirty Data**
  - Constraint-based Cleaning
  - Outlier-based and Statistical Methods
  - Entity Resolution
  - Data Fusion

2. Causes of "Dirty" Data

- Manual data entry or result of erroneous integration
  - Typos:
    - **"Peter" vs. "Pteer"**
  - Switching fields
    - "**FirstName**: New York, **City**: Peter"
  - Incorrect information
    - "**City**:New York, **Zip**: 60616"
  - Missing information
    - "**City**: New York, **Zip**: "

2. Causes of "Dirty" Data

- Manual data entry or result of erroneous integration (cont.)
  - Redundancy:
    - (**ID**:1, **City**: Chicago, **Zip**: 60616)
    - (**ID**:2, **City**: Chicago, **Zip**: 60616)
  - Inconsistent references to entities
    - Dept. of Energy, DOE, Dep. Of Energy, …

2. Cleaning Methods

- Enforce Standards
  - Applied in real world
  - How to develop a standard not a fit for this lecture
  - Still relies on no human errors
- Constraint-based cleaning
  - Define constraints for data
  - "Make" data fit the constraints
- Statistical techniques
  - Find outliers and smoothen or remove
    - E.g., use a clustering algorithm

2. Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- Topics covered in this part
  - Causes of Dirty Data
  - **Constraint-based Cleaning**
  - Outlier-based and Statistical Methods
  - Entity Resolution
  - Data Fusion

2.1 Cleaning Methods

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Constraint-based cleaning**
  - Choice of constraint language
  - Detecting violations to constraints
  - Fixing violations (automatically?)

2.1 Constraint Languages

ILLINOIS INSTITUTE OF TECHNOLOGY

- First work focused on functional dependencies (FDs)
- Extensions of FDs have been proposed to allow rules that cannot be expressed with FDs
  - E.g., conditional FDs only enforce the FD is a condition is met
    - -> finer grained control, e.g., zip -> city only if country is US
- Constraints that consider master data
  - Master data is highly reliable data such as a government issued zip, city lookup table

2.1 Constraint Languages (cont.)

ILLINOIS INSTITUTE OF TECHNOLOGY

- Denial constraints
  - Generalize most other proposed constraints
  - State what should not be true
  - Negated conjunction of relational and comparison atoms

$$\forall \vec{x} : \neg(\phi(\vec{x}))$$

- Here we will look at FDs mainly and a bit at denial constraints
  - Sometimes use logic based notation introduced previously

2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraints Languages

SSN	zip	city	name	boss	salary
333-333-3333	60616	New York	Peter	Gert	50,000
333-333-9999	60615	Chicago	Gert	NULL	40,000
333-333-5599	60615	Schaumburg	Gertrud	Hans	10,000
333-333-6666	60616	Chicago	Hans	NULL	1,000,000
333-355-4343	60616	Chicago	Malcom	Hans	20,000

C_1: The zip code uniquely determines the city

C_2: Nobody should earn more than their direct superior

C_3: Salaries are non-negative

2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraints Languages

SSN	zip	city	name	boss	salary
333-333-3333	60616	New York	Peter	Gert	50,000
333-333-9999	60615	Chicago	Gert	NULL	40,000
333-333-5599	60615	Schaumburg	Gertrud	Hans	10,000
333-333-6666	60616	Chicago	Hans	NULL	1,000,000
333-355-4343	60616	Chicago	Malcom	Hans	20,000

C_1: The zip code uniquely determines the city
- expressible as functional dependency

C_2: Nobody should earn more than their direct superior
- e.g., denial constraint

C_3: Salaries are non-negative
- e.g., denial constraint

2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraints Languages

SSN	zip	city	name	boss	salary
333-333-3333	60616	New York	Peter	Gert	50,000
333-333-9999	60615	Chicago	Gert	NULL	40,000
333-333-5599	60615	Schaumburg	Gertrud	Hans	10,000
333-333-6666	60616	Chicago	Hans	NULL	1,000,000
333-355-4343	60616	Chicago	Malcom	Hans	20,000

C_1: The zip code uniquely determines the city
FD_1: zip -> city

$$\forall \neg (E(x,y,z,u,v,w) \wedge E(x',y',z',u',v',w') \wedge x = x' \wedge y \neq y')$$

C_2: Nobody should earn more than their direct superior

$$\forall \neg (E(x,y,z,u,v,w) \wedge E(x',y',z',u',v',w') \wedge v = u' \wedge w > w')$$

C_3: Salaries are non-negative

$$\forall \neg (E(x,y,z,u,v,w) \wedge w < 0)$$

12 CS520 - 1) Introduction

2.1 Constraint based Cleaning Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- Define constraints
- Given database D
  - **1)** Detect violations of constraints
    - We already saw example of how this can be done using queries. Here a bit more formal
  - **2)** Fix violations
    - In most cases there are many different ways to fix the violation by modifying the database (called **solution**)
      - What operations do we allow: insert, delete, update
      - How do we choose between alternative solutions

13 CS520 - 1) Introduction

2.1 Constraint Repair Problem

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Constraint Repair Problem

Given set of constraints **Σ** and an database instance **I** which violates the constraints find a clean instance I' so that I' fulfills **Σ**

- This would allow us to take any I'
  - E.g., empty for FD constraints
- We do not want to loose the information in I (unless we have to)
- Let us come back to that later

14 CS520 - 1) Introduction

2.1 Constraint based Cleaning Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- Study 1) + 2) for FDs
- Given database D
  - **1)** Detect violations of constraints
    - We already saw example of how this can be done using queries. Here a bit more formal
  - **2)** Fix violations
    - In most cases there are many different ways to fix the violation by modifying the database (called **solution**)
      - What operations do we allow: insert, delete, update
      - How do we choose between alternative solutions

15 CS520 - 1) Introduction

2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraints

SSN	zip	city	name
333-333-3333	60616	New York	Peter
333-333-9999	60615	Chicago	Gert
333-333-5599	60615	Schaumburg	Gertrud
333-333-6666	60616	Chicago	Hans
333-355-4343	60616	Chicago	Malcom

FD_1: zip -> city

16 CS520 - 1) Introduction

2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Violations

SSN	zip	city	name
333-333-3333	60616	New York	Peter
333-333-9999	60615	Chicago	Gert
333-333-5599	60615	Schaumburg	Gertrud
333-333-6666	60616	Chicago	Hans
333-355-4343	60616	Chicago	Malcom

FD_1: zip -> city

17 CS520 - 1) Introduction

2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Violations

SSN	zip	city	name
333-333-3333	60616	New York	Peter
333-333-9999	60615	Chicago	Gert
333-333-5599	60615	Schaumburg	Gertrud
333-333-6666	60616	Chicago	Hans
333-355-4343	60616	Chicago	Malcom

```
How to repair?

Deletion:
  - remove some conflicting tuples
  - quite destructive

Update:
  - modify values to resolve the conflict
  - equate RHS values (city here)
  - disequate LHS value (zip)
```


2.1 Constraint based Cleaning Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- How to repair?
- **Deletion:**
  - remove some conflicting tuples
  - quite destructive
- **Update:**
  - modify values to resolve the conflict
  - equate RHS values (city here)
  - disequate LHS value (zip)
- **Insertion?**
  - Not for FDs, but e.g., FKs

2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

SSN	zip	city	name
333-333-3333	60616	New York	Peter
333-333-9999	60615	Chicago	Gert
333-333-5599	60615	Schaumburg	Gertrud
333-333-6666	60616	Chicago	Hans
333-355-4343	60616	Chicago	Malcom

```
Deletion:

Delete Chicago or Schaumburg?

Delete New York or the two Chicago tuples?
   - one tuple deleted vs. two tuples deleted
```


2.1 Example Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

SSN	zip	city	name
333-333-3333	60616	New York	Peter
333-333-9999	60615	Chicago	Gert
333-333-5599	60615	Schaumburg	Gertrud
333-333-6666	60616	Chicago	Hans
333-355-4343	60616	Chicago	Malcom

```
Update equate RHS:

Update Chicago->Schaumburg or Schaumburg->Chicago

Update New York->Chicago or Chicago->New York
   - one tuple deleted vs. two cells updated

Update disequate LHS:

Which tuple to update?
What value do we use here?
```


2.1 Constraint based Cleaning Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Principle of minimality**
  - Choose repair that minimally modifies database
  - Motivation: consider the solution that deletes every tuple
- Most update approaches **equate RHS** because there is usually no good way to choose LHS values unless we have **master data**
  - **E.g., update zip to 56423 or 52456 or 22322 …**

