UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

# Data Exchange 1

*Lecture 5: Motivation, Relational DE, Chase*

*15 November, 2017*

*Foundations of Ontologies and Databases for Information Systems*

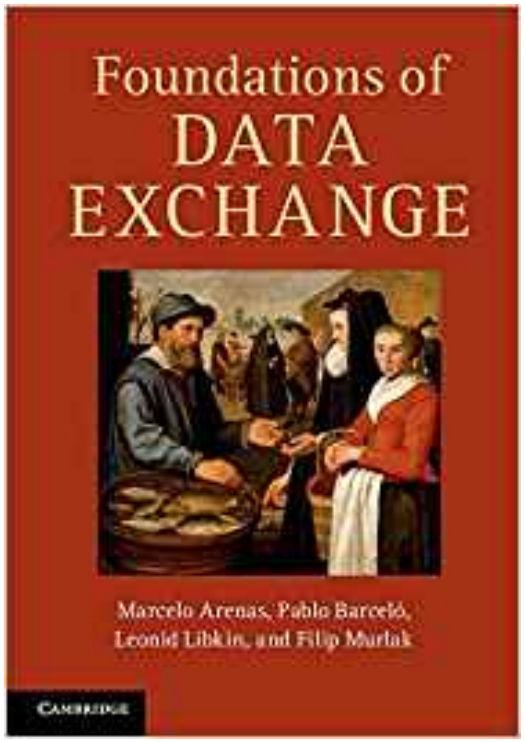
*CS5130 (Winter 17/18)*

# Short Recap

**We should understand now:**

"Finite Model Theory (FMT) is the backbone of database theory"

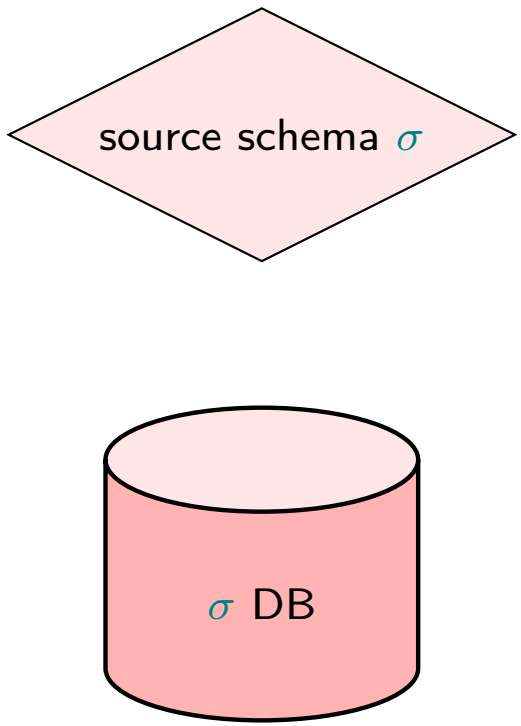
# References



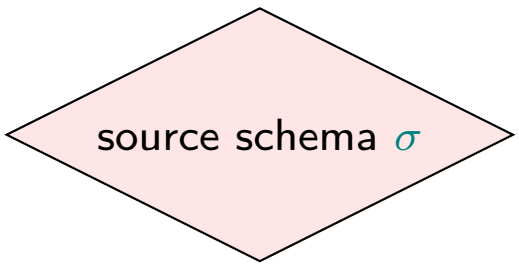

**Lit:** M. Arenas, P. Barceló, L. Libkin, and F. Murlak: Foundations of Data Exchange. Cambridge University Press, 2014.

