UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Data Integration

Lecture 5
7 May 2020

Informationssysteme CS4130
(Summer 2020)

References

- Textbook on data integration (in German)

  Lit: U. Leser and F. Naumann. Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. Dpunkt-Verl., Heidelberg, 2007.

- Another newer textbook

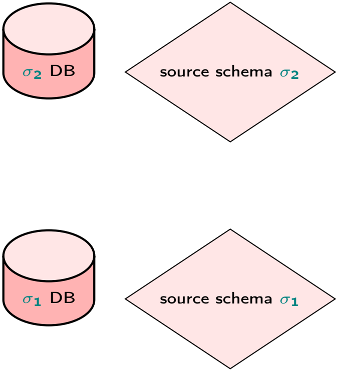
  Lit: A. Doan, A. Halevy, and Z. Ives. Principles of Data Integration. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012.

  Slides `https://research.cs.wisc.edu/dibook/`

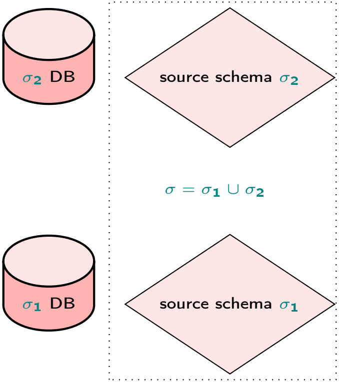
- 2015 Course by L. Libkin on Data integration and Exchange `http://homepages.inf.ed.ac.uk/libkin/teach/dataintegr15/`
- PODS 2002 tutorial by Lenzerini on data integration `http://www.dis.uniroma1.it/~lenzerin/homepagine/talks/TutorialPODS02.pdf`