2.1 Detecting Violations

ILLINOIS INSTITUTE OF TECHNOLOGY

- Given FD A -> B on R
  - Recall logical representation
  - Forall X, X': R(X) and R(X') and A=A' -> B=B'
  - Only violated if we find two tuples where A=A', but B != B'
  - In datalog
    - Q(): R(X), R(X'), A=A', B!=B'
  - In SQL

```
SELECT EXISTS (SELECT *
               FROM R x, R y
               WHERE A=A' AND B<>B')
```

2.1 Example Constraints

Example: SQL Violation Detection

```
Relation: Person(name,city,zip)

FD1: zip -> city

Violation Detection Query

SELECT EXISTS (SELECT *
            FROM Person x, Person y
            WHERE x.zip = y.zip
                  AND x.city <> y.city)

To know which tuples caused the conflict:

SELECT *
FROM Person x, Person y
WHERE x.zip = y.zip
      AND x.city <> y.city)
```


2.1 Fixing Violations

- Principle of minimality
  - Choose solution that minimally modifies the database
  - Updates:
    - Need a cost model
  - Deletes:
    - Minimal number of deletes

2.1 Constraint Repair Problem

Definition: Constraint Repair Problem (restated)

Given set of constraints **Σ** and an database instance **I** which violates the constraints find a clean instance I' (does not violate the constraints) with cost(I,I') being minimal

- Cost metrics that have been used
  - **Deletion + Insertion**

$$\Delta(I, I') = (I - I') \cup (I' - I)$$

    - S-repair: minimize measure above under set inclusion
    - C-repair: minimize cardinality
  - **Update**
    - Assume distance metric d for attribute values

2.1 Cost Metrics

- **Deletion + Insertion**

$$\Delta(I, I') = (I - I') \cup (I' - I)$$

  - **S-repair**: minimize measure above under set inclusion
  - **C-repair**: minimize cardinality
- **Update**
  - Assume single relation R with uniquely identified tuples
  - Assume distance metric **d** for attribute values
  - **Schema(R)** = attributes in schema of relation **R**
  - **t'** is updated version of tuple **t**
  - Minimize: $\sum_{t \in R} \sum_{A \in Schema(R)} d(t.A, t'.A)$

2.1 Cost Metrics

- **Update**
  - Assume single relation R with uniquely identified tuples
  - Assume distance metric **d** for attribute values
  - **Schema(R)** = attributes in schema of relation **R**
  - **t'** is updated version of tuple **t**
  - Minimize: $\sum_{t \in R} \sum_{A \in Schema(R)} d(t.A, t'.A)$
- We focus on this one
- This is NP-hard
  - Heuristic algorithm

2.1 Naïve FD Repair Algorithm

- **FD Repair Algorithm: 1. Attempt**
  - For each FD **X -> Y** in **Σ** run query to find pairs of tuples that violate the constraint
  - For each pair of tuples **t** and **t'** that violate the constraint
    - update **t.Y** to **t'.Y**
      - choice does not matter because cost is symmetric, right?

2.1 Constraint Repair

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

	SSN	zip	city	name
t_1	333-333-3333	60616	New York	Peter
t_2	333-333-9999	60615	Chicago	Gert
t_3	333-333-5599	60615	Schaumburg	Gertrud
t_4	333-333-6666	60616	Chicago	Hans
t_5	333-355-4343	60616	Chicago	Malcom

t_1 and t_4: set t_1.city = Chicago
t_1 and t_5: set t_1.city = Chicago
t_2 and t_3: set t_2.city = Schaumburg

30 CS520 - 1) Introduction

2.1 Problems with the Algorithm

ILLINOIS INSTITUTE OF TECHNOLOGY

- **FD Repair Algorithm: 1. Attempt**
  - For each FD **X -> Y** in **Σ** run query to find pairs of tuples that violate the constraint
  - For each pair of tuples **t** and **t'** that violate the constraint: t.X = t'.X and t.Y != t'.Y
    - update **t.Y** to **t'.Y**
      - ~~choice does not matter because cost is symmetric, right?~~
  - **Our updates may cause new violations!**

31 CS520 - 1) Introduction

2.1 Constraint Repair

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

	SSN	zip	city	name
t_1	333-333-3333	60616	New York	Peter
t_2	333-333-9999	60615	Chicago	Gert
t_3	333-333-5599	60615	Schaumburg	Gertrud
t_4	333-333-6666	60616	Chicago	Hans
t_5	333-355-4343	60616	Chicago	Malcom

t_4 and t_1: set t_4.city = New York
t_1 and t_5: set t_1.city = Chicago
t_2 and t_3: set t_2.city = Schaumburg

Now t_1 and t_4 and t_4 and t_5 in violation!

32 CS520 - 1) Introduction

2.1 Problems with the Algorithm

ILLINOIS INSTITUTE OF TECHNOLOGY

- **FD Repair Algorithm: 2. Attempt**
  - I' = I
  - 1) For each FD **X -> Y** in **Σ** run query to find pairs of tuples that violate the constraint
  - 2) For each pair of tuples **t** and **t'** that violate the constraint: t.X = t'.X and t.Y != t'.Y
    - update **t.Y** to **t'.Y**
      - ~~choice does not matter because cost is symmetric, right?~~
  - 3) If we changed I' goto 1)

33 CS520 - 1) Introduction

2.1 Problems with the Algorithm

ILLINOIS INSTITUTE OF TECHNOLOGY

- **FD Repair Algorithm: 2. Attempt**
  - I' = I
  - 1) For each FD **X -> Y** in **Σ** run query to find pairs of tuples that violate the constraint
  - 2) For each pair of tuples **t** and **t'** that violate the constraint: t.X = t'.X and t.Y != t'.Y
    - update **t.Y** to **t'.Y**
      - ~~choice does not matter because cost is symmetric, right?~~
  - 3) If we changed I' goto 1)
    - **May never terminate**

34 CS520 - 1) Introduction

2.1 Constraint Repair

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

	SSN	zip	city	name
t_1	333-333-3333	60616	New York	Peter
t_2	333-333-9999	60615	Chicago	Gert
t_3	333-333-5599	60615	Schaumburg	Gertrud
t_4	333-333-6666	60616	Chicago	Hans
t_5	333-355-4343	60616	Chicago	Malcom

t_4 and t_1: set t_4.city = New York
t_1 and t_5: set t_1.city = Chicago

Now t_1 and t_4 and t_4 and t_5 in violation!

t_4 and t_1: set t_1.city = New York
T_5 and t_4: set t_4.city = Chicago

repeat

35 CS520 - 1) Introduction

2.1 Problems with the Algorithm

- **FD Repair Algorithm: 2. Attempt**
  - **Even if we succeed the repair may not be minimal. There may be many tuples with the same X values**
    - **They all have to have the same Y value**
    - **Choice which to update matters!**

2.1 Constraint Repair

Example: Constraint Repair

	SSN	zip	city	name
t_1	333-333-3333	60616	New York	Peter
t_2	333-333-9999	60615	Chicago	Gert
t_3	333-333-5599	60615	Schaumburg	Gertrud
t_4	333-333-6666	60616	Chicago	Hans
t_5	333-355-4343	60616	Chicago	Malcom

Cheaper: t_1.city = Chicago
Not so cheap: set t_4.city and t_5.city = New York

2.1 Problems with the Algorithm

- **FD Repair Algorithm: 3. Attempt**
  - Equivalence Classes
    - Keep track of sets of cells (tuple,attribute) that have to have the same values in the end (e.g., all Y attribute values for tuples with same X attribute value)
    - These classes are updated when we make a choice
    - Choose Y value for equivalence class using minimality, e.g., most common value
  - Observation
    - Equivalence Classes may merge, but never split if we only update RHS of all tuples with same X at once
    - -> we can find an algorithm that terminates

2.1 Problems with the Algorithm

- **FD Repair Algorithm: 3. Attempt**
  - **Initialize**:
    - Each cell in its own equivalence class
    - Put all cells in collection **unresolved**
  - While **unresolved** is not empty
    - Remove tuple t from unresolved
    - Pick FD X->Y (e.g., random)
    - Compute set of tuples S that have same value in X
    - Merge all equivalence classes for all tuples in S and attributes in Y
    - Pick values for Y (update all tuples in S to Y)

2.1 Problems with the Algorithm

- **FD Repair Algorithm: 3. Attempt**
- Algorithm using this idea:
  - More heuristics to improve quality and performance
    - Cost-based pick of next EQ's to merge
  - Also for FKs (Inclusion Constraints)