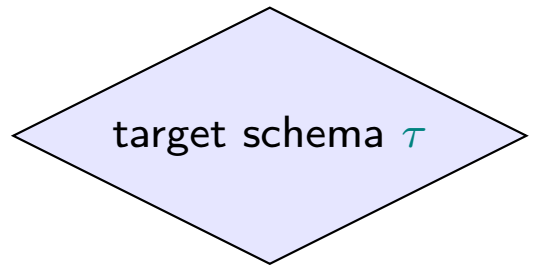
# Data Exchange: Motivation

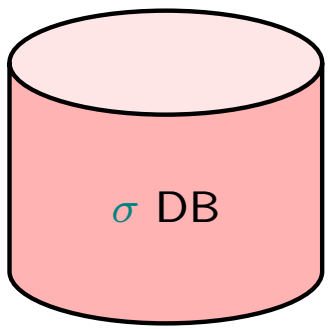
# Data Exchange (DE): Main Setting



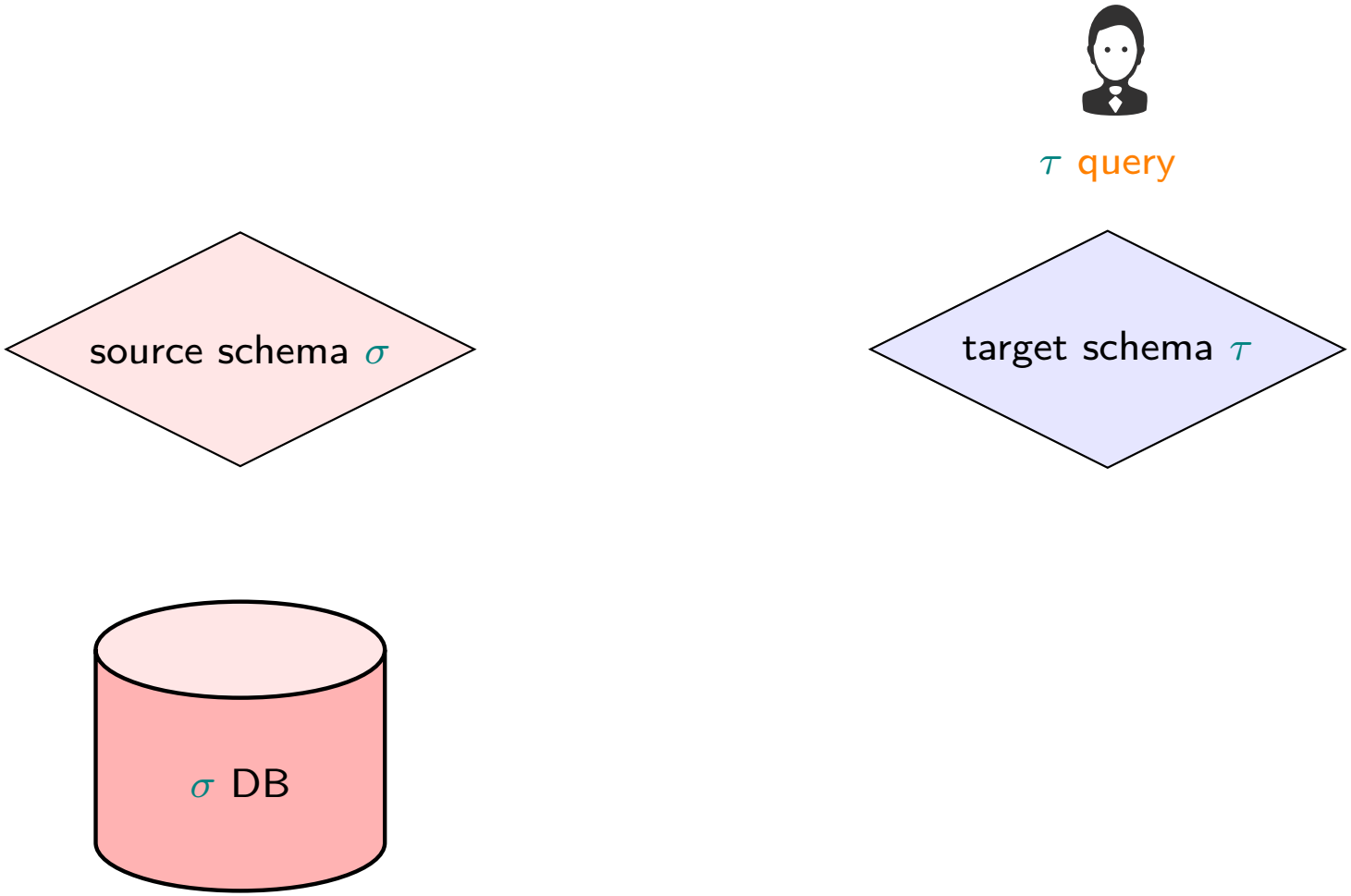
# Data Exchange (DE): Main Setting



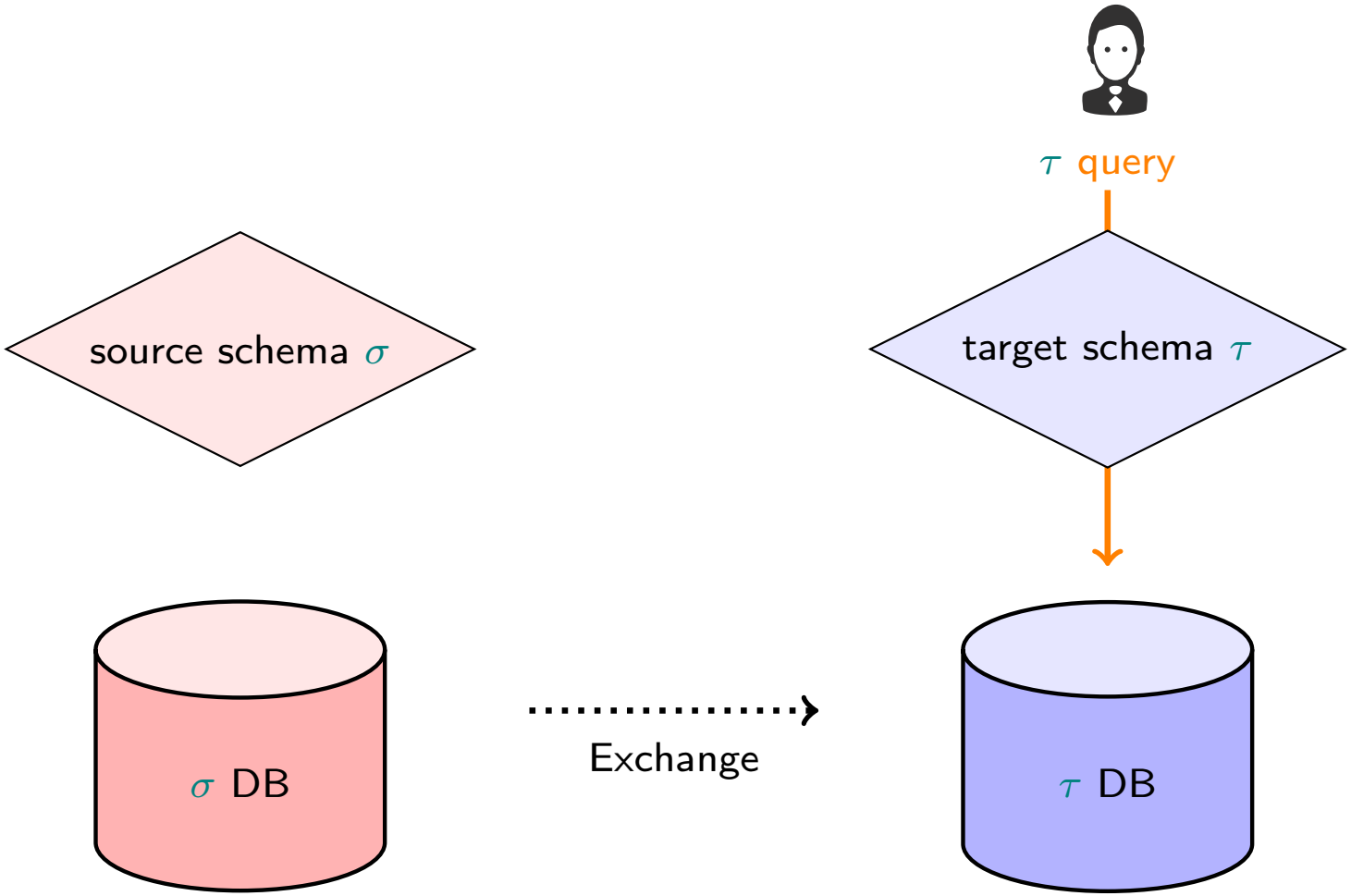



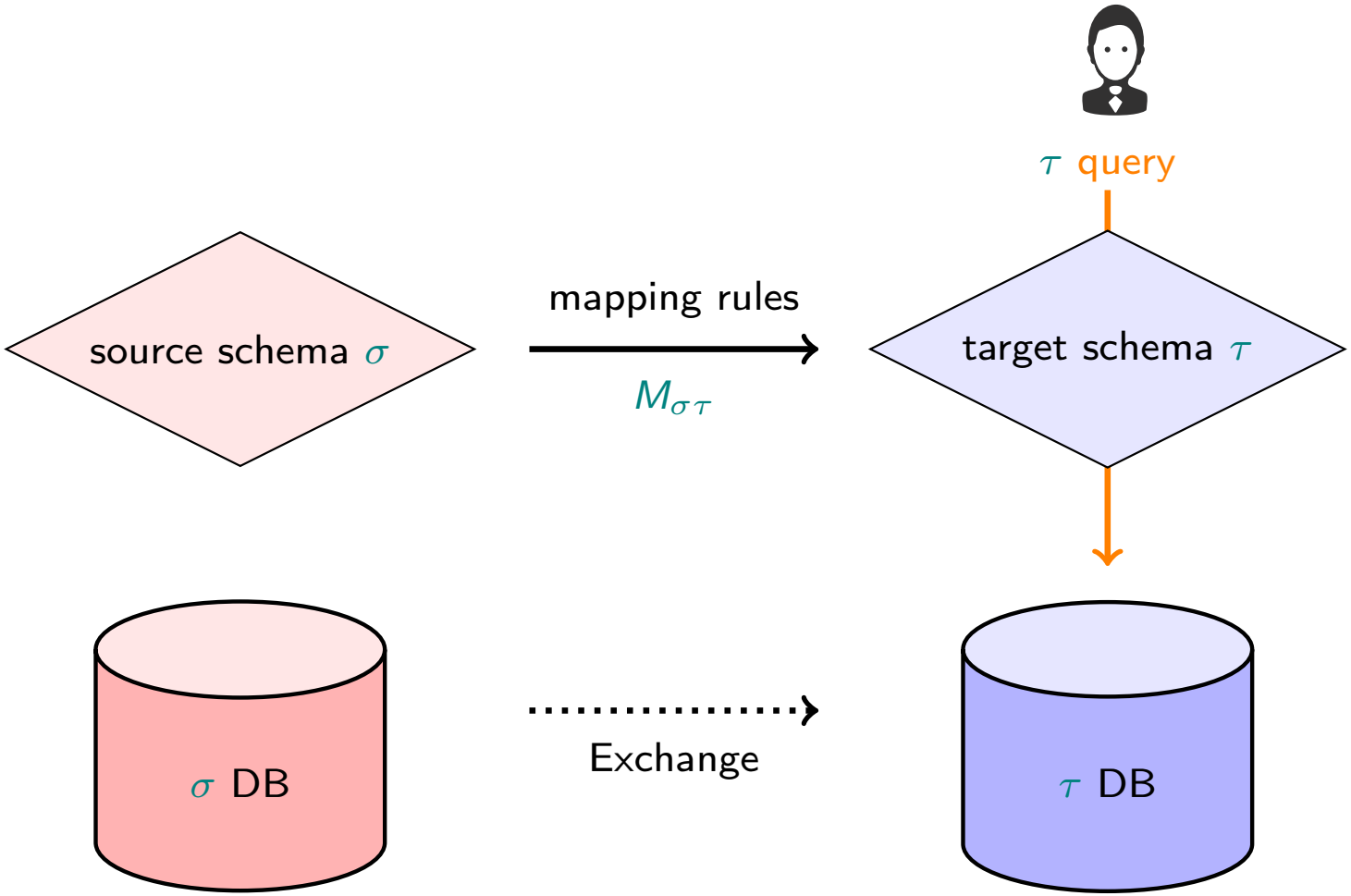
# Data Exchange (DE): Main Setting



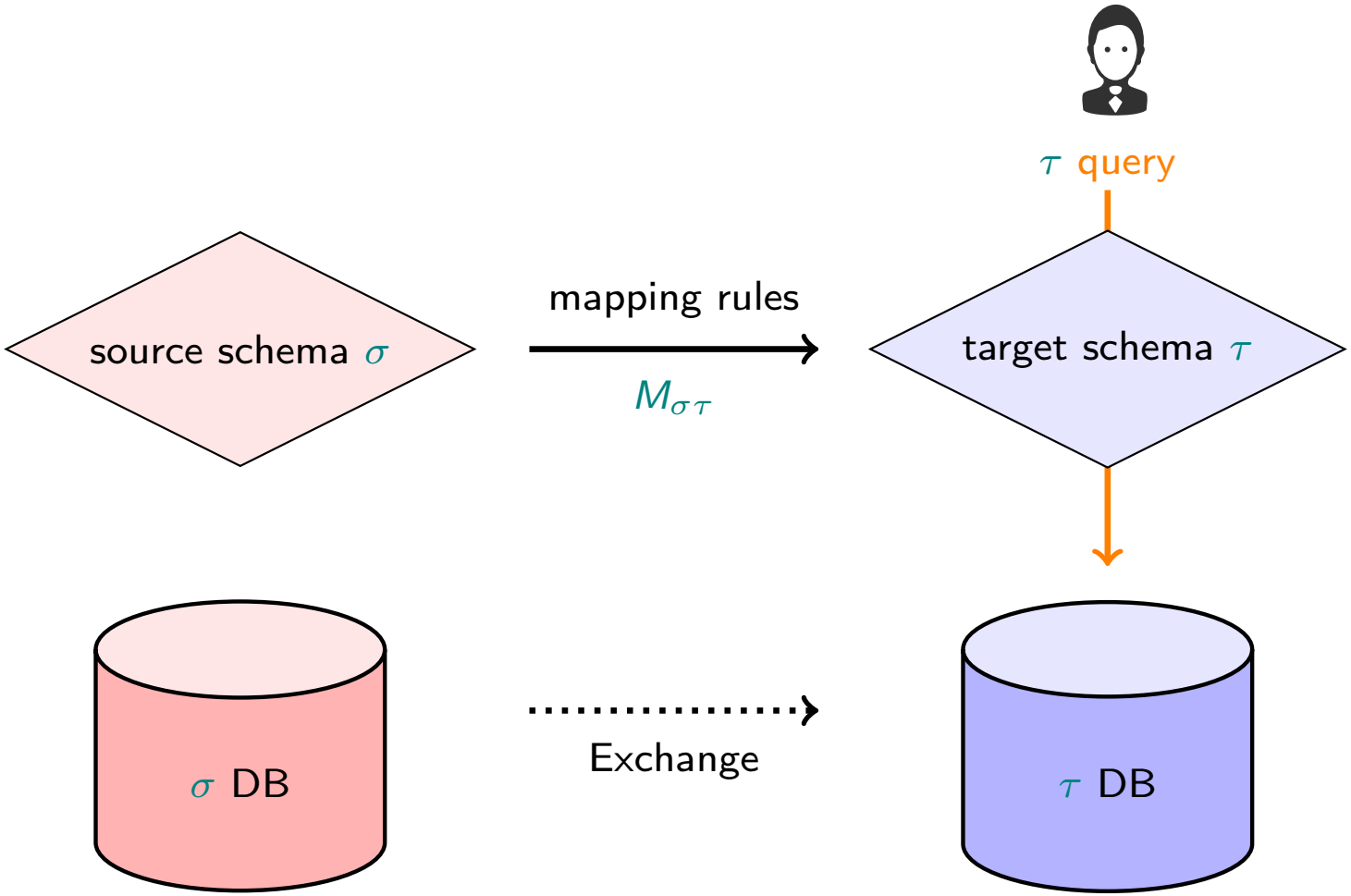
# Data Exchange (DE): Main Setting



# Data Exchange (DE): Main Setting



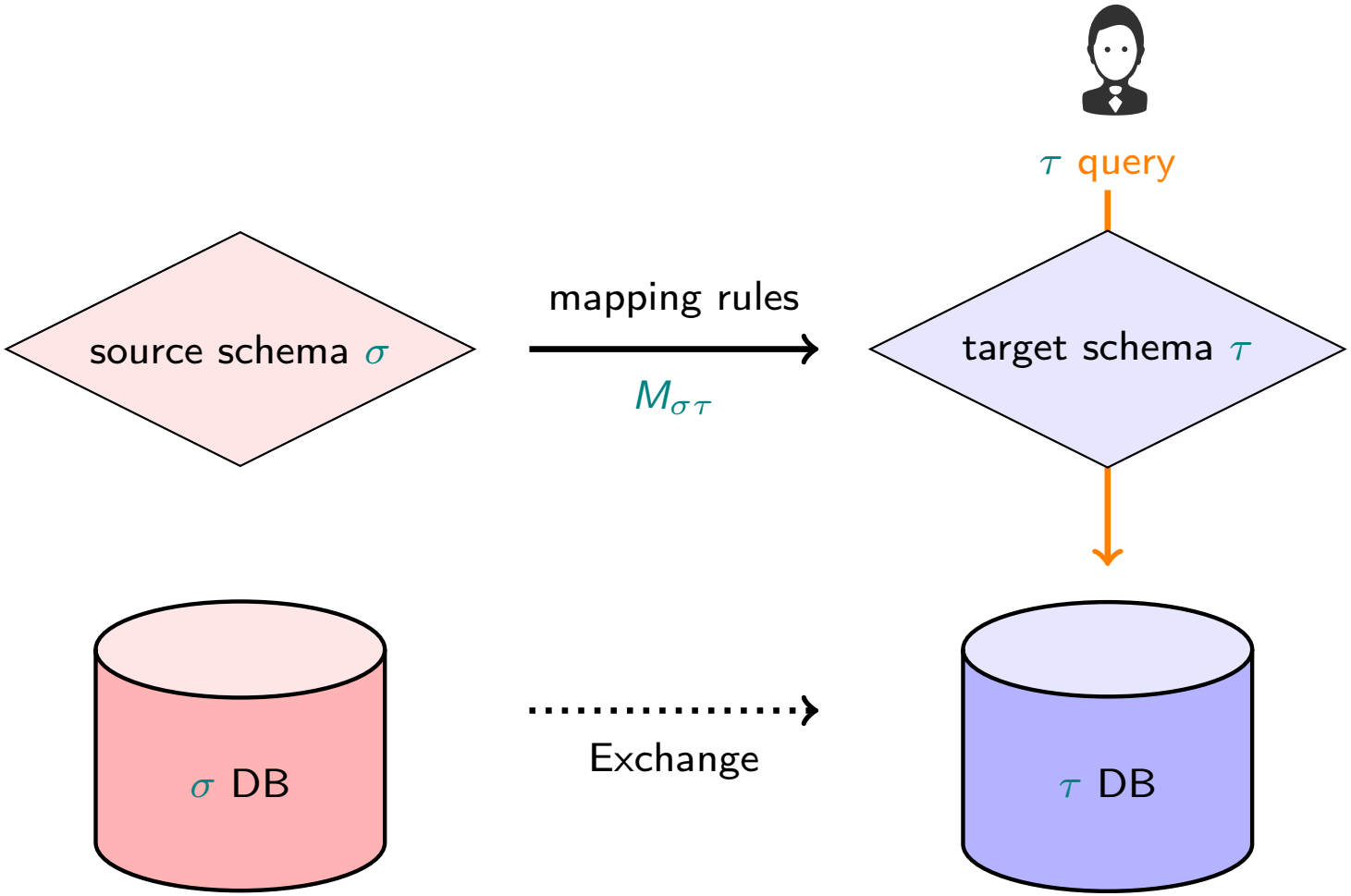
# Data Exchange (DE): Main Setting




Inputs of a DE scenario

- Source ($\sigma$) instance
- Mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, \dots)$

# Data Exchange (DE): Main Setting




Inputs of a DE scenario

- Source ($\sigma$) instance
- Mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$

$M_\tau$ = target constraints

# Data Exchange History

- Much research in DB community
- Formal treatment starts in 2003

  **Lit:** R. Fagin et al. Data exchange: Semantics and query answering. In: Database Theory - ICDT 2003, Proceedings, volume 2572 of LNCS, pages 207–224. Springer, 2003.

  **Lit:** R. Fagin, L. M. Haas, M. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis. Conceptual modeling: Foundations and applications. chapter Clio: Schema Mapping Creation and Data Exchange, pages 198–236. Springer-Verlag, Berlin, Heidelberg, 2009.
- Incorporated into IBM Clio
  `http://dblab.cs.toronto.edu/project/clio/`

# Research Context of DE

**Definition (Semantic Integration (SI))**

Research area dealing with issues (services) related to ensuring interoperability of possibly heterogeneous (data) sources using semantics.

# Research Context of DE

**Definition (Semantic Integration (SI))**

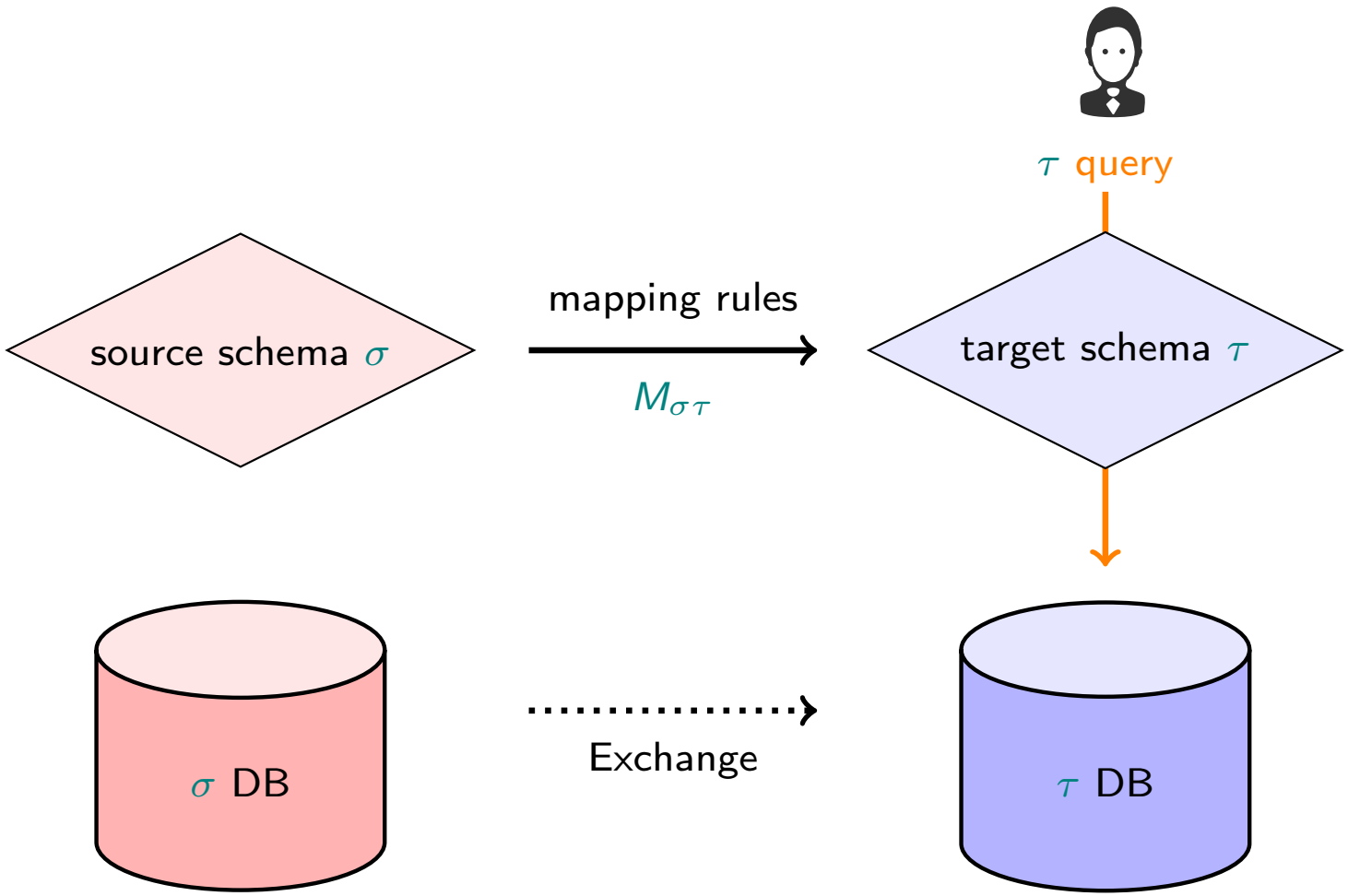
Research area dealing with issues (services) related to ensuring interoperability of possibly heterogeneous (data) sources using semantics.

