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

5. Data Exchange

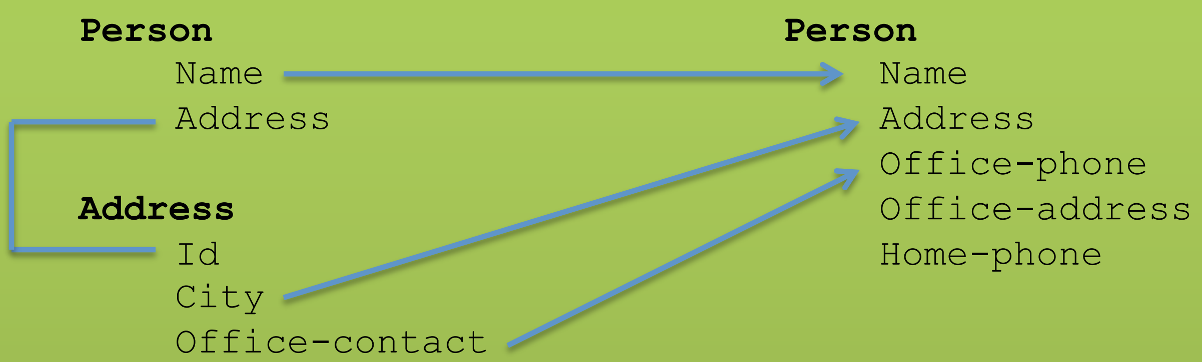
ILLINOIS INSTITUTE OF TECHNOLOGY

- **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”

- **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



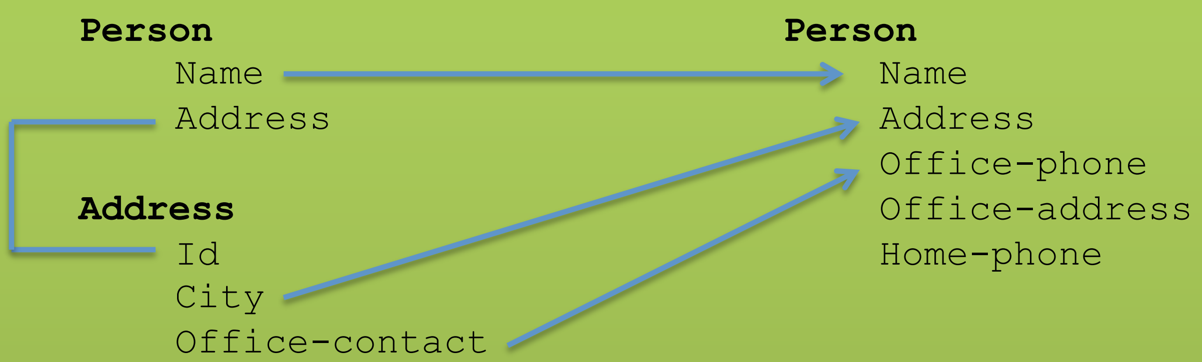
Name	Address
Peter	1
Alice	3
Bob	3

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

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

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		

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**

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**

5.1 Data Exchange Solutions

Example: Solutions

Person Name Address

Person Name Address

Name	Address
Peter	1
Alice	2
Bob	3

ntact

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?

5.1 Data Exchange Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Solutions

Person
Name
Address

Person
Name
Address

Name	Address
Peter	1
Alice	2
Bob	3

ntact

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

5.1 Number of Solutions

- **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)

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

5.1 Data Exchange Solutions

ILLINOIS INSTITUTE OF TECHNOLOGY

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)

ILLINOIS INSTITUTE OF TECHNOLOGY

- **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

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

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

ILLINOIS INSTITUTE OF TECHNOLOGY

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 $\mathbf{R(a_1,\ldots,a_n)}$ in **J** is mapped to a tuple in **J'**:
  $\mathbf{R(a_1,\ldots,a_n)}$ in J -> $\mathbf{R(h(a_1), \ldots, h(a_n))}$ 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'**

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

- **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)