A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification

2.1 Consistent Query Answering

- As an alternative to fixing the database which requires making a choice we could also leave it dirty and try to resolve conflicts at query time
  - Have to reason over answers to the query without knowing which of the possible repairs will be chosen
  - **Intuition**: return tuples that would be in the query result for **every** possible repair

2.1 Constraint Repair

Example: Constraint Repair

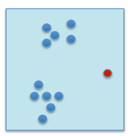
	SSN	zip	city	name
t_1	333-333-3333	60616	New York	Peter
t_2	333-333-9999	60615	Chicago	Gert
t_3	333-333-5599	60615	Schaumburg	Gertrud
t_4	333-333-6666	60616	Chicago	Hans
t_5	333-355-4343	60616	Chicago	Malcom

Cheaper: t_1.city = Chicago
Not so cheap: set t_4.city and t_5.city = New York

2. Overview

- Topics covered in this part
  - Causes of Dirty Data
  - Constraint-based Cleaning
  - **Outlier-based and Statistical Methods**
  - Entity Resolution
  - Data Fusion

2.2 Statistical and Outlier



- Assumption
  - Errors can be identified as outliers
- How do we find outliers?
  - **Similarity-based**:
    - Object is dissimilar to all (many) other objects
    - E.g., clustering, objects not in cluster are outliers
  - **Some type of statistical test**:
    - Given a distribution (e.g., fitted to the data)
    - How probable is it that the point has this value?
    - If low probability -> outlier

2. Overview

- Topics covered in this part
  - Causes of Dirty Data
  - Constraint-based Cleaning
  - Outlier-based and Statistical Methods
  - **Entity Resolution**
  - Data Fusion

2.3 Entity Resolution

- Entity Resolution (ER)
- Alternative names
  - Duplicate detection
  - Record linkage
  - Reference reconciliation
  - Entity matching
  - …

2.3 Entity Resolution

Definition: Entity Resolution Problem

Given sets of tuples **A** compute equivalence relation **E(t,t')** which denotes that tuple **t** and **t'** represent the same entity.

- Intuitively, E should be based on how similar t and t' are
  - Similarity measure?
- E should be an equivalence relation
  - If **t** is the same as **t'** and **t'** is the same as **t''** then **t** should be the same as **t''**

2.3 Constraint Repair

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

SSN	zip	city	name
333-333-3333	60616	Chicago	Peter

SSN	zip	city	name
3333333333	IL 60616		Petre

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- Similarity based on similarity of attribute values
  - Which distance measure is appropriate?
  - How do we combine attribute-level distances?
  - Do we consider additional information?
    - **E.g., foreign key connections**
  - How similar should duplicates be?
    - **E.g., fixed similarity threshold**
  - How to guarantee transitivity of E
    - **E.g., do this afterwards**

2.3 Constraint Repair

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

SSN	zip	city	name
333-333-3333	60616	Chicago	Peter

1 0.8 0? 0.6

SSN	zip	city	name
3333333333	IL 60616		Petre

2.3 Entity Resolution – Distance Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

- Edit-distance
  - measures similarity of two strings
  - d(s,s') = minimal number of insert, replace, delete operations (single character) that transform s into s'
  - Is symmetric (actually a metric)
    - Why?

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Edit Distance

Given two strings s, s' we define the edit distance d(s,s') as the minimum number of single character insert, replacements, deletions that transforms s into s'

Example:

```
NEED -> STREET

Trivial solution: delete all chars in NEED, then
insert all chars in STREET

- gives upper bound on distance len(NEED) +
       len(STREET) = 10
```


2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

```
NEED -> STREET

Minimal solution:
       - insert S
       - insert T
       - replace N with R
       - replace D with T

d(NEED,STREET) = 4
```

2.3 Entity Resolution

- **Principal of optimality**
  - Best solution of a subproblem is part of the best solution for the whole problem
- **Dynamic programming algorithm**
  - D(i,j) is the edit distance between prefix of len i of s and prefix of len j of s'
  - D(len(s),len(s')) is the solution
  - Represented as matrix
  - Populate based on rules shown on the next slide

2.3 Entity Resolution

- **Recursive definition**
  - D(i,0) = i
    - Cheapest way of transforming prefix s[i] into empty string is by deleting all i characters in s[i]
  - D(0,j) = j
    - Same holds for s'[j]
  - D(i,j) = min {
    - D(i-1,j) + 1
    - D(i,j-1) + 1
    - D(i-1,j-1) + d(i,j) with d(i,j) = 1 if s[i] != s[j] and 0 else

    }

2.3 Entity Resolution

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1						
E	2						
E	3						
D	4						

2.3 Entity Resolution

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1	1					
E	2						
E	3						
D	4						

2.3 Entity Resolution

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1	1	2				
E	2	2					
E	3						
D	4						

2.3 Entity Resolution

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1	1	2	3			
E	2	2	2				
E	3	3					
D	4						

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1	1	2	3	4		
E	2	2	2	3			
E	3	3	3				
D	4	4					

60 CS520 - 1) Introduction

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1	1	2	3	4	5	
E	2	2	2	3	3		
E	3	3	3	3			
D	4	4	4				

61 CS520 - 1) Introduction

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1	1	2	3	4	5	6
E	2	2	2	3	3	4	
E	3	3	3	3	3		
D	4	4	4	4			

62 CS520 - 1) Introduction

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

Example:

NEED -> STREET

		S	T	R	E	E	T
	0	1	2	3	4	5	6
N	1	1	2	3	4	5	6
E	2	2	2	3	3	4	5
E	3	3	3	3	3	3	4
D	4	4	4	4	4	4	4

63 CS520 - 1) Introduction

2.3 Entity Resolution – Distance Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

- Other sequence-based measures for string similarity
  - **Needleman-Wunsch**
    - Missing character sequences can be penalized differently from character changes
  - **Affine Gap Measure**
    - Limit influence of longer gaps
    - **E.g., Peter Friedrich Mueller vs. Peter Mueller**
  - **Smith-Waterman Measure**
    - More resistant to reordering of elements in the string
    - **E.g., Prof. Franz Mueller vs. F. Mueller, Prof.**

64 CS520 - 1) Introduction

2.3 Entity Resolution – Distance Measures

ILLINOIS INSTITUTE OF TECHNOLOGY

- Other sequence-based measures for string similarity
  - **Jaro-Winkler**
    - Consider shared prefixes
    - Consider distance of same characters in strings
    - **E.g., johann vs. ojhann vs. ohannj**
  - **See textbook for details!**

65 CS520 - 1) Introduction

2.3 Entity Resolution – Distance Measures

- **Token-set based measures**
  - Split string into tokens
    - E.g., single characters
    - E.g., words if string represents a longer text
  - Potentially normalize tokens
    - **E.g., word tokens replace word with its stem**
      - **Generating, generated, generates are all replaced with generate**
  - Represent string as set (multi-set) of tokens

66 CS520 - 1) Introduction

2.3 Entity Resolution

Example: Tokenization

```
Input string:
S = "the tokenization of strings is commonly used in
information retrieval"

Set of tokens:
Tok(S) = {commonly, in, information, is, of,
          retrieval, strings, the, tokenization, used}

Bag of tokens:
Tok(S) = {commonly:1, in:1, information:1, is:1,
         of:1, retrieval:1,strings:1, the:1,
         tokenization:1, used:1}
```

67 CS520 - 1) Introduction

2.3 Entity Resolution – Distance Measures

- **Jaccard-Measure**
  - $\mathbf{B_s}$ = **Tok(s)** = token set of string **s**
  - Jaccard measures relative overlap of tokens in two strings
    - Number of common tokens divided by total number of tokens

$$d_{jacc}(s, s') = \frac{\|B_s \cap B_{s'}\|}{\|B_s \cup B_{s'}\|}$$

68 CS520 - 1) Introduction

2.3 Entity Resolution

Example: Tokenization

```
Input string:
S = "nanotubes are used in these experiments to…"
S'= "we consider nanotubes in our experiments…"
S''= "we prove that P=NP, thus solving …"

Tok(S)   = {are,experiments,in,nanotubes,these,to,used}
Tok(S')  = {consider,experiments,in,nanotubes,our,we}
Tok(S'') = {P=NP,prove,solving,that,thus,we}

djacc(S,S')=
djacc(S,S'')=
djacc(S',S'')=
```