Data Exchange = Asymmetric DB schema-level SI

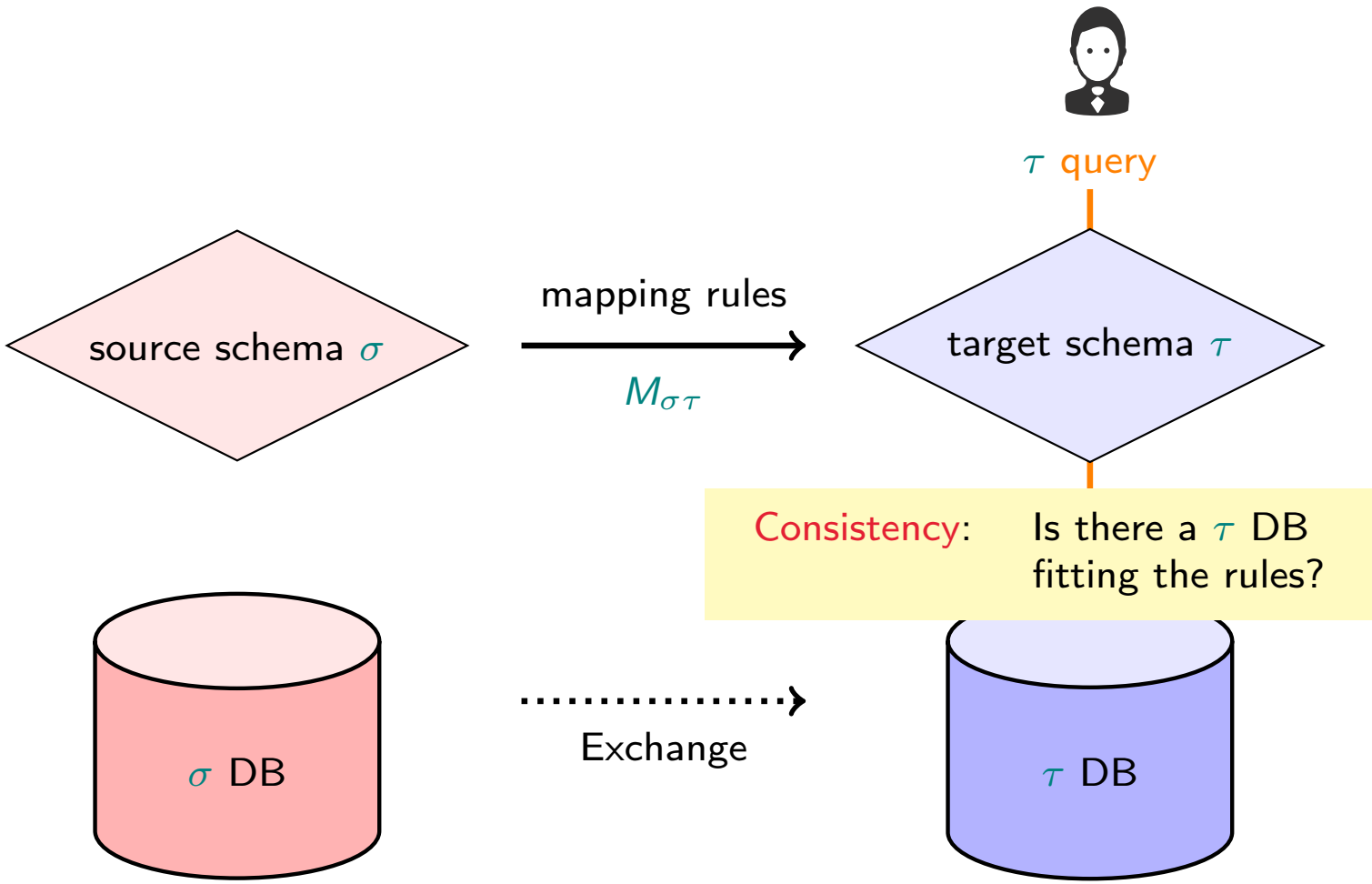
# Lecture Context

- Lecture 5 + 6: Data exchange
  - Will heavily use knowledge on FOL (Lectures 1 + 2) and FMT (Lectures 3 + 4)
- Lecture 7 + 8: Ontology-based data access
- Lecture 9+10: Ontology-level integration