- **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

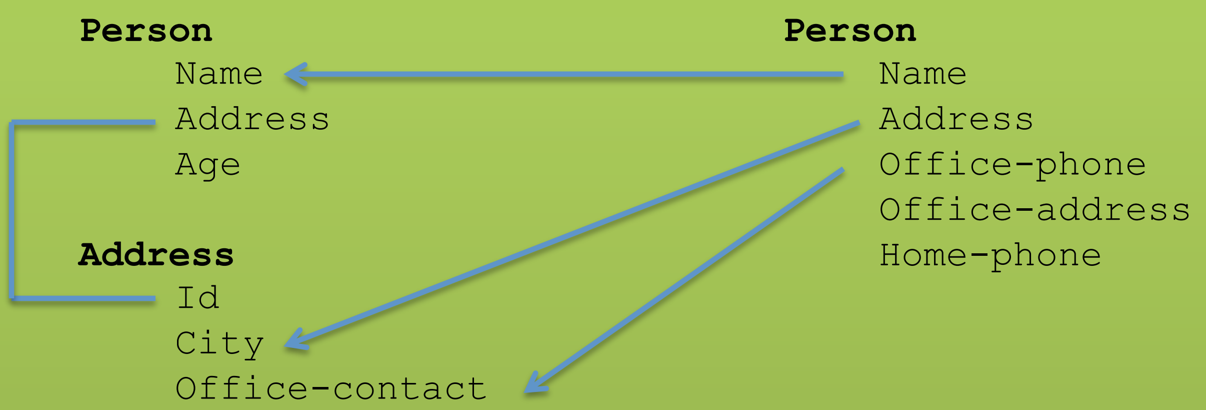
5.2 Skolem Functions

- **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

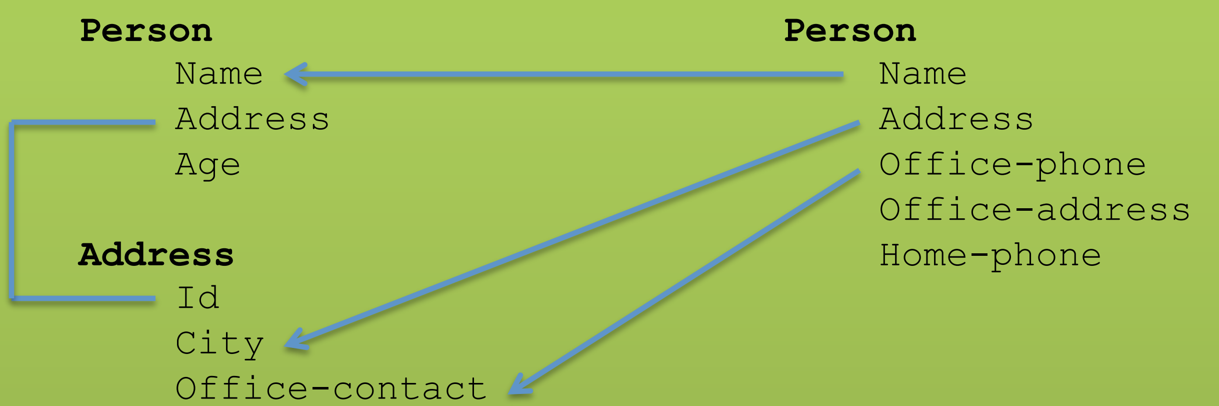
Example: Skolem Functions



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.

5.2 Skolem Functions

ILLINOIS INSTITUTE OF TECHNOLOGY

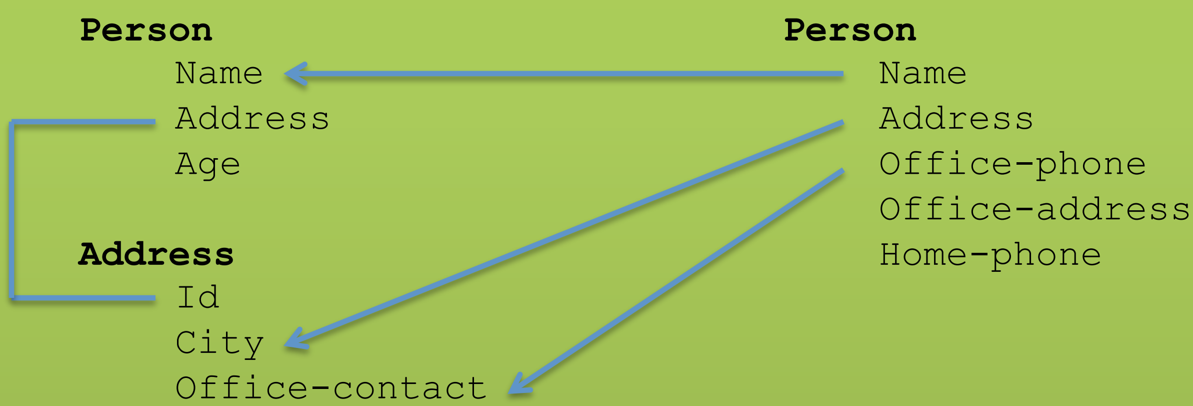
- **Clio Schema Graph Algorithm**
- **Nodes**
  - Create a graph with one node for every target attribute and one node for every target relation
  - Also add nodes for source attribute if they are copied to the target according to the mapping
- **Edges**
  - Edges between a relation and its attributes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

