

ILLINOIS INSTITUTE OF TECHNOLOGY

CS520



Data Integration, Warehousing, and Provenance

1. Introduction

IT DBGroup

Boris Glavic

<http://www.cs.iit.edu/~glavic/>
<http://www.cs.iit.edu/~cs520/>
<http://www.cs.iit.edu/~dbgroup/>





0

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

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




1

1

ILLINOIS INSTITUTE OF TECHNOLOGY

Overview

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



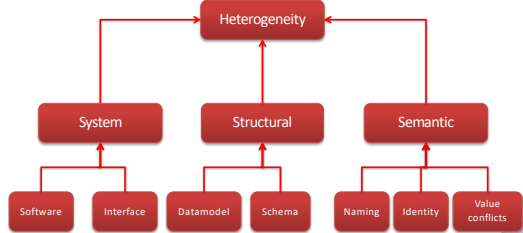

2

2

ILLINOIS INSTITUTE OF TECHNOLOGY

1.1 Heterogeneity + Autonomy

- Taxonomy of Heterogeneity



3

3

ILLINOIS INSTITUTE OF TECHNOLOGY

1.1 System Heterogeneity

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



4

4

ILLINOIS INSTITUTE OF TECHNOLOGY

1.1 System Heterogeneity

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

5

5

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
 - ...

6 CSS20 - I Introduction

6

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, ...

7 CSS20 - I Introduction

7

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

8 CSS20 - I Introduction

8

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- Interface Heterogeneity – Examples
 - SQL

9 CSS20 - I Introduction

9

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- Interface Heterogeneity – Examples
 - SQL

10 CSS20 - I Introduction

10

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

11 CSS20 - I Introduction

11

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- Interface Heterogeneity – Examples
 - Email-client

12

CSS20 - I) Introduction

12

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

13

CSS20 - I) Introduction

13

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- 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

CSS20 - I) Introduction

14

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- Integration system has to process part of the query

```

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

```

15

CSS20 - I) Introduction

15

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- Query requires multiple requests

```

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

```

16

CSS20 - I) Introduction

16

1.1 System Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- Query cannot be answered

```

SELECT title
FROM books
WHERE genre = 'SciFi';

```

17

CSS20 - I) Introduction

17

1.1 Heterogeneity +Autonomy

ILLINOIS INSTITUTE OF TECHNOLOGY

- Taxonomy of Heterogeneity

18

CSS20 - I) Introduction

18

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

19

CSS20 - I) Introduction

19

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

20

CSS20 - I) Introduction

20

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

21

CSS20 - I) Introduction

21

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

22

CSS20 - I) Introduction

22

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Modeling choices

```

Male(id, firstname, lastname)
Female(id, firstname, lastname)
Person(id, firstname, lastname, male, female)
Person(id, firstname, lastname, gender)
    
```

23

CSS20 - I) Introduction

23

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

24

24

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)
    
```

25

25

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)
    
```

Multiple attribute vs one attribute

Missing derivable attribute: Role

Derivable attribute: Compute age from birthday

26

26

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

27

27

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Conflicts between attributes and attributes

SSN	FirstName	LastName	Age
333-333-3333	Peter	Schmeter	30
333-333-9999	Hans	Glanz	NULL

SSN	FirstName	SurName	Age
3333333333	Peter	Schmeter	30
3333339999	Hans	Glanz	-1

28

28

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Conflicts between attributes and attributes

SSN	FirstName	LastName	Age
333-333-3333	Peter	Schmeter	30
333-333-9999	Hans	Glanz	NULL

SSN	FirstName	SurName	Age
3333333333	Peter	Schmeter	30
3333339999	Hans	Glanz	-1

Conflicting format

Conflicting datatype

Conflicting constraint

Conflicting default value

synonym

29

29

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

Example

30

CSS20 - I) Introduction

30

1.1 Structural Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

Example

31

CSS20 - I) Introduction

31

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)