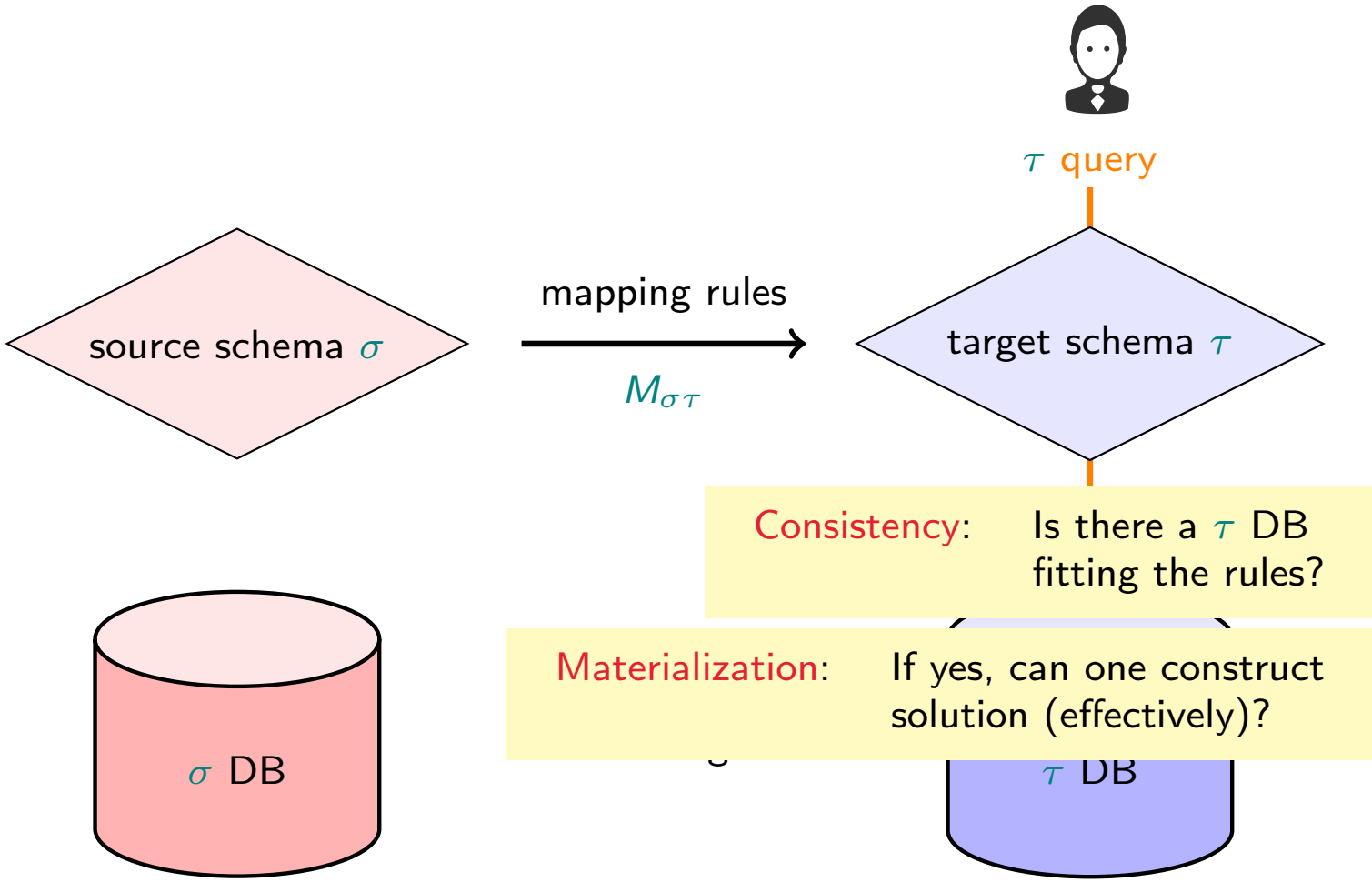
# Data Exchange: Challenges



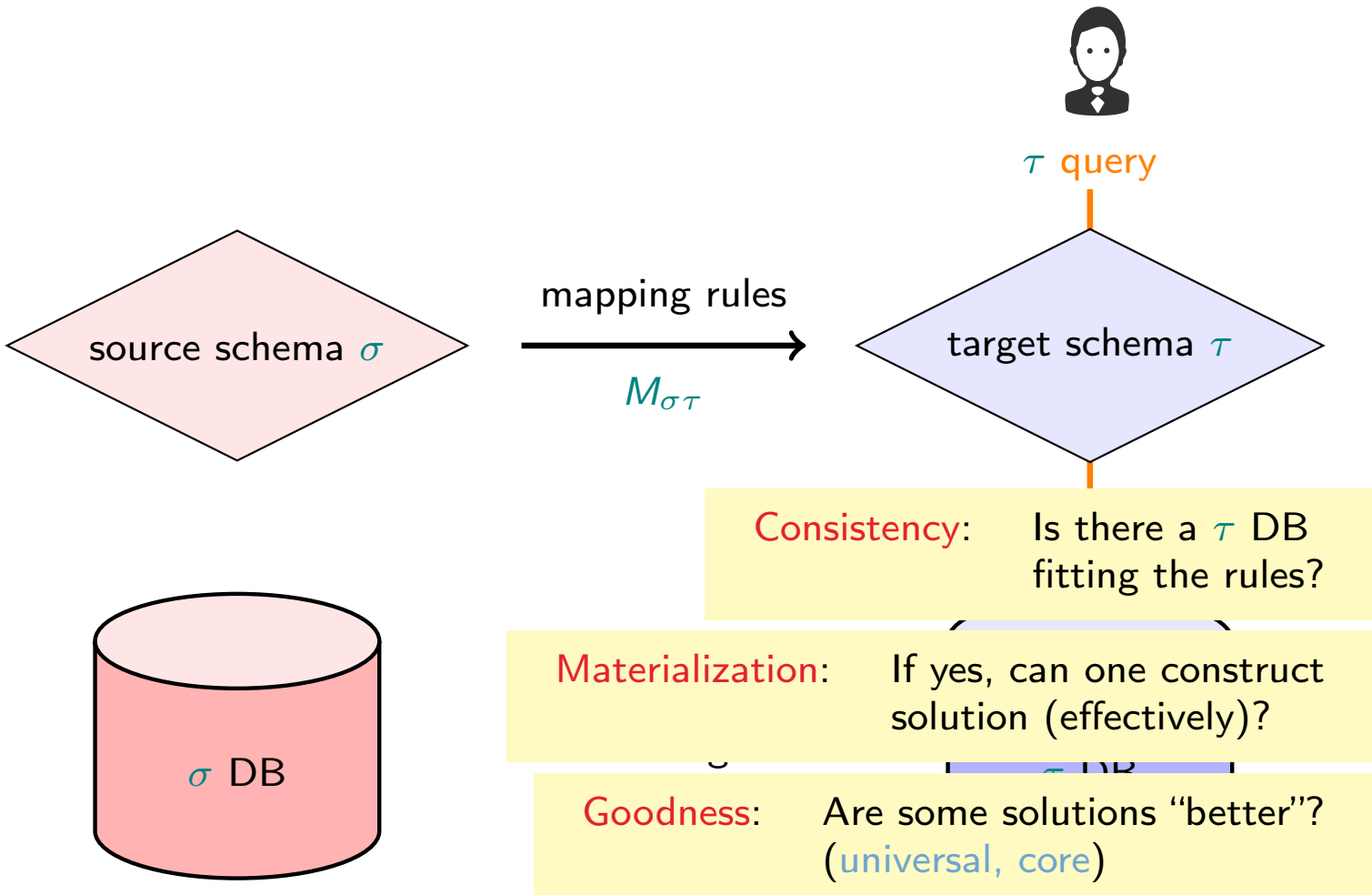
# Data Exchange: Challenges



# Data Exchange: Challenges

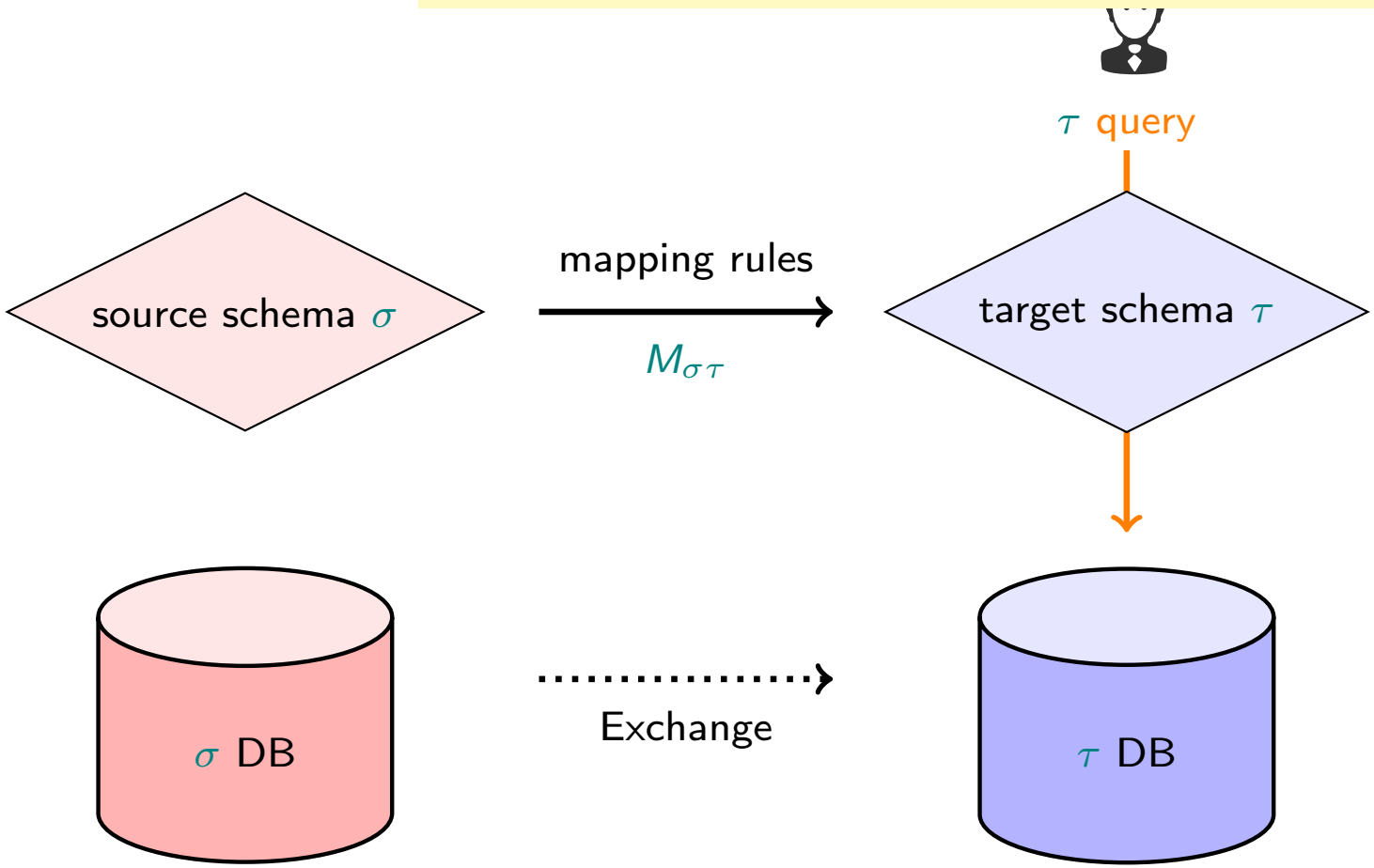


# Data Exchange: Challenges



# Data Exchange: Challenges

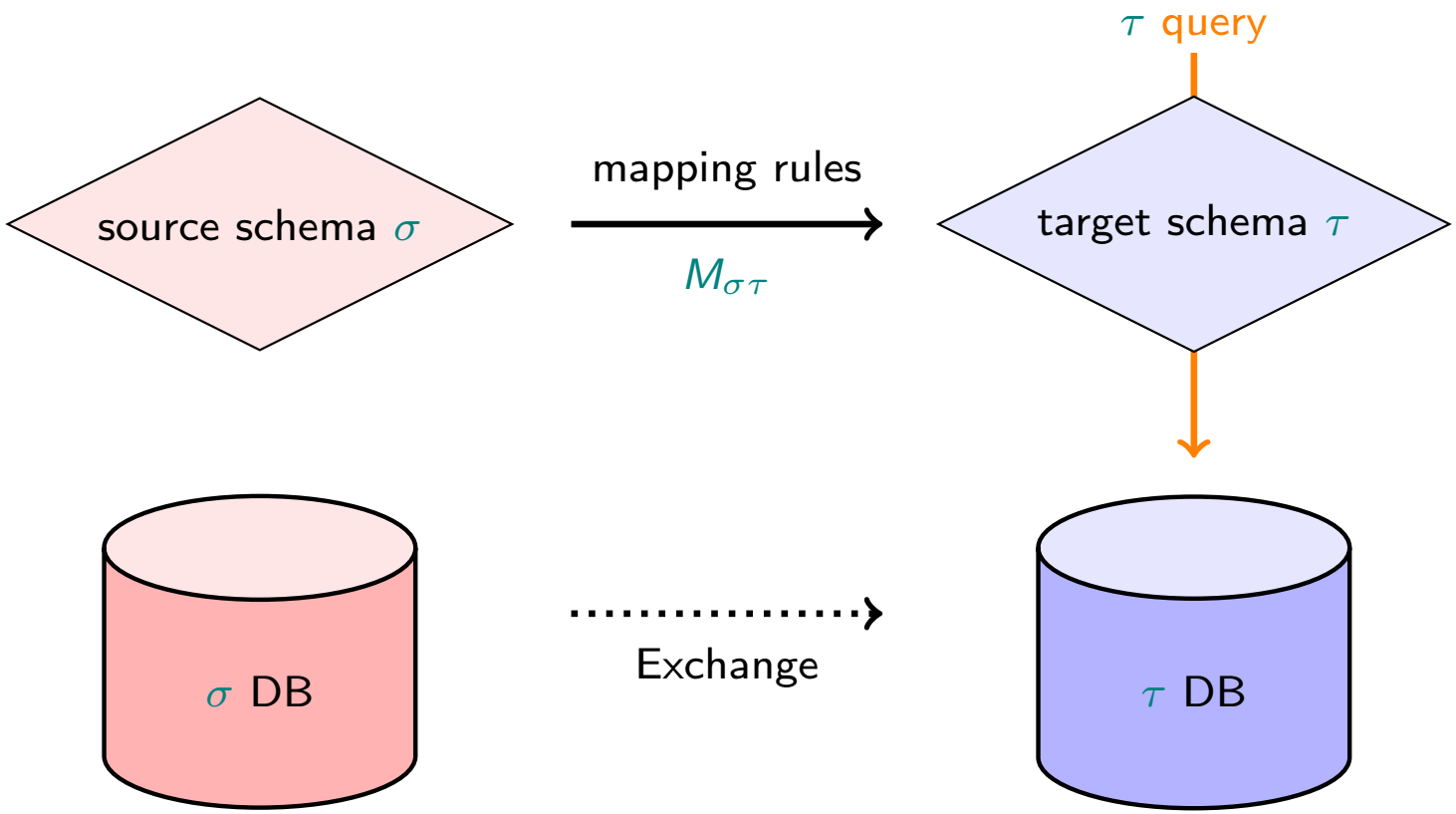
Semantics of QA: What answers are intended?
certain answers



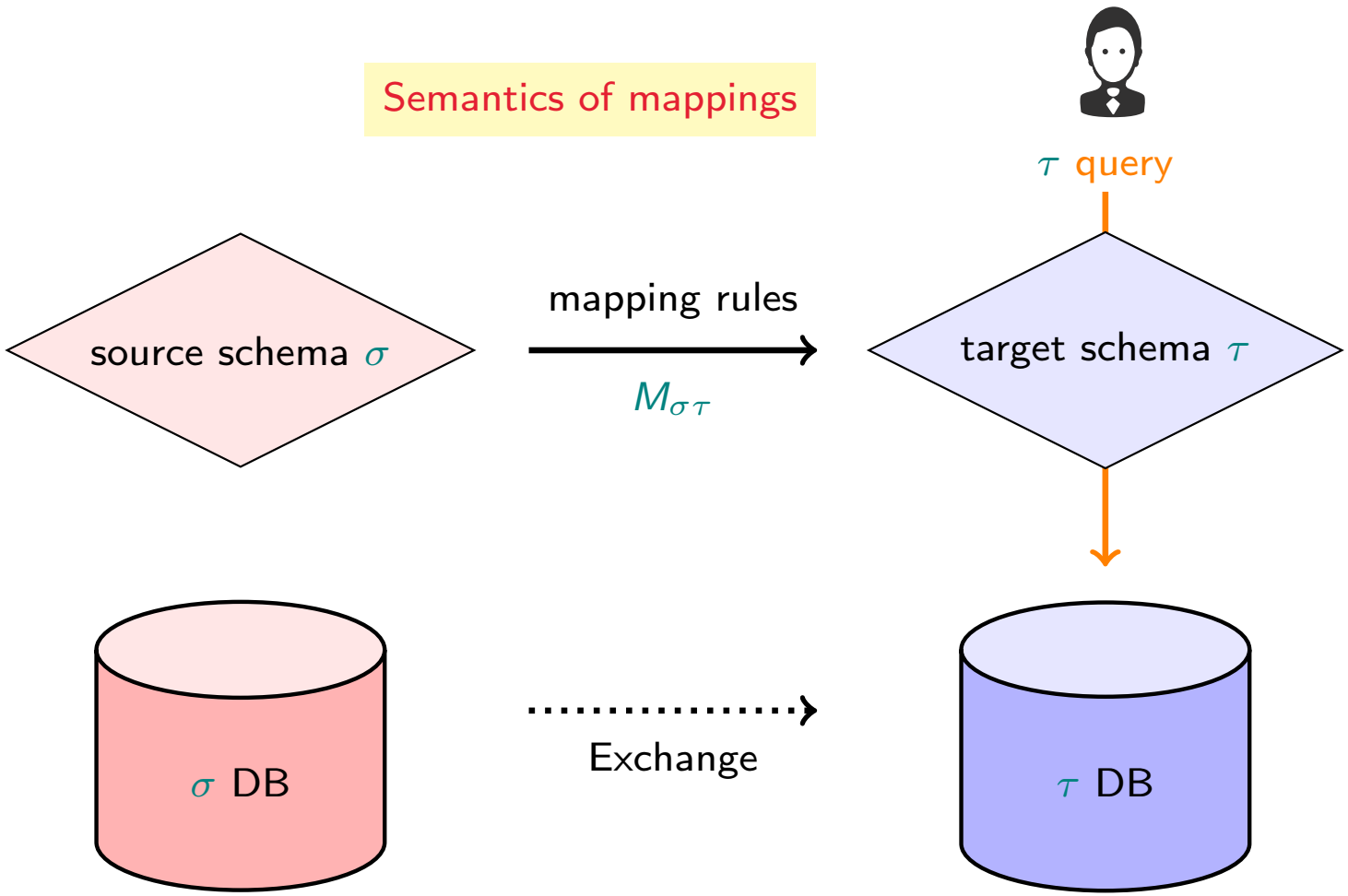
# Data Exchange: Challenges

Semantics of QA: What answers are intended?
certain answers

QA algorithm: How to calculate answers?



# Data Exchange: Challenges



# Running Example

# Example (DE in Flight Domain)

**Source schema** $\sigma$

Geo( city, coun, pop )
Flight( src, dest, airl, dep )

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

## Example (DE in Flight Domain)

**Source schema** $\sigma$

Geo( city, coun, pop )
Flight( src, dest, airl, dep )

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

## Target constraints $M_\tau$

primary key: <u>fno</u>

foreign key: $Info[\underline{fno}] \subseteq_{FK} Route[\underline{fno}]$

# Example (DE in Flight Domain)

**Source schema** $\sigma$

Geo( city, coun, pop )
Flight( src, dest, airl, dep )

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $\mathit{Flight}(\mathit{src}, \mathit{dest}, \mathit{airl}, \mathit{dep}) \longrightarrow \exists \mathit{fno}\, \exists \mathit{arr}\, (\mathit{Route}(\mathit{fno}, \mathit{src}, \mathit{dest}) \wedge \mathit{Info}(\mathit{fno}, \mathit{dep}, \mathit{arr}, \mathit{airl}))$
2. $\mathit{Flight}(\mathit{city}, \mathit{dest}, \mathit{airl}, \mathit{dep}) \wedge \mathit{Geo}(\mathit{city}, \mathit{coun}, \mathit{pop}) \longrightarrow \exists \mathit{phone}\, (\mathit{Serves}(\mathit{airl}, \mathit{city}, \mathit{coun}, \mathit{phone}))$
3. $\mathit{Flight}(\mathit{src}, \mathit{city}, \mathit{airl}, \mathit{dep}) \wedge \mathit{Geo}(\mathit{city}, \mathit{coun}, \mathit{pop}) \longrightarrow \exists \mathit{phone}\, (\mathit{Serves}(\mathit{airl}, \mathit{city}, \mathit{coun}, \mathit{phone}))$

## Example (DE in Flight Domain)

**Source schema** $\sigma$

Geo( city, coun, pop )
Flight( src, dest, airl, dep )

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

## $M_{\sigma\tau}$ = source-to-target tuple generating dependencies

Sufficiently expressive FOL formula of feasible form

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

Geo( city, coun, pop )

Flight( src, dest, airl, dep )

| | | | |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

Geo( city, coun, pop )

Flight( src, dest, airl, dep )

| | | | |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( | city, | coun, | pop ) |
|---|---|---|---|
| Flight( | src, | dest, | airl, dep ) |
| | paris | sant. | airFr 2320 |

**Target schema** $\tau$ and instance

| Route( | fno, | src, | dest ) |
|---|---|---|---|
| | $\perp_1$, | paris, | sant. |
| Info( | fno, | dep, | arr, airl ) |
| | $\perp_1$, | 2320, | $\perp_2$, airFr |
| Serves( | airl, | city, | coun, phone ) |

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists arr\ (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|

| Flight( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$ and instance

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | sant. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_1$, | 2320, | $\perp_2$, | airFr |

| Serves( airl, | city, | coun, | phone ) |
|---|---|---|---|

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

...

## Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|
| Flight( src, | dest, | airl, dep ) |
| paris | sant. | airFr 2320 |

**Target schema** $\tau$ and instance

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | sant. |
| Info( fno, | dep, | arr, airl ) |
| $\perp_1$, | 2320, | $\perp_2$, airFr |
| Serves( airl, | city, | coun, phone ) |

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

   ...

## Materialization of a $\tau$ instance ($\tau$ solution)

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

- Non-complete DB: contains marked NULLs $\perp_1, \perp_2$
- Can answer $\tau$-queries on $\mathfrak{T}$. Semantics?

## Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

Geo( city, coun, pop )

Flight( src, dest, airl, dep )
paris sant. airFr 2320

**Target schema** $\tau$ and instance

Route( fno, src, dest )
$\perp_1$, paris, sant.

Info( fno, dep, arr, airl )
$\perp_1$, 2320, $\perp_2$, airFr

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

   ...

## Materialization of a $\tau$ instance ($\tau$ solution)

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

- Non-complete DB: contains marked NULLs $\perp_1, \perp_2$
- Can answer $\tau$-queries on $\mathfrak{T}$. Semantics?

## Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|
| Flight( src, | dest, | airl, dep ) |
| paris | sant. | airFr 2320 |

**Target schema** $\tau$ and instance

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | sant. |
| Info( fno, | dep, | arr, airl ) |
| $\perp_1$, | 2320, | $\perp_2$, airFr |
| Serves( airl, | city, | coun, phone ) |

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

   ...

## Materialization of a $\tau$ instance ($\tau$ solution)

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

- Non-complete DB: contains marked NULLs $\perp_1, \perp_2$
- Can answer $\tau$-queries on $\mathfrak{T}$. Semantics?

**Definition (Certain answers over incomplete DB (informally))**

$cert(Q, \mathfrak{T}) \quad = \quad$ intersection of answers over all complete DBs represented by $\mathfrak{T}$

**Definition (Certain answers over incomplete DB (informally))**

$cert(Q, \mathfrak{T})$ = intersection of answers over all complete DBs represented by $\mathfrak{T}$

$Rep(\mathfrak{T})$ = all complete DBs resulting from $\mathfrak{T}$ by substituting marked NULLs (consistently) with constants

## Definition (Certain answers over incomplete DB (formally))

$$cert(Q, \mathfrak{T}) = Q(\mathfrak{T}) = \bigcap_{\mathfrak{T}' \in Rep(\mathfrak{T})} Q(\mathfrak{T}')$$

$Rep(\mathfrak{T})$ = all complete DBs resulting from $\mathfrak{T}$ by substituting marked NULLs (consistently) with constants

## Definition (Certain answers over incomplete DB (formally))

$$cert(Q, \mathfrak{T}) = Q(\mathfrak{T}) = \bigcap_{\mathfrak{T}' \in Rep(\mathfrak{T})} Q(\mathfrak{T}')$$

$Rep(\mathfrak{T})$ = all complete DBs resulting from $\mathfrak{T}$ by substituting marked NULLs (consistently) with constants

- Exemplifies two important general strategies
  - Reduction (here: to answering over complete DBs)
  - Minimal compromise/Summarization over all possible solutions
- Compare with full bayesian learning in ML:
  Considers all hypotheses ($\sim$ completions) consistent with training data and averages ($\sim$ intersects)

## Definition (Certain answers over incomplete DB (formally))

$$cert(Q, \mathfrak{T}) = Q(\mathfrak{T}) = \bigcap_{\mathfrak{T}' \in Rep(\mathfrak{T})} Q(\mathfrak{T}')$$

$Rep(\mathfrak{T})$ = all complete DBs resulting from $\mathfrak{T}$ by substituting marked NULLs (consistently) with constants

## Example (Answer for $\tau$ solution from flight domain)

- $\mathfrak{T} = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\}$

  $Rep(\mathfrak{T})$ = $\{\{Route(123, paris, sant), Info(123, 2320, 0815, airFr)\},$
  $\{\{Route(124, paris, sant), Info(124, 2320, 0915, airFr)\},$
  $\ldots,\}$
- $Q_1 = \exists fno\ Route(fno, paris, sant)$ $\quad cert(Q_1, \mathfrak{T}) = \{()\} = yes$
- $Q_2 = Route(123, paris, sant)$ $\quad cert(Q_2, \mathfrak{T}) = \emptyset = no$