5.2 Skolem Functions

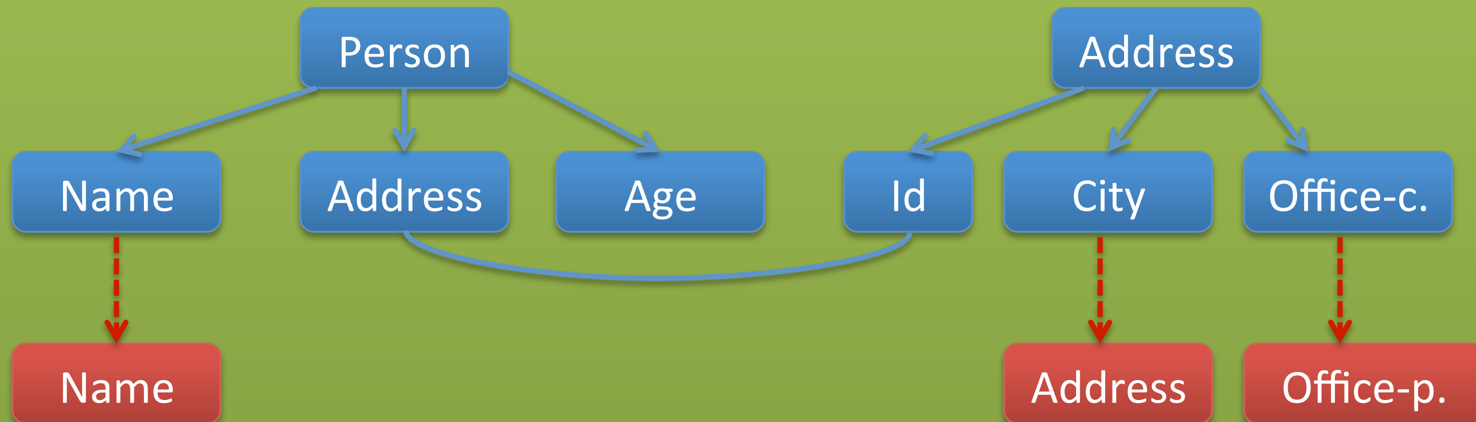
- **Clio Schema Graph Algorithm**
- **Annotations**
  - Annotate each target attribute connected to a source attribute with that source attribute
  - Propagate annotations according to the following rules
    - Propagate annotations from attributes to relations
    - Propagate annotations from relations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

5.2 Skolem Functions

Example: Skolem Functions



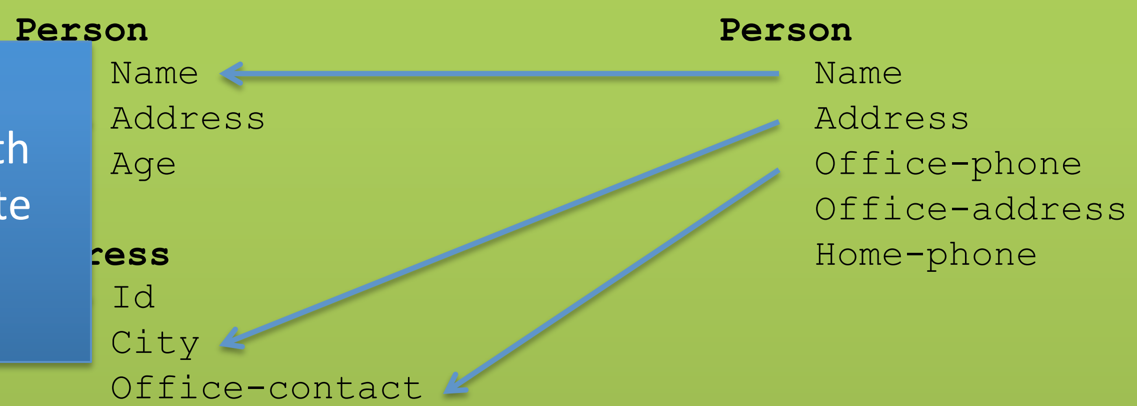
$$\forall a, b, c, d, e : Person(a, b, c, d, e) \rightarrow \exists f, g Person(a, f, g) \wedge Address(f, b, c)$$