69 CS520 - 1) Introduction

2.3 Entity Resolution

Example: Tokenization

```
Input string:
S = "nanotubes are used in these experiments to…"
S'= "we consider nanotubes in our experiments…"
S''= "we prove that P=NP, thus solving …"

Tok(S)   = {are,experiments,in,nanotubes,these,to,used}
Tok(S')  = {consider,experiments,in,nanotubes,our,we}
Tok(S'') = {P=NP,prove,solving,that,thus,we}

djacc(S,S')   = 3 / 10 = 0.3
djacc(S,S'')  = 0 / 13 = 0
djacc(S',S'') = 1 / 11 = 0.0909
```

70 CS520 - 1) Introduction

2.3 Entity Resolution

- **Other set-based measures**
  - **TF/IDF**: term frequency, inverse document frequency
    - Take into account that certain tokens are more common than others
    - If two strings (called documents for TF/IDF) overlap on uncommon terms they are more likely to be similar than if they overlap on common terms
      - **E.g., the vs. carbon nanotube structure**

71 CS520 - 1) Introduction

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **TF/IDF**: term frequency, inverse document frequency
  - Represent documents as feature vectors
    - One dimension for each term
    - Value computed as frequency times IDF
      - Inverse of frequency of term in the set of all documents
  - Compute cosine similarity between two feature vectors
    - Measure how similar they are in term distribution (weighted by how uncommon terms are)
    - Size of the documents does not matter
  - **See textbook for details**

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Entity resolution**
  - Concatenate attribute values of tuples and use string similarity measure
    - Loose information encoded by tuple structure
    - **E.g., [Gender:male,Salary:9000]**
      - **-> "Gender:male,Salary:9000"**
      - **or -> "male,9000"**
  - Combine distance measures for single attributes
    - Weighted sum or more complex combinations
      - E.g., $d(t,t') = w_1 \times d_A(t.A, t'.A) + w_2 \times d_B(t.B, t'.B)$
  - Use quadratic distance measure
    - E.g., earth-movers distance

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Entity resolution**
  - Rule-based approach
    - Set of **if this than that** rules
  - Learning-based approaches
  - Clustering-based approaches
  - Probabilistic approaches to matching
  - Collective matching

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Weighted linear combination**
  - Say tuples have **n** attributes
  - $\mathbf{w_i}$: predetermined weight of an attribute
  - $\mathbf{d_i(t,t')}$: similarity measure for the **i^th** attribute

$$d(t,t') = \sum_{i=0}^{n} w_i \times d_i(t,t')$$

- Tuples match if **d(t,t') > β** for a threshold **β**

2.3 Constraint Repair

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Constraint Repair

SSN	zip	city	name
333-333-3333	60616	Chicago	Peter

1 ⇕ 0.8 ⇕ 0? ⇕ 0.6 ⇕

SSN	zip	city	name
3333333333	IL 60616		Petre

Assumption: SSNs and names are most important, city and zip are not very predictive

$$w_{SSN} = 0.4, w_{zip} = 0.05, w_{city} = 0.15, w_{name} = 0.4$$

$$d(t,t') = 0.4 \times 1 + 0.05 \times 0.8 + 0.15 \times 0 + 0.4 \times 0.6$$
$$= 0.4 + 0.04 + 0 + 0.24$$
$$= 0.68$$

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Weighted linear combination**
  - How to determine weights?
    - **E.g., have labeled training data and use ML to learn weights**
  - Use non-linear function?

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Entity resolution**
  - **Rule-based approach**
  - Learning-based approaches
  - Clustering-based approaches
  - Probabilistic approaches to matching
  - Collective matching

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Rule-based approach**
  - Collection (list) of rules
  - **if** $d_{name}(t,t') < 0.6$ **then** unmatched
  - **if** $d_{zip}(t,t') = 1$ **and** t.country = USA **then** matched
  - **if** t.country != t'.country **then** unmatched
- **Advantages**
  - Easy to start, can be incrementally improved
- **Disadvantages**
  - Lot of manual work, large rule-bases hard to understand

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Entity resolution**
  - Rule-based approach
  - **Learning-based approaches**
  - Clustering-based approaches
  - Probabilistic approaches to matching
  - Collective matching

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Learning-based approach**
  - Build all pairs (t,t') for training dataset
  - Represent each pair as feature vector from, e.g., similarities
  - Train classifier to return {match,no match}
- **Advantages**
  - automated
- **Disadvantages**
  - Requires training data

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Entity resolution**
  - Rule-based approach
  - Learning-based approaches
  - **Clustering-based approaches**
  - Probabilistic approaches to matching
  - Collective matching

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clustering-based approach**
  - Apply clustering method to group inputs
  - Typically hierarchical clustering method
  - Clusters now represent entities
    - Decide how to merge based on similarity between clusters
- **Advantages**
  - Automated, no training data required
- **Disadvantages**
  - Choice of cluster similarity critical

2.3 Entity Resolution

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Entity resolution**
  - Rule-based approach
  - Learning-based approaches
  - Clustering-based approaches
  - **Probabilistic approaches to matching**
  - **Collective matching**
    - **See text book**

2. Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- Topics covered in this part
  - Causes of Dirty Data
  - Constraint-based Cleaning
  - Outlier-based and Statistical Methods
  - Entity Resolution
  - **Data Fusion**

2.4 Data Fusion

ILLINOIS INSTITUTE OF TECHNOLOGY

- Data Fusion = how to combine (possibly conflicting) information from multiple objects representing the same entity
  - Choose among conflicting values
    - If one value is missing (NULL) choose the other one
    - Numerical data: e.g., median, average
    - Consider sources: have more trust in certain data sources
    - Consider value frequency: take most frequent value
    - Timeliness: latest value

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

CS520 Data Integration, Warehousing, and Provenance

3. Schema Matching and Mapping

IIT DBGroup

Boris Glavic
http://www.cs.iit.edu/~glavic/
http://www.cs.iit.edu/~cs520/
http://www.cs.iit.edu/~dbgroup/

Outline

3. Why matching and mapping?

- **Problem: Schema Heterogeneity**
  - Sources with different schemas store overlapping information
  - Want to be able to translate data from one schema into a different schema
    - Datawarehousing
    - Data exchange
  - Want to be able to translate queries against one schema into queries against another schema
    - Virtual dataintegration

3. Why matching and mapping?

- **Problem: Schema Heterogeneity**
  - We need to know how elements of different schemas are related!
  - **Schema matching**
    - Simple relationships **such as attribute name of relation person in the one schema corresponds to attribute lastname of relation employee in the other schema**
  - **Schema mapping**
    - Also model correlations and missing information such as links caused by foreign key constraints

3. Why matching and mapping?

- **Why both mapping and matching**
  - Split complex problem into simpler subproblems
    - Determine matches and then correlate with constraint information into mappings
  - Some tasks only require matches
    - E.g., matches can be used to determine attributes storing the same information in data fusion
  - Mappings are naturally an generalization of matchings

3. Overview

- Topics covered in this part
  - **Schema Matching**
  - Schema Mappings and Mapping Languages

3.1 Schema Matching

- **Problem: Schema Matching**
  - Given two (or more schemas)
    - For now called **source** and **target**
  - Determine how elements are related
    - Attributes are representing the same information
      - **name = lastname**
    - Attribute can be translated into an attribute
      - **MonthlySalary * 12 = Yearly Salary**
    - **1-1** matches vs. **M-N** matches
      - **name** to **lastname**
      - **name** to **concat(firstname, lastname)**

6 CS520 - 3) Matching and Mapping

3.1 Schema Matching

- **Why is this hard?**
  - **Insufficient information**: schema does not capture full semantics of a domain
  - **Schemas can be misleading**:
    - E,g., attributes are not necessarily descriptive
    - E.g., finding the right way to translate attributes not obvious

7 CS520 - 3) Matching and Mapping

3.1 Schema Matching

- **What information to consider?**
  - Attribute names
    - or more generally element names
  - Structure
    - e.g., belonging to the same relation
  - Data
    - Not always available
- **Need to consider multiple types to get reasonable matching quality**
  - Single types of information not predictable enough

8 CS520 - 3) Matching and Mapping

3.1 Schema Mapping

Example: Types of Matching

Name	Address
Peter	1
Alice	3
Bob	3

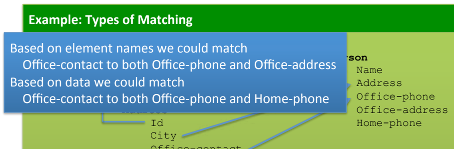
Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	Chicago, IL 60655	(333) 323 3344
Alice	Chicago	(312) 555 7777	Chicago, IL 60633	(123) 323 3344
Bob	New York	(465) 123 1234	New York, NY 55443	(888) 323 3344