## Definition (Certain answers over incomplete DB (formally))

$$cert(Q, \mathfrak{T}) = Q(\mathfrak{T}) = \bigcap_{\mathfrak{T}' \in Rep(\mathfrak{T})} Q(\mathfrak{T}')$$

$Rep(\mathfrak{T})$ = all complete DBs resulting from $\mathfrak{T}$ by substituting marked NULLs (consistently) with constants

## Example (Answer for $\tau$ solution from flight domain)

- $\mathfrak{T} = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\}$

  $Rep(\mathfrak{T})$ = $\{\{Route(123, paris, sant), Info(123, 2320, 0815, airFr)\},$
  $\{\{Route(124, paris, sant), Info(124, 2320, 0915, airFr)\},$
  $\ldots,\}$
- $Q_1 = \exists fno\ Route(fno, paris, sant)$ $\quad cert(Q_1, \mathfrak{T}) = \{()\} = yes$
- $Q_2 = Route(123, paris, sant)$ $\quad cert(Q_2, \mathfrak{T}) = \emptyset = no$

## Definition (Certain answers over incomplete DB (formally))

$$cert(Q, \mathfrak{T}) = Q(\mathfrak{T}) = \bigcap_{\mathfrak{T}' \in Rep(\mathfrak{T})} Q(\mathfrak{T}')$$

$Rep(\mathfrak{T})$ = all complete DBs resulting from $\mathfrak{T}$ by substituting marked NULLs (consistently) with constants

## Example (Answer for $\tau$ solution from flight domain)

- $\mathfrak{T} = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\}$

  $Rep(\mathfrak{T})$ = $\{\{Route(123, paris, sant), Info(123, 2320, 0815, airFr)\},$
  $\{\{Route(124, paris, sant), Info(124, 2320, 0915, airFr)\},$
  $\ldots,\}$
- $Q_1 = \exists fno\ Route(fno, paris, sant)$ $\quad cert(Q_1, \mathfrak{T}) = \{()\} = yes$
- $Q_2 = Route(123, paris, sant)$ $\quad cert(Q_2, \mathfrak{T}) = \emptyset = no$

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

Geo( city, coun, pop )

Flight( src, dest, airl, dep )

| | | | |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$ and instance

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow$
   $\exists fno\ \exists arr\ (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

   ...

## Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|

| Flight( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$ and instance

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | sant. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_1$, | 2320, | $\perp_2$, | airFr |

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

   ...

## Many $\tau$ solutions $SOL_{\mathcal{M}}(\mathfrak{S})$

$$\mathfrak{T} = \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_2, airFr)\}$$
$$\mathfrak{T}' = \{Route(\perp_3, paris, sant)\ ,\ Info(\perp_3, 2320, \perp_2, airFr)\}$$
$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_1, airFr)\}$$
$$\mathfrak{T}''' = \{Route(123, paris, sant)\ ,\ Info(123, 2320, \perp_2, airFr)\}\ldots$$

## Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|

| Flight( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$ and instance

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_3$, | paris, | sant. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_3$, | 2320, | $\perp_2$, | airFr |

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

   ...

## Many $\tau$ solutions $SOL_{\mathcal{M}}(\mathfrak{S})$

$$\begin{aligned}
\mathfrak{T} &= \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_2, airFr)\} \\
\mathfrak{T}' &= \{Route(\perp_3, paris, sant)\ ,\ Info(\perp_3, 2320, \perp_2, airFr)\} \\
\mathfrak{T}'' &= \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_1, airFr)\} \\
\mathfrak{T}''' &= \{Route(123, paris, sant)\ ,\ Info(123, 2320, \perp_2, airFr)\} \ldots
\end{aligned}$$

## Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|

| Flight( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$ and instance

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | sant. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_1$, | 2320, | $\perp_1$, | airFr |

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists arr\ (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

   ...

## Many $\tau$ solutions $SOL_{\mathcal{M}}(\mathfrak{S})$

$$\begin{aligned}
\mathfrak{T} &= \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_2, airFr)\} \\
\mathfrak{T}' &= \{Route(\perp_3, paris, sant)\ ,\ Info(\perp_3, 2320, \perp_2, airFr)\} \\
\mathfrak{T}'' &= \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_1, airFr)\} \\
\mathfrak{T}''' &= \{Route(123, paris, sant)\ ,\ Info(123, 2320, \perp_2, airFr)\} \ldots
\end{aligned}$$

## Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|

| Flight( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$ and instance

| Route( fno, | src, | dest ) |
|---|---|---|
| 123, | paris, | sant. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| 123, | 2320, | $\perp_2$, | airFr |

| Serves( airl, | city, | coun, | phone ) |
|---|---|---|---|

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

...

## Many $\tau$ solutions $SOL_{\mathcal{M}}(\mathfrak{S})$

$$
\begin{aligned}
\mathfrak{T} &= \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_2, airFr)\} \\
\mathfrak{T}' &= \{Route(\perp_3, paris, sant)\ ,\ Info(\perp_3, 2320, \perp_2, airFr)\} \\
\mathfrak{T}'' &= \{Route(\perp_1, paris, sant)\ ,\ Info(\perp_1, 2320, \perp_1, airFr)\} \\
\mathfrak{T}''' &= \{Route(123, paris, sant)\ ,\ Info(123, 2320, \perp_2, airFr)\} \ldots
\end{aligned}
$$

# Certain Answers Help Again

- Same idea: Compromise over all possible solutions
- Certain answers w.r.t. mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ and source DB $\mathfrak{S}$

# Certain Answers Help Again

- Same idea: Compromise over all possible solutions
- Certain answers w.r.t. mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ and source DB $\mathfrak{S}$

**Definition (Certain answers (formally))**

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) = \bigcap_{\mathfrak{T} \in Sol_{\mathcal{M}}(\mathfrak{S})} Q(\mathfrak{T})$$

# Certain Answers Help Again

- Same idea: Compromise over all possible solutions
- Certain answers w.r.t. mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ and source DB $\mathfrak{S}$

## Definition (Certain answers (formally))

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) = \bigcap_{\mathfrak{T} \in Sol_{\mathcal{M}}(\mathfrak{S})} Q(\mathfrak{T})$$

## Example (Certain answers in flight domain)

$$\begin{aligned}
\mathfrak{T} &= \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\} \\
\mathfrak{T}' &= \{Route(\bot_3, paris, sant), Info(\bot_3, 2320, \bot_2, airFr)\} \\
\mathfrak{T}'' &= \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_1, airFr)\} \\
\mathfrak{T}''' &= \{Route(123, paris, sant), Info(123, 2320, \bot_2, airFr)\} \ldots
\end{aligned}$$

- $Q_1 = \exists fno\ Route(fno, paris, sant)$ $\quad cert_{\mathcal{M}}(Q_1, \mathfrak{S}) = \{()\} = yes$
- $Q_2 = Route(123, paris, sant)$ $\quad cert_{\mathcal{M}}(Q_2, \mathfrak{S}) = \emptyset = no$

# But wait . . .

- In DE one aims at materializing exactly one $\tau$ solution?
- Is there a single solution $\mathfrak{T}_u$ capturing the certain answers?

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

# But wait . . .

- In DE one aims at materializing exactly one $\tau$ solution?
- Is there a single solution $\mathfrak{T}_u$ capturing the certain answers?

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- Yes! Universal solution
  - Contains facts which are as specific as necessary, i.e., all other solutions more specific
  - Works for CQs = conjunctive queries (= SPJ fragment)
  - Universality fundamental property ubiquitous in CS
    - e.g., most general unifier in resolution
  - If existent, can be constructed by chase procedure

# But wait . . .

- In DE one aims at materializing exactly one $\tau$ solution?
- Is there a single solution $\mathfrak{T}_u$ capturing the certain answers?

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- Yes! Universal solution
  - Contains facts which are as specific as necessary, i.e., all other solutions more specific
  - Works for CQs = conjunctive queries (= SPJ fragment)
  - Universality fundamental property ubiquitous in CS
    - e.g., most general unifier in resolution
  - If existent, can be constructed by chase procedure

# But wait . . .

- In DE one aims at materializing exactly one $\tau$ solution?
- Is there a single solution $\mathfrak{T}_u$ capturing the certain answers?

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- Yes! Universal solution
  - Contains facts which are as specific as necessary, i.e., all other solutions more specific
  - Works for CQs = conjunctive queries (= SPJ fragment)
  - Universality fundamental property ubiquitous in CS
    - e.g., most general unifier in resolution
  - If existent, can be constructed by chase procedure

# But wait . . .

- In DE one aims at materializing exactly one $\tau$ solution?
- Is there a single solution $\mathfrak{T}_u$ capturing the certain answers?

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- Yes! Universal solution
  - Contains facts which are as specific as necessary, i.e., all other solutions more specific
  - Works for CQs = conjunctive queries (= SPJ fragment)
  - Universality fundamental property ubiquitous in CS
    - e.g., most general unifier in resolution
  - If existent, can be constructed by chase procedure

# But wait . . .

- In DE one aims at materializing exactly one $\tau$ solution?
- Is there a single solution $\mathfrak{T}_u$ capturing the certain answers?

$$cert_{\mathcal{M}}(Q, \mathfrak{S}) \stackrel{?}{=} Q(\mathfrak{T}_u)$$

- Yes! Universal solution
  - Contains facts which are as specific as necessary, i.e., all other solutions more specific
  - Works for CQs = conjunctive queries (= SPJ fragment)
  - Universality fundamental property ubiquitous in CS
    - e.g., most general unifier in resolution
  - If existent, can be constructed by chase procedure

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( | city, | coun, | pop ) |
|---|---|---|---|
| Flight( | src, | dest, | airl, dep ) |
| | paris | sant. | airFr 2320 |

**Target schema** $\tau$

Route( fno, src, dest )

Info( fno, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

# Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( | city, | coun, | pop ) |
|---|---|---|---|
| Flight( | src, | dest, | airl, dep ) |
| | paris | sant. | airFr 2320 |

**Target schema** $\tau$

Route( fno, src, dest )

Info( fno, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

# Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

non-necessary co-reference

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

Geo( city, coun, pop )

Flight( src, dest, airl, dep )

| | | | |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

# Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\}$$

non-necessary instantiation

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|

| Flight( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

# Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

Why is $\mathfrak{T}$ universal?

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

Geo( city, coun, pop )

Flight( src, dest, airl, dep )

| | | | |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $\mathit{Flight}(\mathit{src}, \mathit{dest}, \mathit{airl}, \mathit{dep}) \longrightarrow \exists \mathit{fno}\, \exists \mathit{arr}\, (\mathit{Route}(\mathit{fno}, \mathit{src}, \mathit{dest}) \wedge \mathit{Info}(\mathit{fno}, \mathit{dep}, \mathit{arr}, \mathit{airl}))$

# Universal solutions

$\bot_1 \mapsto \bot_3$

$$\mathfrak{T} = \{\mathit{Route}(\bot_1, \mathit{paris}, \mathit{sant}), \mathit{Info}(\bot_1, 2320, \bot_2, \mathit{airFr})\}$$

$$\mathfrak{T}' = \{\mathit{Route}(\bot_3, \mathit{paris}, \mathit{sant}), \mathit{Info}(\bot_3, 2320, \bot_2, \mathit{airFr})\}$$

$$\mathfrak{T}'' = \{\mathit{Route}(\bot_1, \mathit{paris}, \mathit{sant}), \mathit{Info}(\bot_1, 2320, \bot_1, \mathit{airFr})\}$$

$$\mathfrak{T}''' = \{\mathit{Route}(123, \mathit{paris}, \mathit{sant}), \mathit{Info}(123, 2320, \bot_2, \mathit{airFr})\} \ldots$$

Why is $\mathfrak{T}$ universal?

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( | city, | coun, | pop ) |
|---|---|---|---|
| Flight( | src, | dest, | airl, dep ) |
| | paris | sant. | airFr 2320 |

**Target schema** $\tau$

Route( fno, src, dest )

Info( fno, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

# Universal solutions

$$\mathfrak{T} = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\}$$

$\bot_2 \mapsto \bot_1$

$$\mathfrak{T}' = \{Route(\bot_3, paris, sant), Info(\bot_3, 2320, \bot_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \bot_2, airFr)\} \ldots$$

Why is $\mathfrak{T}$ universal?

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

Geo( city, coun, pop )

Flight( src, dest, airl, dep )

| | | | |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$

Route( <u>fno</u>, src, dest )

Info( <u>fno</u>, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

# Universal solutions

$$\mathfrak{T} = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\bot_3, paris, sant), Info(\bot_3, 2320, \bot_2, airFr)\}$$

$\bot_1 \mapsto 123$

$$\mathfrak{T}'' = \{Route(\bot_1, paris, sant), Info(\bot_1, 2320, \bot_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \bot_2, airFr)\} \ldots$$

Why is $\mathfrak{T}$ universal?

# Example (DE in Flight Domain)

**Source schema** $\sigma$ and instance $\mathfrak{S}$

| Geo( city, | coun, | pop ) |
|---|---|---|

| Flight( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | sant. | airFr | 2320 |

**Target schema** $\tau$

Route( fno, src, dest )

Info( fno, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules** $M_{\sigma\tau}$

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$

# Universal solutions

$$\mathfrak{T} = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}' = \{Route(\perp_3, paris, sant), Info(\perp_3, 2320, \perp_2, airFr)\}$$

$$\mathfrak{T}'' = \{Route(\perp_1, paris, sant), Info(\perp_1, 2320, \perp_1, airFr)\}$$

$$\mathfrak{T}''' = \{Route(123, paris, sant), Info(123, 2320, \perp_2, airFr)\} \ldots$$

Any universal solution works: $cert_{\mathcal{M}}(Q, \mathfrak{S}) = Q(\mathfrak{T}) = Q(\mathfrak{T}')$

## Definition (Universal Solution)

A solution $\mathfrak{T}$ for $\mathfrak{S}$ and $\mathcal{M}$ is called a universal solution iff it can be mapped homomorphically into all other solutions.

$$\text{For all } \mathfrak{T}' \in SOL_{\mathcal{M}}(\mathfrak{S}) \text{ there is } h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$$

# Homomorphism

- $CONST(\mathfrak{T})$ = set of all constants in $\mathfrak{T}$
- $VAR(\mathfrak{T})$ = set of all marked nulls in $\mathfrak{T}$

# Homomorphism

- $CONST(\mathfrak{T})$ = set of all constants in $\mathfrak{T}$
- $VAR(\mathfrak{T})$ = set of all marked nulls in $\mathfrak{T}$

## Definition

A homomorphism $h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$ is a map

$$h : Var(\mathfrak{T}) \cup CONST \to VAR(\mathfrak{T}') \cup CONST$$

s.t.