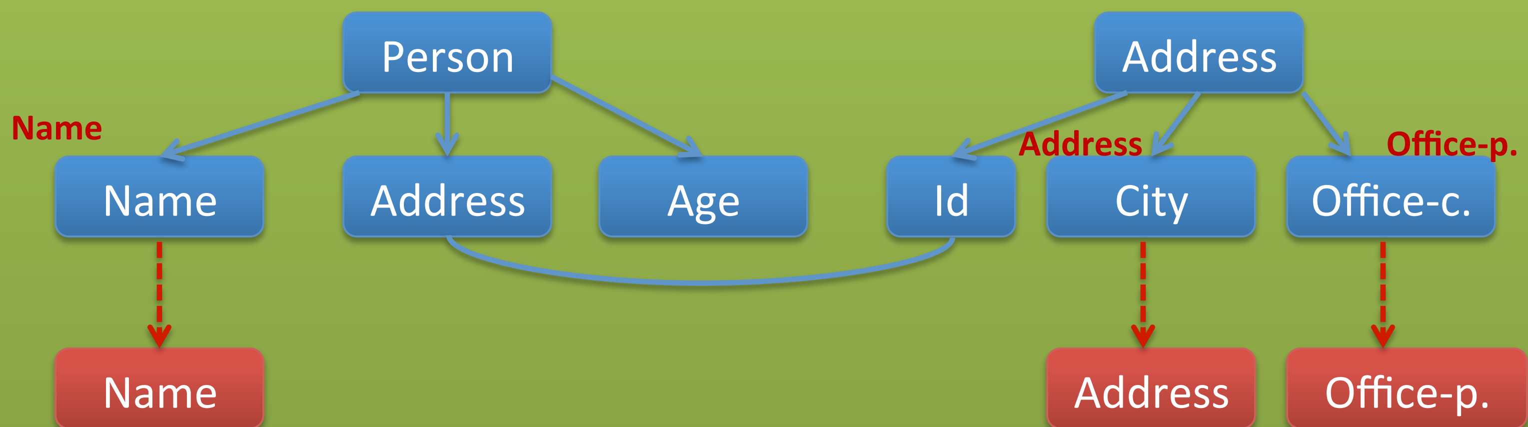
5.2 Skolem Functions

Example: Skolem Functions

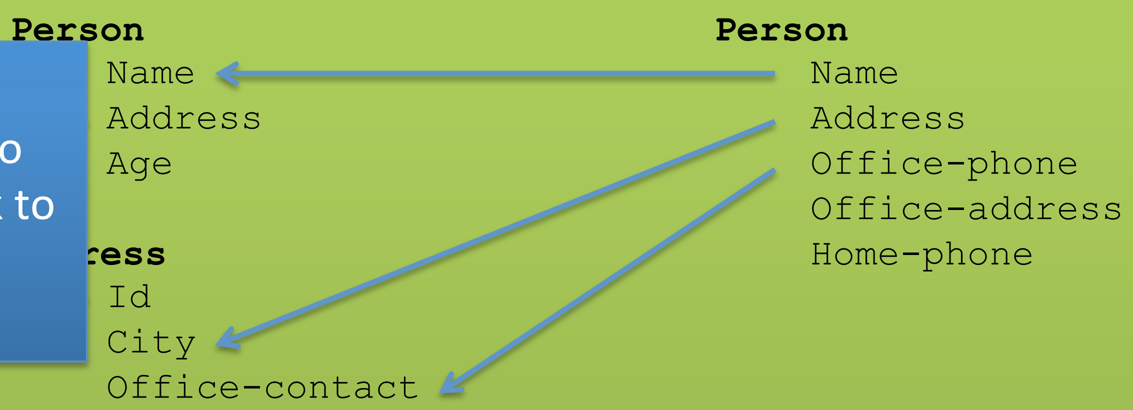
1) Initialize with source attribute names