32

CSS20 - I) Introduction

32

1.1 Heterogeneity + Autonomy

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Taxonomy of Heterogeneity**

33

CSS20 - I) Introduction

33

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

34

CSS20 - I) Introduction

34

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

35

CSS20 - I) Introduction

35

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

36 CSS20 - I) Introduction

36

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

Example

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

37 CSS20 - I) Introduction

37

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

Example

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

38 CSS20 - I) Introduction

38

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

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

Example

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

39 CSS20 - I) Introduction

39

1.1 Semantic Heterogeneity

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Naming concepts (units)**

Example

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

40 CSS20 - I) Introduction

40

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 CSS20 - I) Introduction

41

1.1 Semantic Heterogeneity


ILLINOIS INSTITUTE OF TECHNOLOGY

- Identity Conflicts

Example

(IBM,300000000,USA)

(International Business Machines Corporation,50000)



42


CSS20 - I) Introduction

42

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 entity first!
 - Resolving such conflicts requires **Data Fusion**
 - Pick value from conflicting values
 - Numerical methods: e.g., average
 - Preferred value
 - ...



43


CSS20 - I) Introduction

43

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
 - ...



44


CSS20 - I) Introduction

44

1.2 Data integration tasks

ILLINOIS INSTITUTE OF TECHNOLOGY

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



45


CSS20 - I) Introduction

45

1.3 Data integration architectures

ILLINOIS INSTITUTE OF TECHNOLOGY

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



46


CSS20 - I) Introduction

46

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



47

CSS20 - I) Introduction

47

1.4 Boolean Logic

ILLINOIS INSTITUTE OF TECHNOLOGY

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

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

48 CSS20 - I Introduction

48

1.4 Boolean Logic

ILLINOIS INSTITUTE OF TECHNOLOGY

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

$\top \wedge \perp = \perp$
$\top \wedge \top = \top$
$\perp \vee \top = \top$
...

49 CSS20 - I Introduction

49

1.4 Boolean Logic

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

Formula:

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

A possible valuation:

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

Evaluating the formula:

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

50 CSS20 - I Introduction

50

1.4 First-order logic (FO)

ILLINOIS INSTITUTE OF TECHNOLOGY

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

$$R_1, \dots, R_n$$
 - These represent relations (in the mathematical sense)
 - An infinite set of variables \mathcal{X}**

51 CSS20 - I Introduction

51

1.4 FO Syntax

ILLINOIS INSTITUTE OF TECHNOLOGY

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

$\psi \wedge \phi$	$\psi \vee \phi$	$\neg \phi$
$\psi \rightarrow \phi$	$\exists x : \psi$	$\forall x : \psi$

52 CSS20 - I Introduction

52

1.4 Free / Bound Variables

ILLINOIS INSTITUTE OF TECHNOLOGY

- Free variables of a formula**
 - All variables not bound by quantifiers

$$free(\neg \psi) = free(\psi)$$

$$free(\psi \wedge \phi) = free(\psi) \cup free(\phi)$$

$$free(\psi \vee \phi) = free(\psi) \cup free(\phi)$$

$$free(\forall x : \psi) = free(\psi) - \{x\}$$

$$free(\exists x : \psi) = free(\psi) - \{x\}$$

$$free(R(t_1, \dots, t_n)) = free(t_1) \cup \dots \cup free(t_n)$$

$$free(x) = \{x\}$$

$$free(c) = \emptyset$$

53 CSS20 - I Introduction

53

1.4 FO Semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

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

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

54 CSS20 - I Introduction

54

1.4 FO Semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Given a model \mathcal{M} and valuation μ**
 - The “result” of a formula $\llbracket \psi \rrbracket_{\mathcal{M}, \mu}$