9 CS520 - 3) Matching and Mapping

3.1 Schema Mapping

Example: Types of Matching



Name	Address
Peter	1
Alice	3
Bob	3

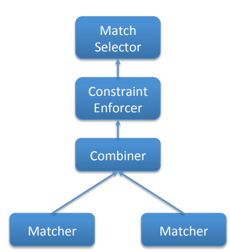
Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	Chicago, IL 60655	(333) 323 3344
Alice	Chicago	(312) 555 7777	Chicago, IL 60633	(123) 323 3344
Bob	New York	(465) 123 1234	New York, NY 55443	(888) 323 3344

10 CS520 - 3) Matching and Mapping

3.1 Schema Matching

- **Typical Matching System Architecture**



Determine actual matches

Use constraints to modify similarity matrix

Combine individual similarity matrices

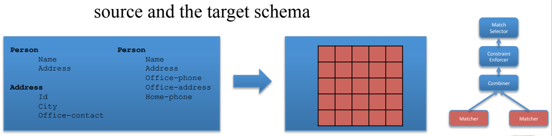
Each matcher uses one type of information to compute similarity matrix

11 CS520 - 3) Matching and Mapping

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Matcher**
  - **Input:** Schemas
    - Maybe also data, documentation
  - **Output**: Similarity matrix
    - Storing value [0,1] for each pair of elements from the source and the target schema



3.1 Schema Matching

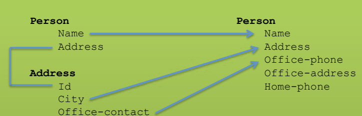
ILLINOIS INSTITUTE OF TECHNOLOGY

- **Name-Based Matchers**
  - String similarities measures
    - E.g., Jaccard and other measure we have discussed
  - Preprocessing
    - Tokenization?
    - Normalization
      - Expand abbreviations and replace synonyms
    - Remove stop words
      - In, and, the

3.1 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching



	Name	Address	Office-phone	Office-address	Home-phone
Name	1	0	0	0	0
Address	0	1	0	0.4	0
Id	0	0	0	0	0
City	0	0	0	0	0
Office-contact	0	0	0.5	0.5	0

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Data-Based Matchers**
  - Determine how similar the values of two attributes are
  - Some techniques
    - Recognizers
      - Dictionaries, regular expressions, rules
    - Overlap matcher
      - Compute overlap of values in the two attributes
    - Classifiers

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Recognizers**
  - Dictionaries
    - Countries, states, person names
  - Regular expression matchers
    - **Phone numbers:** `(\+\d{2})? \(\d{3}\) \d{3} \d{4}`

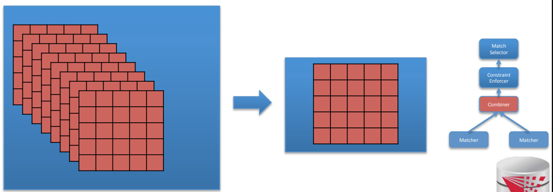
3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Overlap of attribute domains**
  - Each attribute value is a token
  - Use set-based similarity measure such as Jaccard
- **Classifier**
  - Train classifier to identify values of one attribute **A** from the source
    - Training set are values from **A** as positive examples and values of other attributes as negative examples
  - Apply classifier to all values of attributes from target schema
    - Aggregate into similarity score

3.1 Schema Matching

- **Combiner**
  - **Input:** Similarity matrices
    - Output of the individual matchers
  - **Output**: Single Similarity matrix

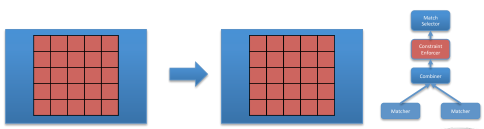


3.1 Schema Matching

- **Combiner**
  - Merge similarity matrices produced by the matchers into single matrix
  - Typical strategies
    - Average, Minimum, Max
    - Weighted combinations
    - Some script

3.1 Schema Matching

- **Constraint Enforcer**
  - **Input:** Similarity matrix
    - Output of Combiner
  - **Output**: Similarity matrix



3.1 Schema Matching

- **Constraint Enforcer**
  - Determine most probably match by assigning each attribute from source to one target attribute
    - Multiple similarity scores to get likelihood of match combination to be true
  - Encode domain knowledge into constraints
    - **Hard constraints**: Only consider match combinations that fulfill constraints
    - **Soft constraints**: violating constraints results in penalty of scores
      - Assign cost for each constraint
  - Return combination that has the maximal score

3.1 Schema Matching

Example: Constraints

Constraint 1: An attribute matched to **source.cust-phone** has to get a score of 1 from the phone regexpr matcher

Constraint 2: Any attribute matched to **source.fax** has to have fax in its name

Constraint 3: If an attribute is matched to **source.firstname** with score > 0.9 then there has to be another attribute from the same target table that is matched to **source.lastname** with score > 0.9

3.1 Schema Matching

- **How to search match combinations**
  - Full search
    - Exponentially many combinations potentially
  - Informed search approaches
    - A* search
  - Local propagation
    - Only local optimizations

3.1 Schema Matching

- **A* search**
  - Given a search problem
    - Set of states: start state, goal states
    - Transitions about states
    - Costs associated with transitions
    - Find cheapest path from start to goal states
  - Need admissible heuristics **h**
    - For a path **p**, **h** computes lower bound for any path from start to goal with prefix **p**
  - Backtracking best-first search
    - Choose next state with lowest estimated cost
    - Expand it in all possible ways

3.1 Schema Matching

- **A* search**
  - Estimated cost of a state **f(n) = g(n) + h(n)**
    - **g(n)** = cost of path from start state to **n**
    - **h(n)** = lower bound for path from **n** to goal state
  - No path reaching the goal state from **n** can have a total cost lower than **f(n)**

3.1 Schema Matching

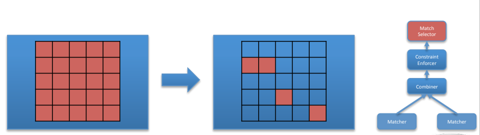
- **Algorithm**
  - Data structures
    - Keep a priority queue **q** of states sorted on f(n)
      - Initialize with start state
    - Keep set **v** of already visited nodes
      - Initially empty
  - While **q** is not empty
    - pop state **s** from head of **q**
    - If **s** is goal state return
    - Foreach **s'** that is direct neighbor of **s**
      - If **s'** not in **v**
      - Compute **f(s')** and insert **s'** into **q**

3.1 Schema Matching

- **Application to constraint enforcing**
  - Source attributes: A_1 to A_n
  - Target attributes: B_1 to B_m
  - States
    - Vector of length n with values B_i or * indicating that no choice has not been taken
    - [B_1, *, *, B_3]
  - Initial state
    - [*, *, *, *]
  - Goal states
    - All states without *

3.1 Schema Matching

- **Match Selector**
  - **Input:** Similarity matrix
    - Output of the individual matchers
  - **Output**: Matches



3.1 Schema Matching

- **Match Selection**
  - Merge similarity matrices produced by the matchers into single matrix
  - Typical strategies
    - Average, Minimum, Max
    - Weighted combinations
    - Some script

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Many-to-many matchers**
  - Combine multiple columns using a set of functions
    - **E.g., concat, +, currency exchange, unit exchange**
  - Large or even unlimited search space
  - -> need method that explores interesting part of the search space
  - Specific searchers
    - Only concatenation of columns (limit number of combinations, e.g., 2)

30 CS520 - 3) Matching and Mapping

3. Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

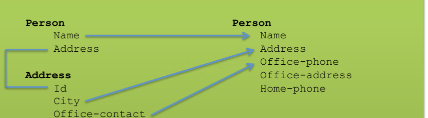
- Topics covered in this part
  - Schema Matching
  - **Schema Mappings and Mapping Languages**

31 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Matching Result



Name	Address
Peter	1
Alice	3
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

Assume: We have data in the source as shown above

What data should we create in the target? Copy values based on matches?