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

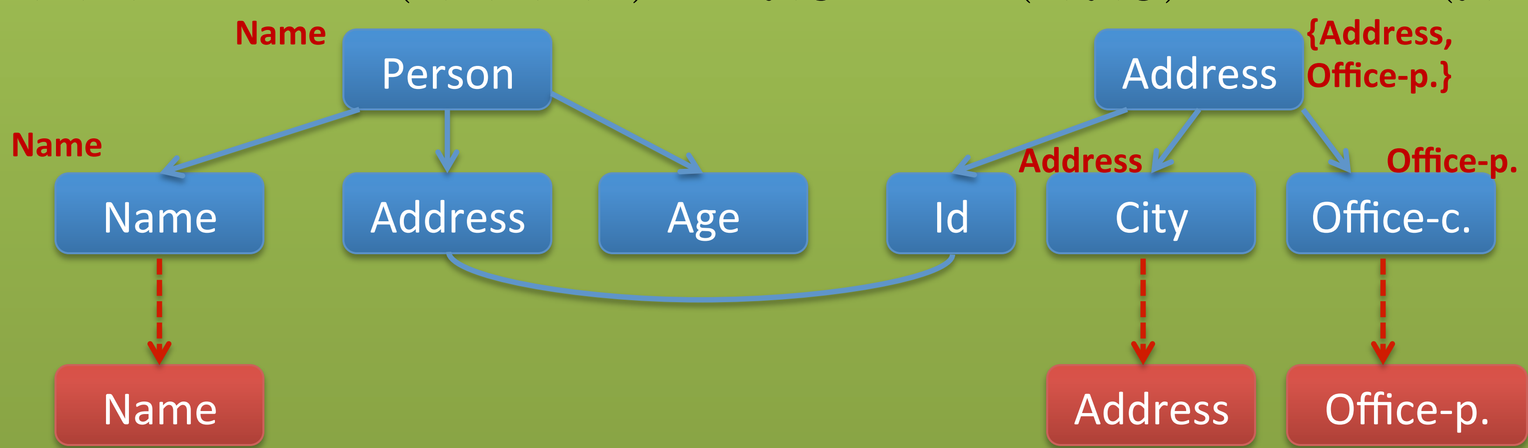


Example: Skolem Functions



2) Propagate to parent and back to children

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



5.2 Skolem Functions

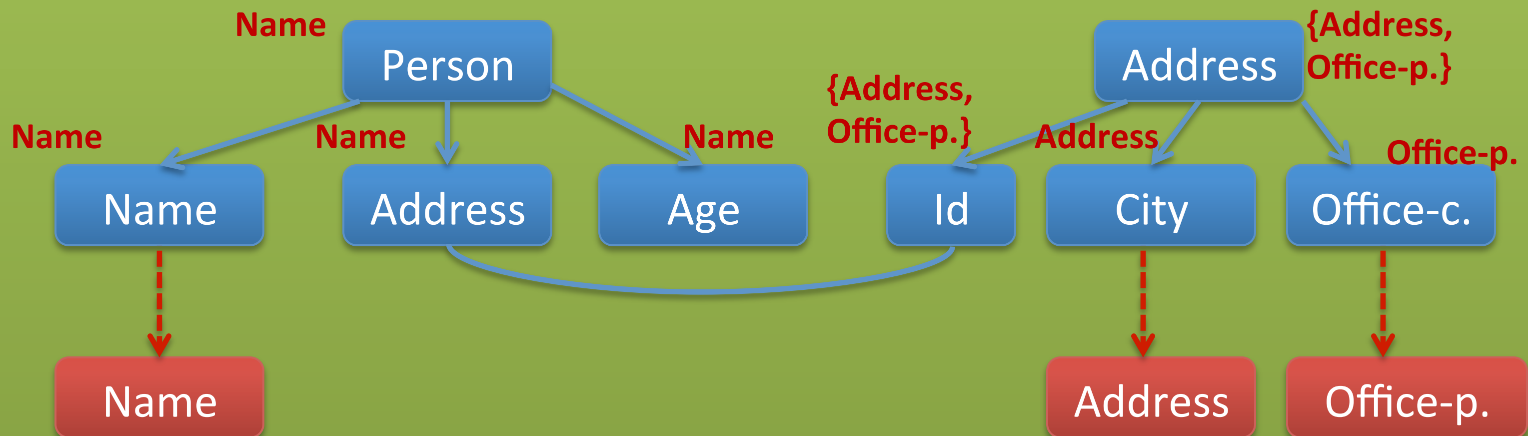
ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Skolem Functions

2) Propagate to parent and back to children

```
Person                    Person
  Name                      Name
  Address                   Address
  Age                       Office-phone
                            Office-address
 ess                        Home-phone
  Id
  City
  Office-contact
```