$$\llbracket c \rrbracket_{\mathcal{M}, \mu} = c$$

$$\llbracket x \rrbracket_{\mathcal{M}, \mu} = \mu(x)$$

$$\llbracket R(t_1, \dots, t_n) \rrbracket_{\mathcal{M}, \mu} = \begin{cases} \top & \text{if } (\llbracket t_1 \rrbracket_{\mathcal{M}, \mu}, \dots, \llbracket t_n \rrbracket_{\mathcal{M}, \mu}) \in R \\ \perp & \text{otherwise} \end{cases}$$

$$\llbracket \psi \wedge \phi \rrbracket_{\mathcal{M}, \mu} = \llbracket \psi \rrbracket_{\mathcal{M}, \mu} \wedge \llbracket \phi \rrbracket_{\mathcal{M}, \mu}$$

$$\llbracket \psi \vee \phi \rrbracket_{\mathcal{M}, \mu} = \llbracket \psi \rrbracket_{\mathcal{M}, \mu} \vee \llbracket \phi \rrbracket_{\mathcal{M}, \mu}$$

$$\llbracket \neg \psi \rrbracket_{\mathcal{M}, \mu} = \neg \llbracket \psi \rrbracket_{\mathcal{M}, \mu}$$

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

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

55 CSS20 - I Introduction

55

1.4 FO semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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

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

Valuation: $\mu(x) = 1$

Result:

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

56 CSS20 - I Introduction

56

1.4 FO Problems

ILLINOIS INSTITUTE OF TECHNOLOGY

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

57 CSS20 - I Introduction

57

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

58 CSS20 - I Introduction

58

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
 - ...


59 CSS20 - I Introduction

59

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



60 CSS20 - I Introduction

60

1.4 Integrity Constraints

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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



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

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

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

Foreign Key $R(\underline{A}, B)$, $S(C, D)$ where D is FK to R :

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


61 CSS20 - I Introduction

61

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



62 CSS20 - I Introduction


62

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

$$Q(\vec{x}) : -R_1(\vec{x}_1), \dots, R_n(\vec{x}_n).$$
- One Q is specified as the answer relation (the relation returned by the query)



63 CSS20 - I Introduction

63

1.4 Datalog - Intuition

ILLINOIS INSTITUTE OF TECHNOLOGY


- A **Datalog rule**

$$Q(\vec{x}) : -R_1(\vec{x}_1), \dots, R_n(\vec{x}_n).$$
- **Procedural Interpretation:** For all bindings of variables that makes the RHS true (conjunction) return bindings of \vec{x}

Example

$Q(\text{Name}) :- \text{Person}(\text{Name}, \text{Age}).$

Return names of persons



64 CSS20 - I Introduction


64

1.4 Datalog - Syntax

ILLINOIS INSTITUTE OF TECHNOLOGY

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

$$Q(\vec{x}) : -R_1(\vec{x}_1), \dots, R_n(\vec{x}_n).$$
- **X's** are lists of variables and constants
- **Ri's** are relation names
- **Q** is a relation name



65 CSS20 - I Introduction

65

1.4 Datalog - Terminology

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

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

66

1.4 Datalog - Terminology

Example

$$Q(N) :- \text{Person}(N, A).$$

N, A are **variables**
 $Q(N)$, $\text{Person}(N, A)$ are **atoms**
 Person and Q are **predicates**

Name	Age
peter	34
bob	45

Active domain
 $\text{adom}(I) = \{\text{peter}, \text{bob}, 34, 45\}$

67

1.4 Datalog - Terminology

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

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

68

1.4 Datalog - Safety

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

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

Example

$$Q(\text{Name}) :- \text{Person}(\text{Name}, \text{Age}). \quad \text{(safe)}$$

$$Q(\text{Name}, \text{Sal}) :- \text{Peron}(\text{Name}, \text{Age}). \quad \text{(unsafe)}$$