32 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

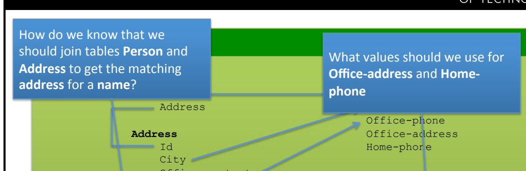
ILLINOIS INSTITUTE OF TECHNOLOGY

- Matches do not determine completely how to create the target instance data! (**Data Exchange**)
  - How do we choose values for attributes that do not have a match?
  - How do we combine data from different source tables?
- Matches do not determine completely what the answers to queries over a mediated schema should be! (**Virtual Data Integration**)

33 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY



Name	Address
Peter	1
Alice	3
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343		
Alice	Chicago	(312) 555 7777		
Bob	New York	(465) 123 1234		

34 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Schema mappings**
  - Generalize matches
  - Describe relationship between instances of schemas
  - Mapping languages
    - LAV, GAV, GLAV
    - Mapping as Dependencies: tuple-generating dependencies
- **Mapping generation**
  - **Input**: Matches, Schema constraints
  - **Output**: Schema mappings

35 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

- **Instance-based definition of mappings**
  - Global schema **G**
  - Local schemas $\mathbf{S_1}$ to $\mathbf{S_n}$
  - Mapping **M** can be expressed as for each set of instances of the local schemas what are allowed instances of the global schema
    - Subset of $(\mathbf{I_G} \times \mathbf{I_1} \times \ldots \times \mathbf{I_n})$
  - Useful as a different way to think about mappings, but not a practical way to define mappings

3.2 Schema Mapping

- **Certain answers**
  - Given mapping **M** and **Q**
  - Instances $\mathbf{I_1}$ to $\mathbf{I_n}$ for $\mathbf{S_1}$ to $\mathbf{S_n}$
  - Tuple **t** is a certain answer for **Q** over $\mathbf{I_1}$ to $\mathbf{I_n}$
    - If for every instance $\mathbf{I_G}$ so that $(\mathbf{I_G} \times \mathbf{I_1} \times \ldots \times \mathbf{I_n})$ in **M** then **t** in $\mathbf{Q(I_G)}$

3.2 Schema Mapping

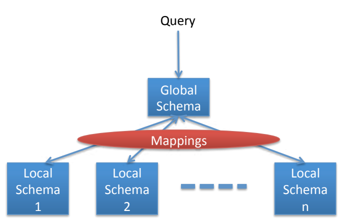
- **Languages for Specifying Mappings**
- **Describing mappings as inclusion relationships between views:**
  - Global as View (**GAV**)
  - Local as View (**LAV**)
  - Global and Local as View (**GLAV**)
- **Describing mappings as dependencies**
  - Source-to-target tuple-generating dependencies (**st-tgds**)

3.2 Schema Mapping

- **Describing mappings as inclusion relationships between views:**
  - Global as View (**GAV**)
  - Local as View (**LAV**)
  - Global and Local as View (**GLAV**)
- Terminology stems from virtual integration
  - Given a **global** (or mediated, or virtual) schema
  - A set of data sources (**local** schemas)
  - Compute answers to queries written against the global schema using the local data sources

3.2 Schema Mapping

- **Excursion Virtual Data Integration**
  - More in next section of the course



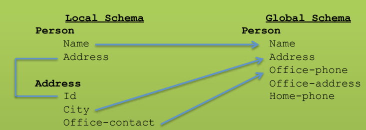
3.2 Schema Mapping

- **Global-as-view (GAV)**
  - Express the global schema as views over the local schemata
  - What query language do we support?
    - CQ, UCQ, SQL, …?
  - **Closed vs. open world** assumption
    - Closed world: $R = Q(S_1,\ldots,S_n)$
      - Content of global relation R is defined as the result of query Q over the sources
    - Open world: $R \supseteq Q(S_1,\ldots,S_n)$
      - Relation R has to contain the result of query Q, but may contain additional tuples

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching



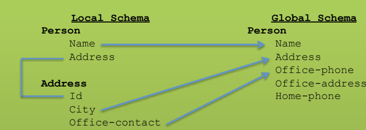

```
Person(X',Y',Z',A',B')
= Q(X,Z,A, NULL, NULL) :- Person(X,Y), Address(Y,Z,A)

Since heads of LHS and RHS have to be the same we can use
simpler notation without the head of the view expression:

Person(X,Z,A, NULL, NULL) = Person(X,Y), Address(Y,Z,A)
```

42 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching




```
Consider switching local and global schema

Person(X,NULL) = Person(X,Y,Z,A,B)
Address(NULL,Y,Z) = Person(X,Y,Z,A,B)
```

43 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-as-view (GAV)**
- **Solutions (mapping M)**
  - Unique solutions (1 solution!)
  - Intuitively, execute queries over local instance that produced global instance

44 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-as-view (GAV)**
- **Answering Queries**
  - Simply replace references to global tables with the view definition
- Mapping **R(X,Y) = S(X,Y), T(Y,Z)**
- **Q(X) :- R(X,Y)**
- Rewrite into
- **Q(X) :- S(X,Y), T(Y,Z)**

45 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-as-view (GAV) Discussion**
  - Hard to add new source
    - -> have to rewrite the view definitions
  - Does not deal gracefully with missing values
  - Easy query processing
    - -> view unfolding

46 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

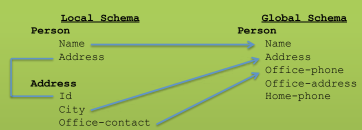
- **Local-as-view (LAV)**
  - Express the local schema as views over the global schemata
  - What query language do we support?
    - CQ, UCQ, SQL, …?
  - **Closed vs. open world** assumption
    - Closed world: $S_{ij} = Q(G)$
      - Content of local relation S_{ij} is defined as the result of query Q over the sources
    - Open world: $S_{ij} \supseteq Q(G)$
      - Local relation S_{ij} has to contain the result of query Q, but may contain additional tuples

47 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching




```
Person(X,NULL) = Person(X,Y,Z,A,B)
Address(NULL,Y,Z) = Person(X,Y,Z,A,B)
```


3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Local-as-view (LAV)**
- **Solutions (mapping M)**
  - May be many solutions

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Local-as-view (GAV)**
- **Answering Queries**
  - Need to find equivalent query using only the views (this is a hard problem, more in next course section)
- Mapping **S(X,Z) = R(X,Y), T(Y,Z)**
- **Q(X) :- R(X,Y)**
- Rewrite into ???
  - Need to come up with missing values
  - Give up query equivalence?

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Local-as-view (LAV) Discussion**
  - Easy to add new sources
    - -> have to write a new view definition
    - May take some time to get used to expressing sources like that
  - Still does not deal gracefully with all cases of missing values
    - Loosing correlation
  - Hard query processing
    - Equivalent rewriting using views only
    - Later: give up equivalence

3.2 Schema Mapping

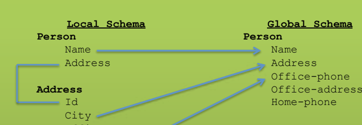
ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-Local-as-view (GLAV)**
  - Express both sides of the constraint as queries
  - What query language do we support?
    - CQ, UCQ, SQL, …?
  - **Closed vs. open world** assumption
    - Closed world: Q'(G) = Q(S)
    - Open world: Q'(G) ⊇Q(S)

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching




```
Source: Q(X,Y,Z) :- Person(X',Y'), Address(Y',Z',A')
=
Target: Q(X',Y',Z') :- Person(X',Y',Z',A',B')
```

3.2 Schema Mapping

- **Local-as-view (GLAV) Discussion**
  - Kind of best of both worlds (almost)
  - Complexity of query answering is the same as for LAV
  - Can address the lost correlation and missing values problems we observed using GAV and LAV

3.2 Schema Mapping

- **Source-to-target tuple-generating dependencies (st-tgds)**
  - Local way of expressing GLAV mappings

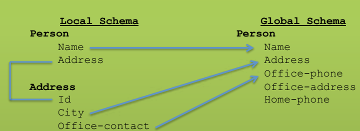
$$\forall \vec{x} : \phi(\vec{x}) \rightarrow \exists \vec{y} : \psi(\vec{x}, \vec{y})$$

  - Equivalence to a containment constraint:

    Q'(G) ⊇ Q(S)

3.2 Schema Mapping

Example: Types of Matching



$$\forall x, y, z, a : Person(x, y) \wedge Address(y, z, a) \rightarrow \exists b, c : Person(x, z, a, b, c)$$

```
Source: Q(X,Y,Z) :- Person(X',Y'), Address(Y',Z',A')
=
Target: Q(X',Y',Z') :- Person(X',Y',Z',A',B')
```