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

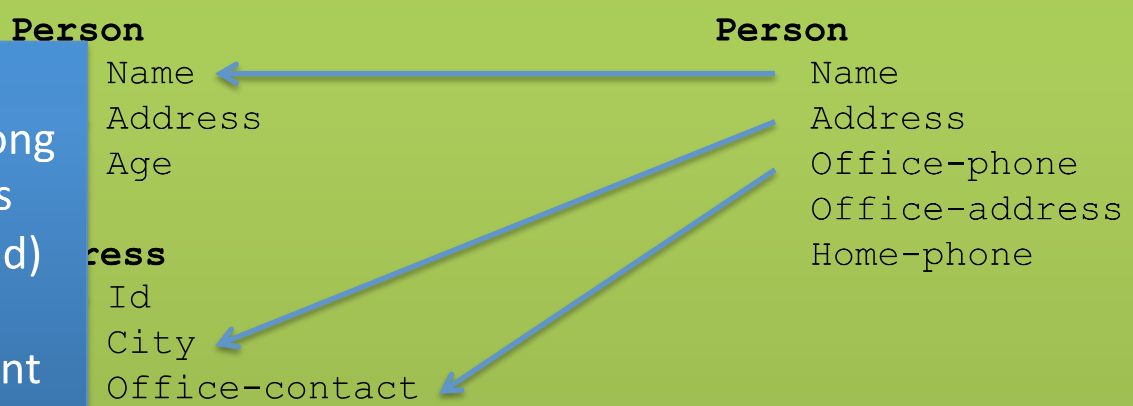


Example: Skolem Functions

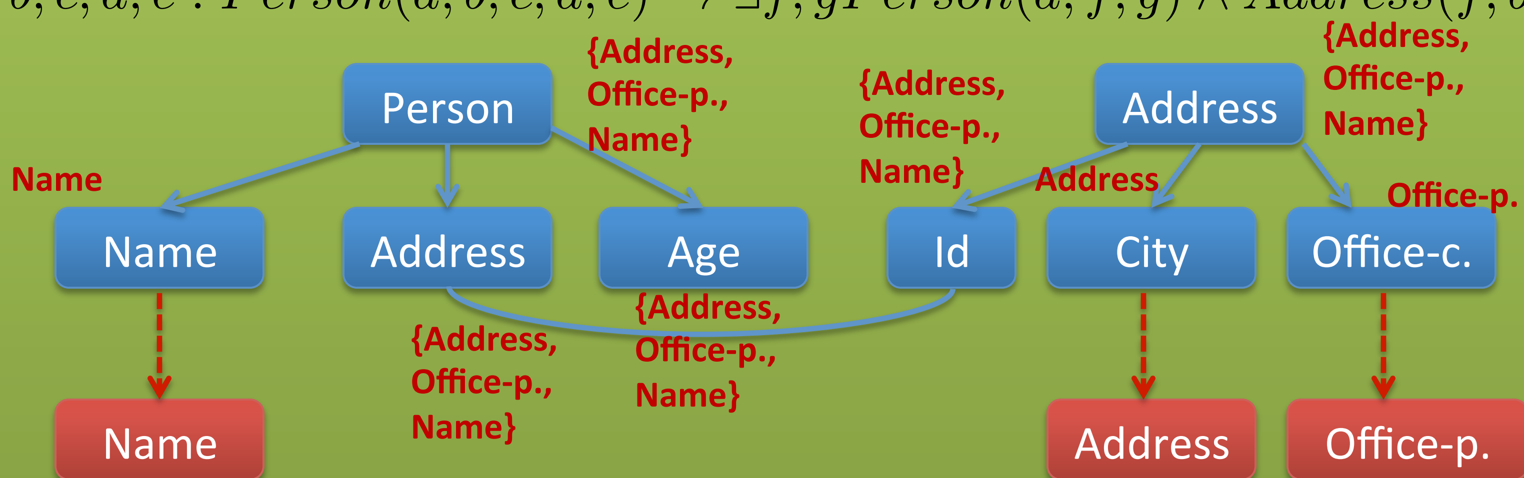
3) Propagate along equality edges (here address=id)

…

Compute fixpoint



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



- **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)

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

5.2 Executable Transformations

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Skolem Functions

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

<u>For Person atom in RHS:</u>

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

<u>For Address atom in RHS:</u>

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

5.3 Recap Data Exchange Steps

- 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

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

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