- $h(c) = c$ for all $c \in CONST$ and
- if $R(t_1, \ldots, t_n) \in \mathfrak{T}$, then $R(h(t_1), \ldots, h(t_n)) \in \mathfrak{T}'$ (for all relations $R$)

# Homomorphism

- $CONST(\mathfrak{T})$ = set of all constants in $\mathfrak{T}$
- $VAR(\mathfrak{T})$ = set of all marked nulls in $\mathfrak{T}$

## Definition

A homomorphism $h : \mathfrak{T} \xrightarrow{hom} \mathfrak{T}'$ is a map

$$h : Var(\mathfrak{T}) \cup CONST \to VAR(\mathfrak{T}') \cup CONST$$

s.t.

- $h(c) = c$ for all $c \in CONST$ and
- if $R(t_1, \ldots, t_n) \in \mathfrak{T}$, then $R(h(t_1), \ldots, h(t_n)) \in \mathfrak{T}'$ (for all relations $R$)

See blackboard for (non)examples

# Example (Core in Flight Domain)

**Source schema $\sigma$ and instance**

Geo( city, coun, pop )
paris, france, 2M

Flight ( src, dest, airl, dep )
paris amst. klm 1410
paris amst. klm 2230

**Target schema $\tau$**

Route( fno, src, dest )

Info( fno, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules $M_{\sigma\tau}$**

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists arr\ (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$

# Example (Core in Flight Domain)

**Source schema $\sigma$ and instance**

Geo( city, coun, pop )
paris, france, 2M

Flight ( src, dest, airl, dep )
paris amst. klm 1410
paris amst. klm 2230

**Target schema $\tau$**

Route( fno, src, dest )

Info( fno, dep, arr, airl )

Serves( airl, city, coun, phone )

**Mapping rules $M_{\sigma\tau}$**

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\ \exists arr\ (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\ (Serves(airl, city, coun, phone))$

# Example (Core in Flight Domain)

**Source schema $\sigma$ and instance**

| Geo( city, | coun, | pop ) |
|---|---|---|
| paris, | france, | 2M |

| Flight ( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | amst. | klm | 1410 |
| paris | amst. | klm | 2230 |

**Target schema $\tau$ and universal solution $\mathfrak{T}$**

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | amst. |
| $\perp_3$, | paris, | amst. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_1$, | 1410, | $\perp_2$, | klm |
| $\perp_3$, | 2320, | $\perp_4$, | klm |

| Serves( airl, | city, | coun, | phone ) |
|---|---|---|---|
| klm, | paris, | france, | $\perp_5$ |
| klm, | paris, | france, | $\perp_6$ |

**Mapping rules $M_{\sigma\tau}$**

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

# Example (Core in Flight Domain)

**Source schema $\sigma$ and instance**

| Geo( city, | coun, | pop ) |
|---|---|---|
| paris, | france, | 2M |

| Flight ( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | amst. | klm | 1410 |
| paris | amst. | klm | 2230 |

**Target schema $\tau$ and core solution**

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | amst. |
| $\perp_3$, | paris, | amst. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_1$, | 1410, | $\perp_2$, | klm |
| $\perp_3$, | 2320, | $\perp_4$, | klm |

| Serves( airl, | city, | coun, | phone ) |
|---|---|---|---|
| klm, | paris, | france, | $\perp_5$ |
| ~~klm,~~ | ~~paris,~~ | ~~france,~~ | ~~$\perp_6$~~ |

**Mapping rules $M_{\sigma\tau}$**

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$

# Example (Core in Flight Domain)

**Source schema $\sigma$ and instance**

| Geo( city, | coun, | pop ) |
|---|---|---|
| paris, | france, | 2M |

| Flight ( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | amst. | klm | 1410 |
| paris | amst. | klm | 2230 |

**Target schema $\tau$ and core solution**

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | amst. |
| $\perp_3$, | paris, | amst. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_1$, | 1410, | $\perp_2$, | klm |
| $\perp_3$, | 2320, | $\perp_4$, | klm |

| Serves( airl, | city, | coun, | phone ) |
|---|---|---|---|
| klm, | paris, | france, | $\perp_5$ |
| ~~klm,~~ | ~~paris,~~ | ~~france,~~ | ~~$\perp_6$~~ |

**Mapping rules $M_{\sigma\tau}$**

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep)$ )

Why not delete similarly $Route(\perp_3, paris, amst)$?

# Example (Core in Flight Domain)

**Source schema $\sigma$ and instance**

| Geo( city, | coun, | pop ) |
|---|---|---|
| paris, | france, | 2M |

| Flight ( src, | dest, | airl, | dep ) |
|---|---|---|---|
| paris | amst. | klm | 1410 |
| paris | amst. | klm | 2230 |

**Target schema $\tau$ and core solution**

| Route( fno, | src, | dest ) |
|---|---|---|
| $\perp_1$, | paris, | amst. |
| $\perp_3$, | paris, | amst. |

| Info( fno, | dep, | arr, | airl ) |
|---|---|---|---|
| $\perp_1$, | 1410, | $\perp_2$, | klm |
| $\perp_3$, | 2320, | $\perp_4$, | klm |

| Serves( airl, | city, | coun, | phone ) |
|---|---|---|---|
| klm, | paris, | france, | $\perp_5$ |
| ~~klm,~~ | ~~paris,~~ | ~~france,~~ | ~~$\perp_6$~~ |

**Mapping rules $M_{\sigma\tau}$**

1. $Flight(src, dest, airl, dep) \longrightarrow \exists fno\, \exists arr\, (Route(fno, src, dest) \wedge Info(fno, dep, arr, airl))$
2. $Flight(city, dest, airl, dep) \wedge Geo(city, coun, pop) \longrightarrow \exists phone\, (Serves(airl, city, coun, phone))$
3. $Flight(src, city, airl, dep)$

Why not delete similarly $Route(\perp_3, paris, amst)$?

There are additional facts distinguishing $\perp_1$ and $\perp_3$

Identification $\perp_1 = \perp_3$ would violate primary key constraint

# Core Solution vs. Universal Solution

- Core solutions contain less redundant information and are unique
- but are harder to construct (next lecture)

# Core Solution vs. Universal Solution

- Core solutions contain less redundant information and are unique
- but are harder to construct (next lecture)

- Which one to use?
  - Aim "only" answering CQs $\Longrightarrow$ universal solution
  - Aim goes further $\Longrightarrow$ core solution
    - Need to query with more expressive language (negation, counting)
    - Need to calculate sufficient statistics in an ML algorithm

# Outlook

- Next lecture
  - Formalization of mappings
  - Algorithmic problem of testing for solutions
  - Chase algorithm for constructing solutions
- Thereafter
  - Universal solutions and cores
  - Certain Answers
  - Rewriting

# Testing for and Constructing Solutions

# Relational Mappings

- Going to deal mainly with relational mappings, i.e. mappings for relational DBs
- Relational DB (Codd 1970) very successful and still highly relevant
- There were other opinions...

> "Some of the ideas presented in the paper are interesting and may be of some use, but, in general, this very preliminary work fails to make a convincing point as to their implementation, performance, and practical usefulness. The paper's general point is that the tabular form presented should be suitable for general data access, but I see two problems with this statement: expressivity and efficiency. [...] The formalism is needlessly complex and mathematical, using concepts and notation with which the average data bank practitioner is unfamiliar." Cited according to (Santini 2005)

**Lit:** E. F. Codd. A relational model of data for large shared data banks. Commun. ACM, 13(6):377–387, June 1970.

**Lit:** S. Santini. We are sorry to inform you ... Computer, December 2005.

# Relational Mappings Formally

## Definition

A relational mapping $\mathcal{M}$ is a tuple of the form

$$\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$$

where

- $\sigma$ is the source schema
- $\tau$ is the target schema with all relation symbols different from those in $\sigma$
- $M_{\sigma\tau}$ is a finite set of FOL formulae over $\sigma \cup \tau$ called source-to-target dependencies
- $M_\tau$ is a set of constraints on the target schema called target dependencies

# DB Instances of Schemata

- Schemata are relational signatures
- Concrete database instance
  - For a given schema $\sigma$ a concrete DB instance is a $\sigma$ FOL structure with active domain
  - Active domain: Domain contains all and only individuals (also called constants) occurring in relations
  - Usually: All source instances are concrete DBs

- Generalized DB instances
  - For some attributes in target schema (Example: flight number fno) no corresponding attribute in source may exist
  - Next to constants CONST allow disjoint set of marked NULLs, denoted VAR
  - A generalized DB instance may contain elements from CONST $\cup$ VAR
  - "Incomplete" DB

# DB Instances of Schemata

- Schemata are relational signatures
- Concrete database instance
  - For a given schema $\sigma$ a concrete DB instance is a $\sigma$ FOL structure with active domain
  - Active domain: Domain contains all and only individuals (also called constants) occurring in relations
  - Usually: All source instances are concrete DBs

- Generalized DB instances
  - For some attributes in target schema (Example: flight number fno) no corresponding attribute in source may exist
  - Next to constants CONST allow disjoint set of marked NULLs, denoted VAR
  - A generalized DB instance may contain elements from CONST $\cup$ VAR
  - "Incomplete" DB

# Source-Target-Dependencies $M_{\sigma\tau}$

- Source-Target-Dependencies may be arbitrary FOL formula
- Usually they have a simple directed form (decidability!)

**Definition**

A source-to-target tuple-generating dependencies (st-tgds) is a FOL formula of the form

$$\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, \psi_\tau(\vec{x},\vec{z}))$$

where

- $\phi_\sigma$ is a conjunction of atoms over source schema $\sigma$
- $\psi_\tau$ is a conjunction of atoms over target schema $\tau$

- So in particular, antecedens is a conjunctive query (CQ)
- CQs "well-behaved"

# Source-Target-Dependencies $M_{\sigma\tau}$

- Source-Target-Dependencies may be arbitrary FOL formula
- Usually they have a simple directed form (decidability!)

**Definition**

A source-to-target tuple-generating dependencies (st-tgds) is a FOL formula of the form

$$\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, \psi_\tau(\vec{x},\vec{z}))$$

where

- $\phi_\sigma$ is a conjunction of atoms over source schema $\sigma$
- $\psi_\tau$ is a conjunction of atoms over target schema $\tau$

- So in particular, antecedens is a conjunctive query (CQ)
- CQs "well-behaved"

# Reminder: Conjunctive Queries (CQs)

- Class of sufficiently expressive and feasible FOL queries of form

$$ans(\vec{x}) = \exists \vec{y} \left( \alpha_1(\vec{x_1}, \vec{y_1}) \wedge \cdots \wedge \alpha_n(\vec{x_n}, \vec{y_n}) \right)$$

  where
  - $\alpha_i(\vec{x_i}, \vec{y_i})$ are atomic FOL formula and
  - $\vec{x_i}$ variable vectors among $\vec{x}$ and $\vec{y_i}$ variables among $\vec{y}$
- Corresponds to SELECT-PROJECT-JOIN Fragment of SQL

**Example (Conjunctive Query from Flight Domain)**

$$ans(src, dest, airl, dep) = \exists fno \; \exists \; arr(Routes(fno, src, dest) \wedge Info(fno, dep, arr, airl))$$

# Reminder: Conjunctive Queries (CQs)

**Theorem**

- *Answering CQs is NP-complete w.r.t. combined complexity (Chandra,Merlin 1977)*
- *Subsumption test for CQs is NP complete*
- *Answering CQs is in $AC^0$ (and thus in P) w.r.t. data complexity*

**Lit:** A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In: Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, STOC'77, pages 77–90, New York, NY, USA, 1977. ACM.

## Wake-Up Question

Are st-tgds Datalog rules?

## Wake-Up Question

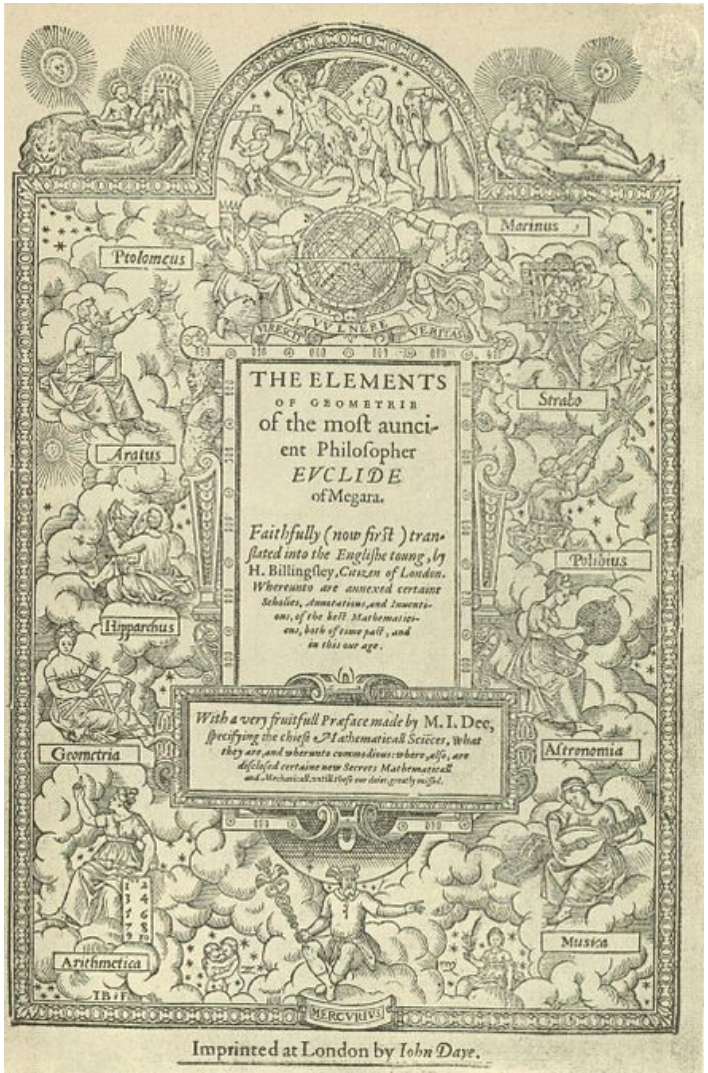
Are st-tgds Datalog rules?

- No, as Datalog rules do not allow existentials in the head of the query
- But there is the extended logic called Datalog$^{+/-}$
  - Has been investigated in last years also in context of ontology-based data access (see net lectures)
  - Provides many interesting sub-fragments

**Lit:** A. Calì, G. Gottlob, and T. Lukasiewicz. Datalog+/-: A unified approach to ontologies and integrity constraints. In Proceedings of the 12th International Conference on Database Theory, pages 14–30. ACM Press, 2009.