3.2 Schema Mapping

- **Generating Schema Mappings**
  - **Input**: Schemas (Constraints), matches
  - **Output**: Schema mappings
- Ideas:
  - Schema matches tell us which source attributes should be copied to which target attributes
  - Foreign key constraints tell us how to join in the source and target to not loose information

3.2 Schema Mapping

- **Clio**
  - Clio is a data exchange system prototype developed by IBM and University of Toronto researchers
  - The concepts developed for Clio have been implemented in IBM InfoSphere Data Architect
  - Clio does matching, mapping generation, and data exchange
    - For now let us focus on the mapping generation

3.2 Schema Mapping

- **Clio Mapping Generation Algorithm**
  - **Inputs**: Source and Target schemas, matches
  - **Output**: Mapping from source to target schema
  - Note, Clio works for nested schemas such as XML too not just for relational data.
    - Here we will look at the relational model part only

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clio Algorithm Steps**
  - 1) Use **foreign keys** to determine all reasonable ways of **joining** data within the source and the target schema
    - Each alternative of joining tables in the source/target is called a logical association
  - 2) For each pair of **source-target logical associations**: Correlate this information with the matches to determine candidate mappings

60 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clio Algorithm: 1) Find logical associations**
  - This part relies on the **chase** procedure that first introduced to test implication of functional dependencies ('77)
  - The idea is that we start use a representation of foreign keys are **inclusion dependencies** (tgds)
    - There are also chase procedures that consider **edgs** (e.g., PKs)
  - Starting point are all single relational atoms
    - E.g., R(X,Y)

61 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Chase step**
  - Works on **tabelau**: set of relational atoms
  - A chase step takes one tgd t where the LHS is fulfilled and the RHS is not fulfilled
    - We fulfill the tgd t by adding new atoms to the tableau and mapping variables from t to the actually occuring variables from the current tablau
- **Chase**
  - Applying the chase until no more changes
  - Note: if there are cyclic constraints this may not terminate

62 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clio Algorithm: 1) Find logical associations**
  - Compute chase R(X) for each atom R in source and target
  - Each chase result is a logical association
  - Intuitively, each such logical association is a possible way to join relations in a schema based on the FK constraints

63 CS520 - 3) Matching and Mapping

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clio Algorithm: 2) Generate Candidate Mappings**
  - For each pair of logical association $\mathbf{A_S}$ in the source and $\mathbf{A_T}$ in the target produced in step 1
  - Find the matches that are covered by $\mathbf{A_S}$ and $\mathbf{A_T}$
    - Matches that lead from an element of $\mathbf{A_S}$ to an element from $\mathbf{A_T}$
  - If there is at least one such match then create mapping by equating variables as indicated by the matches and create st-tgd with $\mathbf{A_S}$ in LHS and $\mathbf{A_T}$ in RHS

64 CS520 - 3) Matching and Mapping

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

65 CS520 - 3) Matching and Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

CS520 Data Integration, Warehousing, and Provenance

5. Data Exchange

IIT DBGroup

Boris Glavic
http://www.cs.iit.edu/~glavic/
http://www.cs.iit.edu/~cs520/
http://www.cs.iit.edu/~dbgroup/

Outline

5. Data Exchange

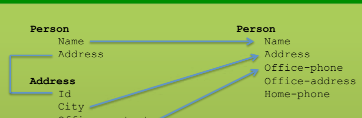
- **Virtual Data Integration**
  - Never materialize instances for the global schema
  - Data of global schema only "visible" through queries
- **Data Exchange**
  - Materialize instance of global instance
    - We call it the "target schema"
  - Based on information from an instance of the local schema
    - We call this the "source schema"

5. Data Exchange

- **Data Exchange Problem Statement**
- **Input:**
  - Given a **source** and a **target schema**
  - + instance of the source schema
  - + set of schema mappings (here st-tgds)
- **Output:**
  - Instance of the target schema that fulfills constraints

5. Data Exchange

Example: Types of Matching



Name	Address
Peter	1
Alice	3
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

$$\forall x, y, z, a : Person(x, y) \wedge Address(y, z, a) \rightarrow \exists b, c : Person(x, z, a, b, c)$$

5. Data Exchange

Example: Types of Matching

Name	Address
Peter	1
Alice	2
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343		
Alice	Chicago	(312) 555 7777		
Bob	New York	(465) 123 1234		

5.1 Data Exchange Setting

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Data Exchange Setting

Data Exchange setting is a tuple **(S,T,I,Σ)**
- Schema **S**
- Schema **T**
- Instance **I** of **S**
- Mappings **Σ** from **S** to **T**

6 CS520 - 5) Data Exchange

5.1 Data Exchange Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition: Data Exchange Solution

Given data exchange setting is a tuple **(S,T,I,Σ)**
- Find instance **J** of **T** so that **(I,J)** fulfills mappings **Σ**
- **J** uses values from a **universe U** and set of **labeled nulls N**

7 CS520 - 5) Data Exchange

5.1 Data Exchange Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Solutions

Person Name → Person Name

Name	Address
Peter	1
Alice	2
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

$$\forall x, y, z, a : Person(x, y) \wedge Address(y, z, a) \rightarrow \exists b, c : Person(x, z, a, b, c)$$

Can we come up with a solution?

8 CS520 - 5) Data Exchange

5.1 Data Exchange Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Solutions

Person Name → Person Name

Name	Address
Peter	1
Alice	2
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

$$\forall x, y, z, a : Person(x, y) \wedge Address(y, z, a) \rightarrow \exists b, c : Person(x, z, a, b, c)$$

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	NULL	NULL
Alice	Chicago	(312) 555 7777	NULL	NULL
Bob	New York	(465) 123 1234	NULL	NULL

9 CS520 - 5) Data Exchange

5.1 Number of Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

- **How many solutions exists?**
  - Depends on how whether we use existentially quantified variables in the mappings?
    - i.e., do we have attributes for which we have to invent values?
  - What attribute values do we allow?
    - Surely values from the source instance (active domain)
    - NULL?
      - Need multiple NULL values as placeholders for missing values that have to be the same
  - Note that this is the open-world assumption
    - there are infinitely many solutions (if domains infinite)

10 CS520 - 5) Data Exchange

5.1 Number of Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Target instance domain**
  - Consider a **universe U**
    - Source instance can only use values from U
  - Consider an infinite **set N of labeled nulls**
    - Target instance can use these as placeholders for missing values

11 CS520 - 5) Data Exchange

5.1 Data Exchange Solutions

Example: Multiple Solutions

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	X	Y
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	C	D

Id
City
Home-phone

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	X	Y
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	C	D
Heinzbert	Pferdegert	111-222-3798	E	

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	Hometown	111-322-3454
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	Other town	D

5.1 Certain answers (… again)

- **Have multiple solutions**
  - Define certain answers for queries as before
  - Every tuple t so that t is in the result of query Q over any valid solution J
- **What's new?**
  - Want to materialize an instance so that computing certain answers over this instance is easy
    - Not immediately clear that this actually possible

5.1 Data Exchange Solutions

Example: Solution generality

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	X	Y
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	C	D

How general is solution (in terms of certain answers)?

Consider query
`Q(n) :- P(n,a,op,oa,hp), oa = Hometown`

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	Hometown	111-322-3454
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	Other town	D

5.1 Universal solutions

- **Universal solution**
  - Want a solution that is as general as possible
  - We call such most general solutions universal solutions
  - How do we know whether it is most general
    - We can map the tuples in this solution to any other less general solution by replacing unspecified values (labelled nulls) with actual data values
- **Query answering with universal solutions**
  - For UCQs: run query over universal instance
  - Remove tuples with labelled nulls
  - Result are the certain answers!

5.1 Universal Solutions

Definition: Homomorphism

A homomorphism **h** from instance **J** to instance **J'** maps the constants and nulls of **J** to the constants and nulls of **J'** and fulfills the following conditions:

- Constants are mapped onto themselves: **h(c) = c**
- Every tuple **R(a_1,…,a_n)** in **J** is mapped to a tuple in **J'**:
  R(a_1,…,a_n) in J -> **R(h(a_1), …,h(a_n))** in **J'**

Definition: Universal solution

Given data exchange setting **(S,T,I,Σ)**. An instance **J** of **T** is called an universal solution for a source instance **I** if it is a solution and for every other solution **J'** hold that

- There exists a homomorphism from **J** to **J'**

5.1 Data Exchange Solutions

Example: Solution generality

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	X	Y
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	C	D

How general is solution (in terms of certain answers)?