69

1.4 Datalog - Semantics

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

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

70

1.4 Datalog - Semantics

Example

$$Q(N) :- \text{Person}(N, A).$$

N=peter, A=peter: $Q(\text{peter}) :- \text{Person}(\text{peter}, \text{peter}).$
 N=peter, A=bob: $Q(\text{peter}) :- \text{Person}(\text{peter}, \text{bob}).$
N=peter, A=34: $Q(\text{peter}) :- \text{Person}(\text{peter}, 34).$
 N=bob, A=peter: $Q(\text{bob}) :- \text{Person}(\text{peter}, \text{peter}).$
 N=bob, A=bob: $Q(\text{bob}) :- \text{Person}(\text{peter}, \text{bob}).$
N=bob, A=34: $Q(\text{bob}) :- \text{Person}(\text{bob}, 34).$
 N=34, A=peter: $Q(34) :- \text{Person}(34, \text{peter}).$
 N=34, A=bob: $Q(34) :- \text{Person}(34, \text{bob}).$
 N=34, A=34: $Q(34) :- \text{Person}(34, 34).$

N
peter
bob

Active domain
 $\text{adom}(I) = \{\text{peter}, \text{bob}, 34\}$

Name	Age
peter	34
bob	34

71

1.4 Datalog

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

72

72

1.4 Datalog

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

73

73

1.4 Datalog – Semantics (Negation)

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

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

74

74

1.4 Datalog - Semantics

Example

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

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

Result

N
bob

Lives

Name
peter

Person

Name	Age
peter	34
bob	34

Active domain
 $\text{adom}(I) = \{\text{peter}, \text{bob}, 34\}$

75

75

1.4 Datalog

Example

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

```
Q2hop(x,z) : hop(x,y), hop(y,z).
```

```
Qreach(x,y) : hop(x,y).
```

```
Qreach(x,z) : Qreach(x,y), Qreach(y,z).
```

```
Qnode(x) : hop(x,y).
```

```
Qnode(x) : hop(y,x).
```

76

76

1.4 Datalog

Example

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

```
Qnode(x) : hop(x,y).
```

```
Qnode(x) : hop(y,x).
```

```
QnotReach(x,y) : Qnode(x), Qnode(y),
                  not Qreach(x,y).
```

77

77

1.4 Datalog versus FO

ILLINOIS INSTITUTE OF TECHNOLOGY


- A Datalog rule is a FO implication:
 $Q(X, Y) : \neg R(X, Z), R(Z, Y).$

Means

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

- Databases can be expressed as rules!
 $R = \{(Peter, Bob), (Bob, Alice)\}$

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

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



78

78

1.4 Model-theoretic semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

- The result of a Datalog program P is the smallest model \mathcal{M} for the program if interpreted as a logical formula




79

79

1.4 Free Datalog Systems

ILLINOIS INSTITUTE OF TECHNOLOGY

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



80

80


1.4 Containment and Equivalence

ILLINOIS INSTITUTE OF TECHNOLOGY

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

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

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

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



81

81

1.4 Equivalence

ILLINOIS INSTITUTE OF TECHNOLOGY

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



82

82

1.4 Containment and Equiv.

ILLINOIS INSTITUTE OF TECHNOLOGY

Example


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

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

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

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

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

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


83

83

1.4 Containment and Equiv.

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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

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

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

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

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

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

84 CS520 - I Introduction

84

1.4 Complexity of Eq. and Cont.

ILLINOIS INSTITUTE OF TECHNOLOGY

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

85 CS520 - I Introduction

85

1.4 Complexity of Eq. and Cont.

ILLINOIS INSTITUTE OF TECHNOLOGY

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

86 CS520 - I Introduction

86

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 reducing these problems to the same problem
 - [Chandra & Merlin, 1977]
- Notational Conventions:
 - **head(Q)** = variables in head of query Q
 - **body(Q)** = atoms in body of Q
 - **vars(Q)** = all variable in Q

87 CS520 - I Introduction

87

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 $Q :- \dots$ as a convention for $Q() :- \dots$
 - What is the result of a Boolean query
 - Empty result {}, e.g., no **hop(x,y), hop(y,z)**
 - If there are tuples matching the body, then a tuple with zero attributes is returned {}
 - > We interpret {} as **false** and {} as **true**
 - Boolean query is essentially an existential check