# Prominent Tuple Generating Dependencies

- Theorems of Euclids "Elements" expressible as tuple generating dependencies




**Lit:** J. Avigad, E. Dean, J. Mumma: "A Formal System for Euclid's Elements", The Review of Symbolic Logic, 2009

# Target Dependencies $M_\tau$

- These define constraints on target schema known also from classical DB theory
- Two different types of dependencies are sufficiently general to capture the classical DB constraints

**Definition**

A tuple-generating dependency (tgd) is a FOL formula of the form

$$\forall \vec{x}\vec{y}(\phi(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, \psi(\vec{x},\vec{z}))$$

where $\phi, \psi$ are conjunctions of atoms over $\tau$.

An equality-generating (egd) is a FOL formula of the form

$$\forall \vec{x}(\phi(\vec{x}) \longrightarrow x_i = x_j)$$

where $\phi(\vec{x})$ is a conjunction of atoms over $\tau$ and $x_i, x_j$ occur in $\vec{x}$.

# Semantics: Solutions

**Definition**

Given: a mapping $\mathcal{M}$ and a $\sigma$ instance $\mathfrak{S}$

A $\tau$ instance $\mathfrak{T}$ is called a solution for $\mathfrak{S}$ under $\mathcal{M}$ iff $(\mathfrak{S}, \mathfrak{T})$ satisfies all rules in $M_{\sigma\tau}$ (for short: $(\mathfrak{S}, \mathfrak{T}) \models M_{\sigma\tau}$) and $\mathfrak{T}$ satisfies all rules in $M_\tau$.

- $(\mathfrak{S}, \mathfrak{T}) \models M_{\sigma\tau}$ iff $\mathfrak{S} \cup \mathfrak{T} \models M_{\sigma\tau}$ where
  - $\mathfrak{S} \cup \mathfrak{T}$ is the union of the instances $\mathfrak{S}, \mathfrak{T}$: Structure containing all relations from $\mathfrak{S}$ and $\mathfrak{T}$ with domain the union of domains of $\mathfrak{S}$ and $\mathfrak{T}$
  - well defined because schemata are disjoint
- $Sol_{\mathcal{M}}(\mathfrak{S})$: Set of solutions for $\mathfrak{S}$ under $\mathcal{M}$

# First Key Problem: Existence of Solutions

**Problem: SOLEXISTENCE$_{\mathcal{M}}$**

Input: Source instance $\mathfrak{S}$
Output: Answer whether there exists a solution for $\mathfrak{S}$ under $\mathcal{M}$

- Note: $\mathcal{M}$ is assumed to be fixed $\Longrightarrow$ data complexity
- This problem is going to be approached with a well known proof tool: chase

# Trivial Case: No Target Dependencies

- Without target constraints there is always a solution

**Proposition**

*Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau})$ with $M_{\sigma\tau}$ consisting of st-tgds. Then for any source instance $\mathfrak{S}$ there are infinitely many solutions and at least one solution can be constructed in polynomial time.*

# Trivial Case: No Target Dependencies

- Without target constraints there is always a solution

**Proposition**

*Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau})$ with $M_{\sigma\tau}$ consisting of st-tgds. Then for any source instance $\mathfrak{S}$ there are infinitely many solutions and at least one solution can be constructed in polynomial time.*

Proof Idea

- For every rule and every tuple $\vec{a}$ fulfilling the antecedens generate facts according to the succedens (using fresh named nulls for the existentially quantified variables)
- Resulting $\tau$ instance $\mathfrak{T}$ is a solution
- Polynomial: Testing whether $\vec{a}$ fulfills the head (a conjunctive query) can be done in polynomial time
- Infinity: From $\mathfrak{T}$ can build any other solution by extension

# Undecidability for General Constraints

**Theorem**

*There is a relational mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ such that SOLEXISTENCE$_\mathcal{M}$ is undecidable.*

- Proof by reduction from embedding problem for finite semigroups which is known to be undecidable (Arenas et al. 2014, Thm 5.3)
- As a consequence: Further restrict mapping rules
- But note that the following chase construction defined for arbitrary st-tgds

# Undecidability for General Constraints

**Theorem**

*There is a relational mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ such that SOLEXISTENCE$_\mathcal{M}$ is undecidable.*

**Wake-Up Question**

As another exercise in reduction prove the following corollary: There is a relational mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau})$ with a single FOL dependency in $M_{\sigma\tau}$ s.t. SOLEXISTENCE$_\mathcal{M}$ is undecidable

# Undecidability for General Constraints

## Theorem

*There is a relational mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ such that SOLEXISTENCE$_\mathcal{M}$ is undecidable.*

## Wake-Up Question

As another exercise in reduction prove the following corollary: There is a relational mapping $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau})$ with a single FOL dependency in $M_{\sigma\tau}$ s.t. SOLEXISTENCE$_\mathcal{M}$ is undecidable

Proof

- Assume otherwise
- Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$
- Construct $\mathcal{M}' = (\sigma, \tau, \{\chi\})$ with $\chi = \bigwedge(M_{\sigma\tau} \cup M_\tau)$

# Existence Proof vs. Construction

- Proposition above showed existence of solution
- Showing existence $\neq$ construction of a verifier
- Actually we are going to construct a solution using the chase

- Interesting debate in philosophy of mathematics whether non-constructive proofs are acceptable
- Mathematical Intuitionism: field allowing only constructive proofs
  - truth = provable = constructively provable
  - Classical logical inference rules s.a. $\neg\neg A \vDash A$ not allowed

- L.E.J. Brouwer (1881 to 1966)
  - Guru of intuitionism
  - Irony of history: Proved many interesting results in classical (non-constructive) mathematics (Brouwer's fixed point theorem)

# The Chase

# Chase Construction

- A widely used tool in DB theory
- Original use: Calculating entailments of DB constraints

  **Lit:** D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. ACM Trans. Database Syst., 4(4):455–469, Dec. 1979.

- Idea
  - Apply tgds as completion/repair rules in a bottom-up strategy
  - until no tgds can be applied anymore
  - Chase construction mail fail if one of the egds is violated

- The chase leads to an instance with desirable properties
  - It produces not too many redundant facts
  - Universality

## Example (Terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$
- Going to build stepwise potential target instances $\mathfrak{T}_i$ considering pairs $(\mathfrak{S}, \mathfrak{T}_i)$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (termination)

## Example (Terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$
- Going to build stepwise potential target instances $\mathfrak{T}_i$ considering pairs $(\mathfrak{S}, \mathfrak{T}_i)$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp)\})$ (termination)

## Example (Terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$
- Going to build stepwise potential target instances $\mathfrak{T}_i$ considering pairs $(\mathfrak{S}, \mathfrak{T}_i)$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp)\})$ (termination)

## Example (Terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$
- Going to build stepwise potential target instances $\mathfrak{T}_i$ considering pairs $(\mathfrak{S}, \mathfrak{T}_i)$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp)\})$ (termination)

## Example (Terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$
- Going to build stepwise potential target instances $\mathfrak{T}_i$ considering pairs $(\mathfrak{S}, \mathfrak{T}_i)$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp)\})$ (termination)

## Example (Terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$
- Going to build stepwise potential target instances $\mathfrak{T}_i$ considering pairs $(\mathfrak{S}, \mathfrak{T}_i)$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp)\})$ (termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$)
- ... (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$)
- ... (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$)
- ... (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$)
- ... (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp)\})$ (violates $\chi_2$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp), G(\perp,\perp_1)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\perp), G(\perp,\perp_1), L(\perp_1,\perp_2)\})$ (violates $\chi_2$)
- ... (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$ )
- . . . (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$ )
- . . . (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$ )
- . . . (non-termination)

## Example (Non-terminating c(h)ase)

- Source schema $\sigma = \{E\}$; target schema $\tau = \{G, L\}$
- $M_{\sigma\tau} = \{ \underbrace{E(x,y) \to G(x,y)}_{\theta_1} \}$

  $M_\tau = \{ \underbrace{G(x,y) \to \exists z\ L(y,z)}_{\chi_1}, \underbrace{L(x,y) \to \exists z\ G(y,z)}_{\chi_2} \}$
- Source instance $\mathfrak{S} = \{E(a,b)\}$

- $(\mathfrak{S}, \emptyset)$ (violates $\theta_1$)
- $(\mathfrak{S}, \{G(a,b)\})$ (violates $\chi_1$)
- $(\mathfrak{S}, \{G(a,b), L(b,\bot)\})$ (violates $\chi_2$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1)\})$ (violates $\chi_1$ )
- $(\mathfrak{S}, \{G(a,b), L(b,\bot), G(\bot,\bot_1), L(\bot_1,\bot_2)\})$ (violates $\chi_2$ )
- ... (non-termination)

# Chase Definition

- Let $\mathfrak{S}$ be a $\sigma$ instance and $dom(\mathfrak{S})$ its domain

## Definition (Chase steps)

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathfrak{S}'$ iff

1. $\chi$ a tgd of form $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$
   - $\mathfrak{S}'$ extends $\mathfrak{S}$ with all atoms occurring in $\psi(\vec{a}, \vec{\perp})$.
2. or $\chi$ is an egd of form $\phi(\vec{x}) \rightarrow x_i = x_j$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$ and
   - ( $a_i$ is constant or null, $a_j$ is null and $\mathfrak{S}' = \mathfrak{S}[a_j/a_i]$ or
   - $a_i$ is null, $a_j$ is constant and $\mathfrak{S}' = \mathfrak{S}[a_i/a_j]$ )

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathit{fail}$ iff

- $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$
- and both $a_i, a_j$ are constants.

# Chase Definition

- Let $\mathfrak{S}$ be a $\sigma$ instance and $dom(\mathfrak{S})$ its domain

## Definition (Chase steps)

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathfrak{S}'$ iff

1. $\chi$ a tgd of form $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$
   - $\mathfrak{S}'$ extends $\mathfrak{S}$ with all atoms occurring in $\psi(\vec{a}, \vec{\perp})$.
2. or $\chi$ is an egd of form $\phi(\vec{x}) \rightarrow x_i = x_j$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$ and
   - ( $a_i$ is constant or null, $a_j$ is null and $\mathfrak{S}' = \mathfrak{S}[a_j/a_i]$ or
   - $a_i$ is null, $a_j$ is constant and $\mathfrak{S}' = \mathfrak{S}[a_i/a_j]$ )

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathit{fail}$ iff

- $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$
- and both $a_i, a_j$ are constants.

# Chase Definition

- Let $\mathfrak{S}$ be a $\sigma$ instance and $dom(\mathfrak{S})$ its domain

## Definition (Chase steps)

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathfrak{S}'$ iff

1. $\chi$ a tgd of form $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$
   - $\mathfrak{S}'$ extends $\mathfrak{S}$ with all atoms occurring in $\psi(\vec{a}, \vec{\perp})$.
2. or $\chi$ is an egd of form $\phi(\vec{x}) \rightarrow x_i = x_j$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$ and
   - ( $a_i$ is constant or null, $a_j$ is null and $\mathfrak{S}' = \mathfrak{S}[a_j/a_i]$ or
   - $a_i$ is null, $a_j$ is constant and $\mathfrak{S}' = \mathfrak{S}[a_i/a_j]$ )

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathit{fail}$ iff

- $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$
- and both $a_i, a_j$ are constants.

# Chase Definition

- Let $\mathfrak{S}$ be a $\sigma$ instance and $dom(\mathfrak{S})$ its domain

## Definition (Chase steps)

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathfrak{S}'$ iff

1. $\chi$ a tgd of form $\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$
   - $\mathfrak{S}'$ extends $\mathfrak{S}$ with all atoms occurring in $\psi(\vec{a}, \vec{\perp})$.
2. or $\chi$ is an egd of form $\phi(\vec{x}) \rightarrow x_i = x_j$ and
   - $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$ and
   - ( $a_i$ is constant or null, $a_j$ is null and $\mathfrak{S}' = \mathfrak{S}[a_j/a_i]$ or
   - $a_i$ is null, $a_j$ is constant and $\mathfrak{S}' = \mathfrak{S}[a_i/a_j]$ )

$\mathfrak{S} \overset{\chi,\vec{a}}{\rightsquigarrow} \mathit{fail}$ iff

- $\mathfrak{S} \models \phi(\vec{a})$ for some elements $\vec{a}$ from $dom(\mathfrak{S})$ with $a_i \neq a_j$
- and both $a_i, a_j$ are constants.

# Chase

## Definition

A chase sequence for $\mathfrak{S}$ under $M$ is a sequence of chase steps $\mathfrak{S}_i \overset{\chi_i, \vec{a}_i}{\rightsquigarrow} \mathfrak{S}_{i+1}$ such that

- $\mathfrak{S}_0 = \mathfrak{S}$
- each $\chi_i$ is in $M$
- for each distinct $i, j$ also $(\chi_i, \vec{a}_i) \neq (\chi_j, \vec{a}_j)$

For a finite chase sequence the last instance is called its result.

- If the result is *fail*, then the sequence is said to be a failing sequence
- If no further dependency from $M$ can be applied to a result, then the sequence is called successful.