Consider query
`Q(n) :- P(n,a,op,oa,hp), oa = Hometown`

5.1 Data Exchange Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Solution generality

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	X	Y
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	C	D

Above is universal solution

How to map to below non-universal solution?
Replace generic labelled Nulls with values:
X -> Hometown, Y-> 111-322-3454, C -> other town,

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	Hometown	111-322-3454
Alice	Chicago	(312) 555 7777	A	A
Bob	New York	(465) 123 1234	Other town	D

5.2 Computing Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Note**
  - Schema mappings (st-tgds) are tuple-generating dependencies
  - What other tgd's do we know
    - Foreign keys
  - How did we solve violations to FKs?
    - **The chase!**
  - Chase produces universal solution!

5.2 Computing Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Can we use a database system to compute solutions?**
  - Yes, systems such as Clio generate queries that compute universal solutions!
    - SQL
    - Java
    - XSLT (for XML docs)

5.2 Computing Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Generating Executable Transformations**
  - How to preserve semantics of labeled nulls
    - n = n' is true if we have the same labeled null only
    - n = n' if one is a constant and the other one is a labeled null

5.2 Skolem Functions

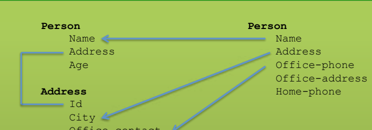
ILLINOIS INSTITUTE OF TECHNOLOGY

- **Skolem functions for labeled nulls**
  - For each existential variable in a tgd we create a new skolem function
  - What should be the arguments of the function?
    - Naïve: all universally quantified variables
    - Better: only relevant ones

5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

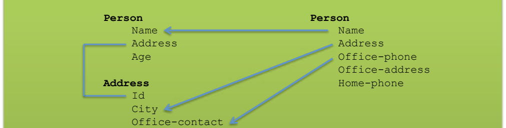
Example: Skolem Functions



5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Skolem Functions



$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f,g Person(a,f,g) \wedge Address(f,b,c)$

Introduce skolem function **sk1** and **sk2** for **f** and **g**.

What arguments to choose for **sk1** and **sk2**?

E.g.,, **f** should be fixed for a certain address and should not depend on the person.

24 CS520 - 5) Data Exchange

5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clio Schema Graph Algorithm**
- **Nodes**
  - Create a graph with one node for every target attribute and one node for every target relation
  - Also add nodes for source attribute if they are copied to the target according to the mapping
- **Edges**
  - Edges between a relation and its attributes
  - Edges between target attributes that use the same variable
  - Edges between source attributes and target attributes if they use the same variable

25 CS520 - 5) Data Exchange

5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

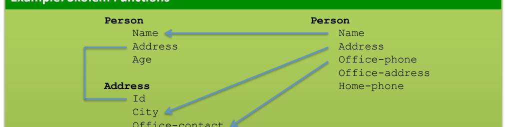
- **Clio Schema Graph Algorithm**
- **Annotations**
  - Annotate each target attribute connected to a source attribute with that source attribute
  - Propagate annotations according to the following rules
    - Propagate annotations from attributes to relations
    - Propagate annotations from relations to attributes
      - Only if attribute uses existentially quantified variable
    - Propagate annotations between target attributes connected by equality edges

26 CS520 - 5) Data Exchange

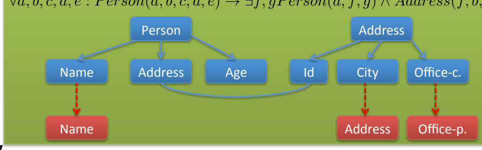
5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Skolem Functions



$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f,g Person(a,f,g) \wedge Address(f,b,c)$

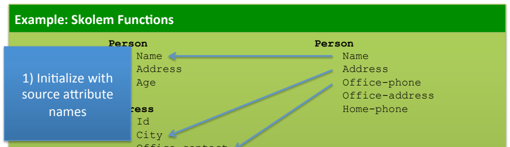


27 CS520 - 5) Data Exchange

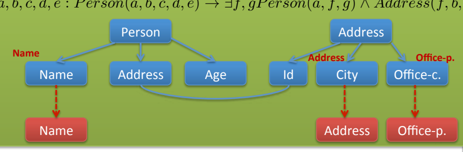
5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Skolem Functions



$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f,g Person(a,f,g) \wedge Address(f,b,c)$

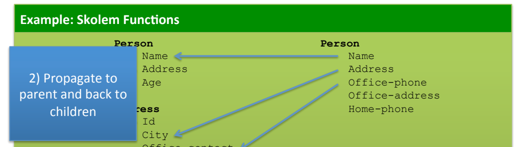


28 CS520 - 5) Data Exchange

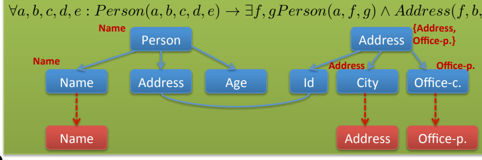
5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Skolem Functions



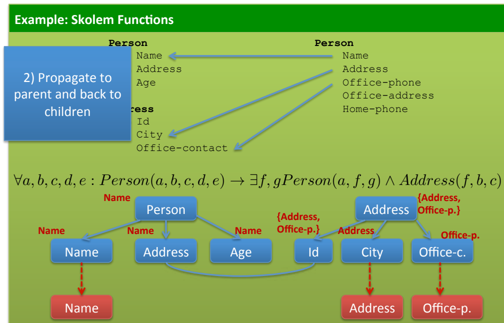
$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f,g Person(a,f,g) \wedge Address(f,b,c)$



29 CS520 - 5) Data Exchange

5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

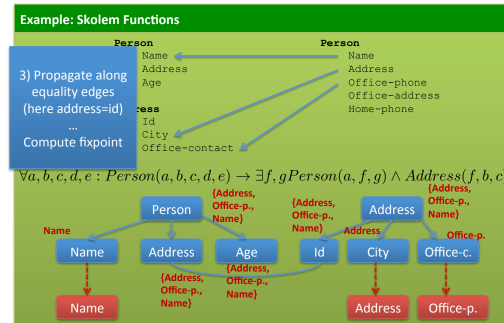


$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f, g Person(a,f,g) \wedge Address(f,b,c)$

30 CS520 - 5) Data Exchange

5.1 Data Exchange Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY



$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f, g Person(a,f,g) \wedge Address(f,b,c)$

31 CS520 - 5) Data Exchange

5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clio Schema Graph Algorithm**
- **Skolem functions**
  - Derive skolem function arguments from the schema graph annotations of an element

Example: Skolem Functions

$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f, g Person(a,f,g) \wedge Address(f,b,c)$

For variable f (id, address) we assign sk1(a,b,c)
For variable g(age) we assign sk2(a,b,c)

32 CS520 - 5) Data Exchange

5.2 Executable Transformations

ILLINOIS INSTITUTE OF TECHNOLOGY

- **SQL Code Generation Example**
  - For each tgd mentioning a target relation R we generate a query fragment
  - All query fragments for R are "unioned" together
  - A query fragment is
    - A FROM and WHERE clause that is a direct translation of the LHS of a tgd into SQL
    - A SELECT clause corresponding the R atom in the RHS using attributes from the FROM clause can the skolem functions we have determined in the previous step

33 CS520 - 5) Data Exchange

5.2 Executable Transformations

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Skolem Functions

$\forall a,b,c,d,e : Person(a,b,c,d,e) \rightarrow \exists f, g Person(a,f,g) \wedge Address(f,b,c)$

For Person atom in RHS:

```
SELECT name,
       'SK1' || name || address || office-phone AS address,
       'SK2' || name || address || office-phone AS age
FROM Person
```

For Address atom in RHS:

```
SELECT 'SK1' || name || address || office-phone AS address,
       address AS city,
       office-phone AS office-contact
FROM Person
```

34 CS520 - 5) Data Exchange

5.3 Recap Data Exchange Steps

ILLINOIS INSTITUTE OF TECHNOLOGY

- Schema Matching
- Generate Schema Mappings
  - Use constraints
- Generate Executable Transformations
  - SQL, XSLT, XQuery
  - Skolems for missing value
- Run Transformations over source instance to generate target instance
  - Universal solution

35 CS520 - 5) Data Exchange

5.3 Comparison with virtual integration

ILLINOIS INSTITUTE OF TECHNOLOGY

- Pay cost upfront instead of at query time
- Making decisions early vs. at query time
  - When generating a solution
  - Caution: bad decisions stick!
- **Universal solutions** allow efficient computation of certain types of queries using, e.g., SQL

36 CS520 - 5) Data Exchange

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

37 CS520 - 5) Data Exchange