88 CS520 - I Introduction

88

1.4 Boolean Conjunctive Queries

ILLINOIS INSTITUTE OF TECHNOLOGY

- BCQ in SQL

Example

Hop relation: Hop(A,B)

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

SELECT EXISTS (SELECT * FROM hop)

Note: in Oracle and DB2 we need a from clause

89 CS520 - I Introduction

89

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

90

90

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)
```

91

91

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

- How to check for containment of CQs (set)

Definition: Variable Mapping

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

Definition: Containment Mapping

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

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

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

92

92

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Theorem: Containment Mappings and Query Containment

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

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

Example

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

Can we find a containment mapping?

93

93

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

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

Q1 -> Q2 : Psi(u)=x, Psi(z)=y
Q2 -> Q1 : Psi(x)=u, Psi(y)=z
```

94

94

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

```
Q1(a, b) : R(a, b), R(b, c) .
Q2(x, y) : R(x, y) .
```

95

95

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?

$Q_1 \rightarrow Q_2$: none exists
 $Q_2 \rightarrow Q_1$: $\Psi(x)=a, \Psi(y)=b$

96

CS520 - I Introduction

96

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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

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

97

CS520 - I Introduction

97

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

98

CS520 - I Introduction

98

1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

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

99

CS520 - I Introduction

99

1.4 Containment Mappings

ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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

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

$Q_1(D) = \{(1,1), (1,2)\}$
 $\varphi(a)=1, \varphi(b)=2, \varphi(c)=1$

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

100

CS520 - I Introduction

100

1.4 Containment Background

ILLINOIS INSTITUTE OF TECHNOLOGY

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

101

CS520 - I Introduction

101

1.4 Containment Mappings

Example

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

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

$Q_2(I^{Q_1}) = \{()\}$
 $\varphi(x)=a, \varphi(y)=b$

φ is our containment mapping Ψ

102

CSS20 - I Introduction



102

1.4 Containment Background

- If you are not scared and want to know more:
 - Look up Chandra and Merlins paper(s)
 - The text book provides a more detailed overview of the proof approach
 - Look at the slides from Phokion Kolaitis excellent lecture on database theory
 - <https://classes.soe.ucsc.edu/cmsc277/Winter10/>

103

CSS20 - I Introduction



103

1.4 Containment Background

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

104

CSS20 - I Introduction



104

1.4 Containment Mappings

Example

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

If there exists tuples
 $R(a,b)$ and $R(c,b)$
 in R that make Q_1 true, then we
 take
 $R(a,b)$
 to fulfill Q_2

105

CSS20 - I Introduction



105

1.4 Containment Background

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

106

CSS20 - I Introduction



106

1.4 Containment Mappings

Example

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

For every
 $R(a,b)$ and $R(c,b)$
 Q_1 returns (a) and for every
 $R(a,b)$
 Q_2 returns (a)

107


CSS20 - I Introduction



107

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, ...**




108 CSS20 - I Introduction

108

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



109 CSS20 - I Introduction

109


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

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

(Note: often scores are normalized to the range [0,1])



110 CSS20 - I Introduction

110



1.4 Similarity Measures ILLINOIS INSTITUTE OF TECHNOLOGY

Example

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

111 CSS20 - I Introduction


111

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** $d(p,q) + d(q,r) >= d(p,r)$



112 CSS20 - I Introduction


112

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** $d(p,q) + d(q,r) >= d(p,r)$



113 CSS20 - I Introduction

113

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 Neighborhood-search: find the most similar object to an object p . This problem can be efficiently solved using index structures that only apply to metric spaces



114

CSS20 - 1) Introduction

114

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



115

CSS20 - 1) Introduction

115

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



116

CSS20 - 1) Introduction

116