# Indeterminism in Chase Construction

- Indeterminism regarding choice of nulls (no problem)
- Indeterminism regarding order of chosen tgds and egds

  This may lead to different chase results

# Use of Chases in Data Exchange

- A chase sequence for $\mathfrak{S}$ under a $\mathcal{M}$ is a chase sequence for $(\mathfrak{S}, \emptyset)$ under $M_{\sigma\tau} \cup M_{\tau}$
- If $(\mathfrak{S}, \mathfrak{T})$ result of a finite sequence, call just $\mathfrak{T}$ the result

- Chase is the right tool for finding solutions

**Proposition**

*Given $\mathcal{M}$ and source instance $\mathfrak{S}$.*

- *If there is a successful chase sequence for $\mathfrak{S}$ with result $\mathfrak{T}$, then $\mathfrak{T}$ is a solution.*
- *If there is a failing chase sequence for $\mathfrak{S}$, then $\mathfrak{S}$ has no solution.*

# Use of Chases in Data Exchange

- A chase sequence for $\mathfrak{S}$ under a $\mathcal{M}$ is a chase sequence for $(\mathfrak{S}, \emptyset)$ under $M_{\sigma\tau} \cup M_{\tau}$
- If $(\mathfrak{S}, \mathfrak{T})$ result of a finite sequence, call just $\mathfrak{T}$ the result

- Chase is the right tool for finding solutions

**Proposition**

*Given $\mathcal{M}$ and source instance $\mathfrak{S}$.*

- *If there is a successful chase sequence for $\mathfrak{S}$ with result $\mathfrak{T}$, then $\mathfrak{T}$ is a solution.*
- *If there is a failing chase sequence for $\mathfrak{S}$, then $\mathfrak{S}$ has no solution.*

- The proposition does no cover all cases: non-terminating chase
- In this case still there still may be a solution

# Weak Acyclicity

- In order to guarantee termination restrict target constraints
- Reason for non-termination: generation of new nulls with same dependencies

## Example (Cycle in Dependencies)

- $\chi_1 = G(x, y) \rightarrow \exists z\ L(y, z)$
- $\chi_2 = L(x, y) \rightarrow \exists z\ G(y, z)$

Possible infinite generation

$$G(a, b) \overset{\chi_1}{\rightsquigarrow} L(b, \perp_1) \overset{\chi_2}{\rightsquigarrow} G(\perp_1, \perp_2) \overset{\chi_1}{\rightsquigarrow} L(\perp_2, \perp_3) \ldots$$

- Problem caused by cycle in dependencies

# Weak Acyclicity

- In order to guarantee termination restrict target constraints
- Reason for non-termination: generation of new nulls with same dependencies

**Example (Cycle in Dependencies)**

- $\chi_1 = G(x, y) \rightarrow \exists z\ L(y, z)$
- $\chi_2 = L(x, y) \rightarrow \exists z\ G(y, z)$

Possible infinite generation

$$G(a, b) \overset{\chi_1}{\rightsquigarrow} L(b, \perp_1) \overset{\chi_2}{\rightsquigarrow} G(\perp_1, \perp_2) \overset{\chi_1}{\rightsquigarrow} L(\perp_2, \perp_3) \dots$$

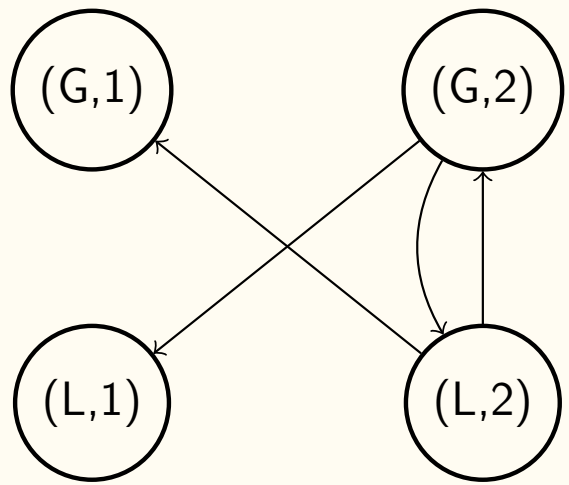
- Problem caused by cycle in dependencies

# Simple Dependency Graphs

- Nodes: pairs $(R, i)$ of predicate $R$ and argument-position $i$
- Edges: From $(R_b, i)$ to $(R_h, j)$ iff there is a tgd $\forall \vec{x} \forall \vec{y} \phi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \psi(\vec{x}, \vec{z})$ and
  1. $R_h$ occurs in $\psi$ and $R_b$ occurs in $\phi$ and
  2. for all $x \in \vec{x}$ in i-position in $R_b$
     - either $x$ occurs in $j$-position in $R_h$
     - or the variable in $j$-position in $R_h$ is existentially quantified

## Example (Simple Dependency Graph with Cycle)

- $\chi_1 = G(y, x) \rightarrow \exists z\ L(x, z)$
- $\chi_2 = L(y, x) \rightarrow \exists z\ G(x, z)$



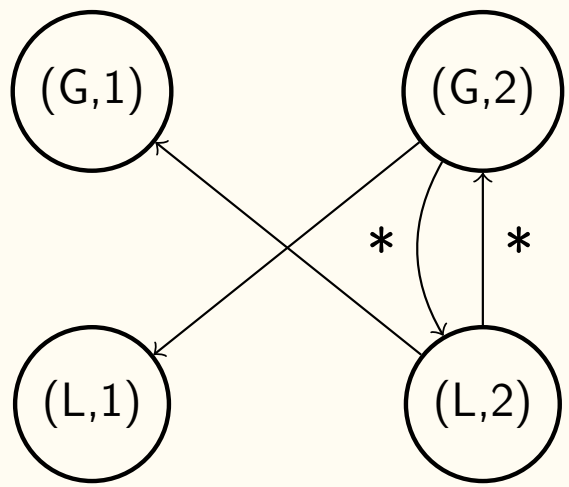

Set of tgds called acyclic if simple dependency graph is acyclic.

# Dependency Graphs (DG)

- Nodes: pairs $(R, i)$ of predicate $R$ and argument-position $i$
- Edges: From $(R_b, i)$ to $(R_h, j)$ iff there is a tgd $\forall\vec{x}\forall\vec{y}\phi(\vec{x},\vec{y}) \rightarrow \exists\vec{z}\psi(\vec{x},\vec{z})$ and
  1. $R_h$ occurs in $\psi$ and $R_b$ occurs in $\phi$ and
  2. for all $x \in \vec{x}$ in i-position in $R_b$
     - either $x$ occurs in $j$-position in $R_h$
     - or the variable in $j$-position in $R_h$ is existentially quantified and these are labelled by *

**Example (Not weakly acyclic Dependency Graph)**

- $\chi_1 = G(y,x) \rightarrow \exists z\ L(x,z)$
- $\chi_2 = L(y,x) \rightarrow \exists z\ G(x,z)$




TGDs weakly acyclic iff DG has no cycle with a * edge.

# Termination for weakly acyclic tgds

**Theorem**

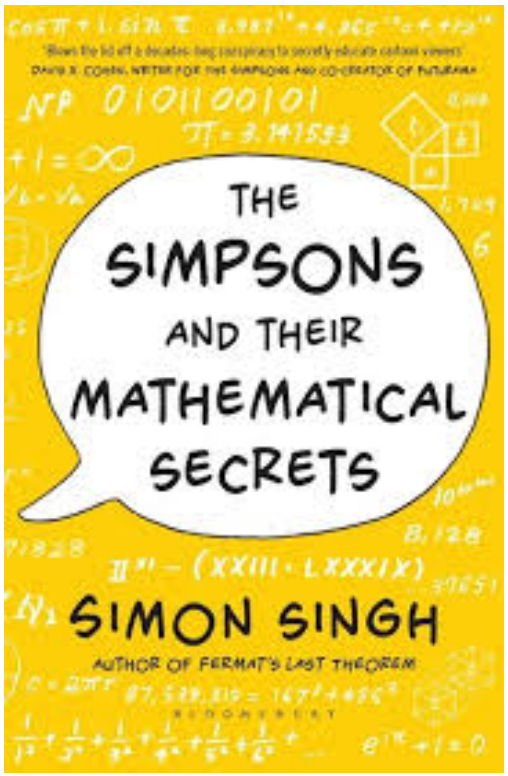
*Let $\mathcal{M} = (\sigma, \tau, M_{\sigma\tau}, M_\tau)$ be a mapping where $M_\tau$ is the union of egds and weakly acyclic tgds. Then the length of every chase sequence for a source $\mathfrak{S}$ is polynomially bounded w.r.t. the size of $\mathfrak{S}$.*

- In particular: Every chase sequence terminates
- Moreover: SOLEXISTENCE$_\mathcal{M}$ can be solved in polynomial time
- a solution can be constructed in polynomial time

# “What is 3 Q plus 7 Q?”

# “What is 3 Q plus 7 Q?”

. . . for you attention! ☺



# Solutions to Exercise 4 (16 Points)

# Solution to Exercise 4.1 (6 Points)

Use Hanf locality in order to proof that the following boolean queries are not FOL-definable: 1. graph acyclicity, 2. tree.

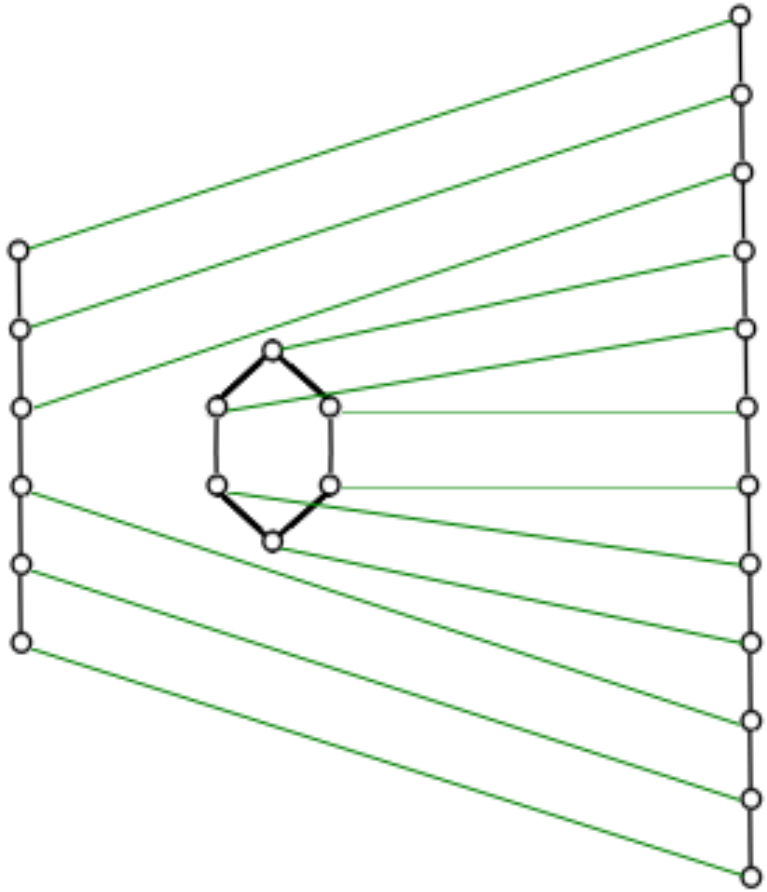
## Solution

Graph Acyclicity (GA).

- For contradiction assume GA is Hanf-local with parameter $r'$. Choose $r = 2r' + 2$
- Let $\mathfrak{G}$ be the disjoint union of a circle of length $r$ and a linear order of length $r$
- Let $\mathfrak{G}'$ be an order of length $2r$.
- Take a bijection $f : \mathfrak{G} \to \mathfrak{G}'$ where
  - the circle is unravelled to the middle of $\mathfrak{G}'$.
  - The lower half part of the order in $\mathfrak{G}$ is mapped to the lower part of $\mathfrak{G}'$
  - The upper half part of the order in $\mathfrak{G}$ is mapped to the upper part of $\mathfrak{G}'$
- an $r'$-neighbourhood of any $a$ in $\mathfrak{G}$ and $f(a) \in \mathfrak{G}'$ is the same: if $a$ is from the circle in $\mathfrak{G}$ then the $r'$-neighbourhood is a 2r'-line and the same for $f(a)$. If $a$ is an element from the line in $\mathfrak{G}$ then in its $r'$-neighbourhood it has to the left and to right the same number of elements as has $f(a)$ in its r'-neigbourhood in $\mathfrak{G}'$.
- Hence $\mathfrak{G} \rightleftarrows_{r'} \mathfrak{G}'$, but: $\mathfrak{G}$ is cyclic and $\mathfrak{G}$ is not. ↯

Tree

- Same construction (as $\mathfrak{G}'$ is tree whereas $\mathfrak{G}$ is not)



# Solution to Exercise 4.2 (4 Points)

Show that $EVEN(\sigma)$ can be defined within second-order logic for any $\sigma$.

Hint: formalize "There is a binary relation which is an equivalence relation having only equivalence classes with exactly two elements" and argue why this shows the axiomatizability.

**Solution**

$$\begin{aligned}\exists R \quad & \forall x R(x,x) \wedge \\ & \forall x \forall y R(x,y) \rightarrow R(y,x) \wedge \\ & \forall x \forall y \forall z ((R(x,y) \wedge R(y,z)) \rightarrow R(x,z)) \wedge \\ & \forall x \exists y (R(x,y) \wedge x \neq y \wedge \forall z (R(x,z) \rightarrow z = x \vee z = y))\end{aligned}$$

Note that $R$ is a quantified variable (!). So we have shown that $EVEN[\emptyset]$ can be defined.

# Solution to Exercise 4.3 (2 Points)

Argue why (in particular within the DB community) one imposes safety conditions for Datalog rules.

**Solution**

- Unsafe negation would lead to infinite answer sets (if domain is infinite.)
- Variables occurring only in head would lead to domain dependence. For example, for $ans(x) \leftarrow R(a)$ all bindings for $x$ in the domain of a DB where $R(a)$ is contained, would have to be in the set of answers. So the answer would not depend only on $R(a)$, i.e., only on the query, but also on the domain of the variables one allows.

# Solution to Exercise 4.4 (4 points)

Give examples of general program rules for which

1. No fixed point exists at all (Hint: "This sentence is not true")
2. Has two minimal fixed points (Hint: "The following sentence is false. The previous sentence is true.")

**Solution** We consider propositional variables as 0-ary predicates. An extension of a propositional variable is then either the empty set $\emptyset$ which is interpreted as the truth value false, for short 0, or is the set consisting of the empty tuple $\{()\}$ which is interpreted as the truth value true, for short 1. Truthvalue assignments $\nu$ can be identified by the set of propositional variables which are assigned the value 1. So, e.g., $\nu(p) = 1, \nu(q)$ is represented by $\{p\}$, whereas $\nu(p) = 1, \nu(q) = 1$ is represented by $\{p, q\}$. So minimality on models becomes minimality w.r.t. set inclusion.

- No fixed point: $p \leftarrow \neg p$
- Two minimal fixed points.

$$q \leftarrow \neg p$$
$$p \leftarrow \neg q$$

Has minimal fixed points $\{p\}$ and $\{q\}$.

# Exercise 5 (16 points)

# Exercise 5.1 (6 Points)

A tuple-generating dependency (tgd) is a FOL formula of the form

$$\forall \vec{x}\vec{y}(\phi(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, \psi(\vec{x},\vec{z}))$$

where $\phi, \psi$ are conjunctions of atoms over $\tau$.

1. Foxy asserts that one can also use existentials in the body (that is in the part before the $\longrightarrow$) without adding expressivity. Is he right? Argue for your answer!
2. Smarty asserts that one can safely assume that $\psi$ is just an atom rather than a conjunction of atoms. Is he right? Argue for your answer!
3. Dumby asserts similarly that one can safely assume that $\phi$ is just an atom rather than a conjunction of atoms. Is he right? Argue for your answer!

# Exercise 5.2 (4 Points)

Prove the folklore proposition that conjunctive queries are preserved under homomorphisms, i.e., show that if there is a homomorphism $h$ from a DB instance $\mathfrak{T}$ to a DB instance $\mathfrak{T}'$, then for any CQ $\phi(\vec{x})$:

$$\{h(\vec{d}) \mid \vec{d} \in ans(\phi(\vec{x}), \mathfrak{T})\} \subseteq ans(\phi(\vec{x}), \mathfrak{T}')$$

# Exercise 5.3 (6 Points)

1. Testing subsumption of CQs is in NP. What about the subsumption of arbitrary FOLs?
2. Show that every CQ (under the usual set semantics) is monotonic and satisfiable.
3. The semantics of CQs is given by the usual FOL set semantics. Inform yourself on the so-called multi-set semantics (aka: bag semantics) for queries and answer the following questions:
   - 3.1 Why should one consider multi-set semantics—in particular if one is interested in SQL queries?
   - 3.2 What does one know about the (complexity of the) sumbsumption test for CQs under multi-set semantics?