Data Integration: Motivation

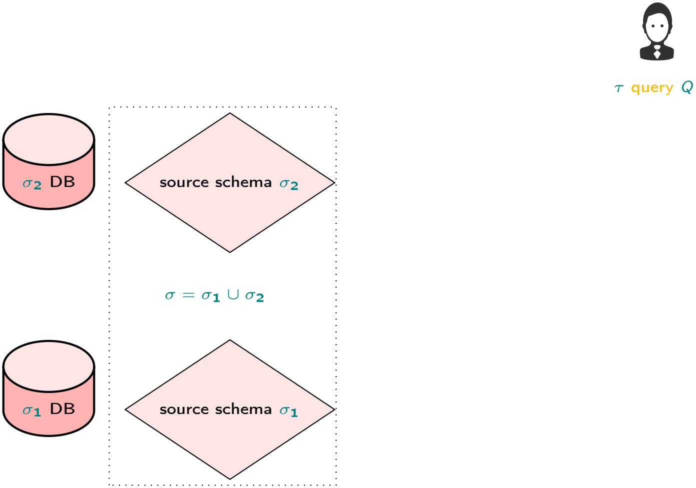
Data Integration (DI): Main Setting



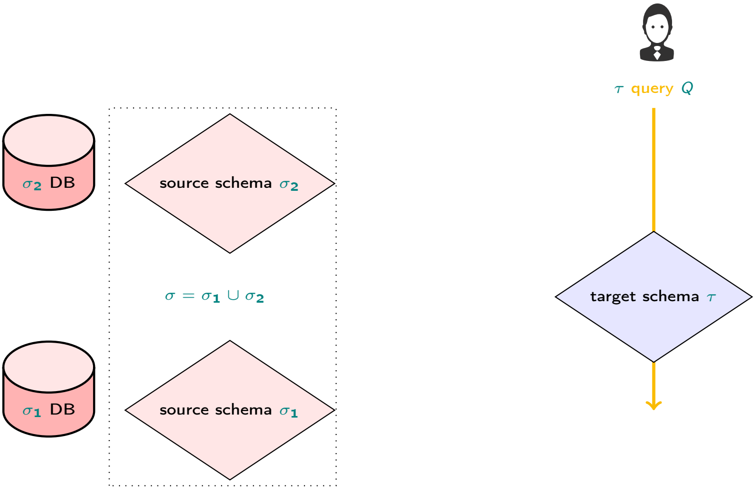
Data Integration (DI): Main Setting



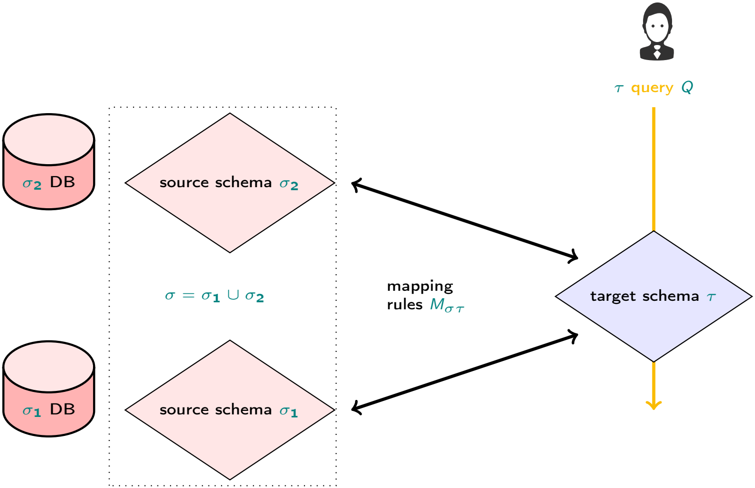
Data Integration (DI): Main Setting



Data Integration (DI): Main Setting



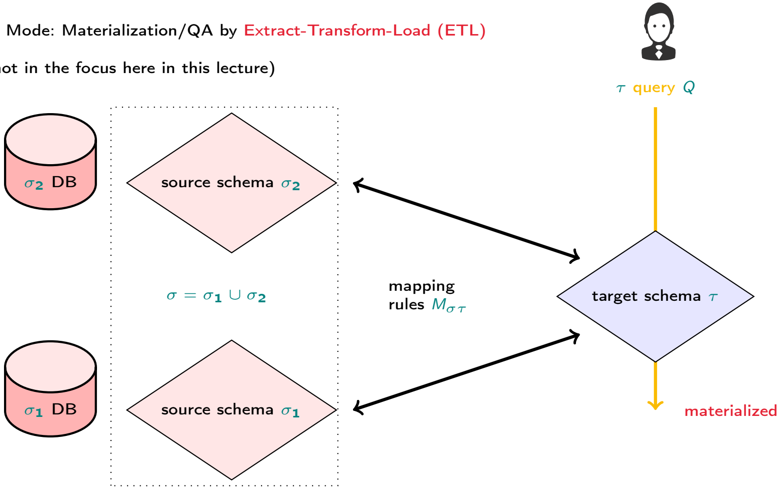
Data Integration (DI): Main Setting



Data Integration (DI): Main Setting

1. Mode: Materialization/QA by Extract-Transform-Load (ETL)

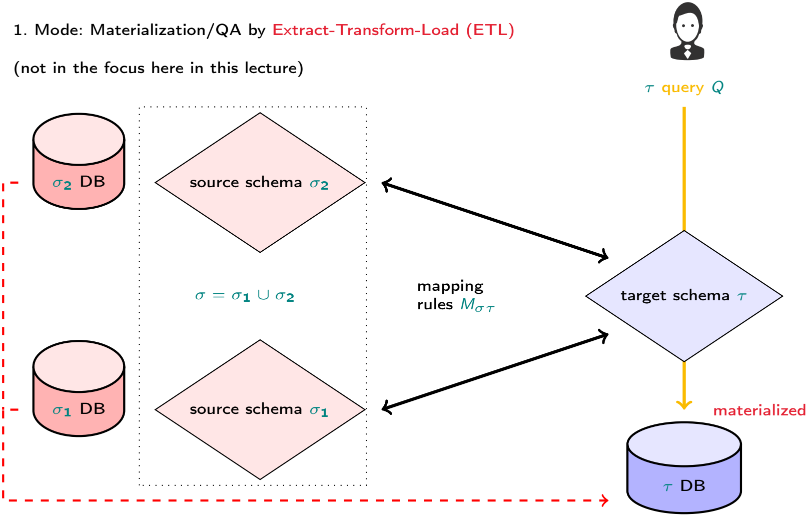
(not in the focus here in this lecture)



Data Integration (DI): Main Setting

1. Mode: Materialization/QA by Extract-Transform-Load (ETL)

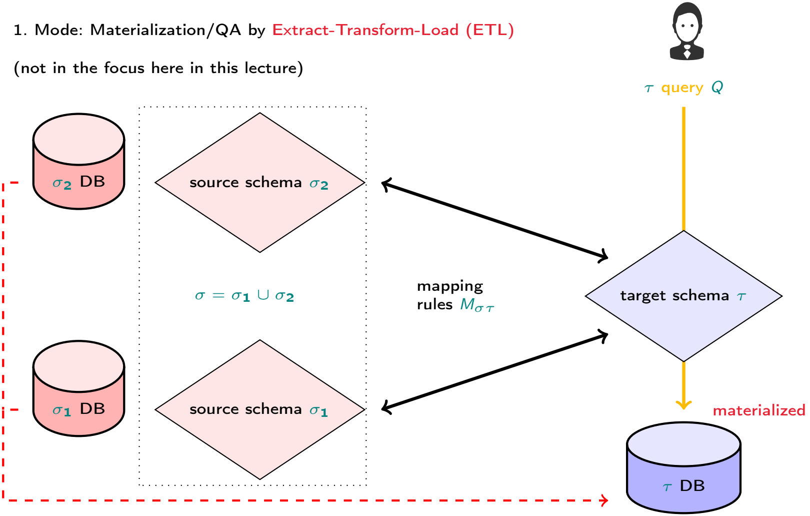
(not in the focus here in this lecture)



Data Integration (DI): Main Setting

1. Mode: Materialization/QA by Extract-Transform-Load (ETL)

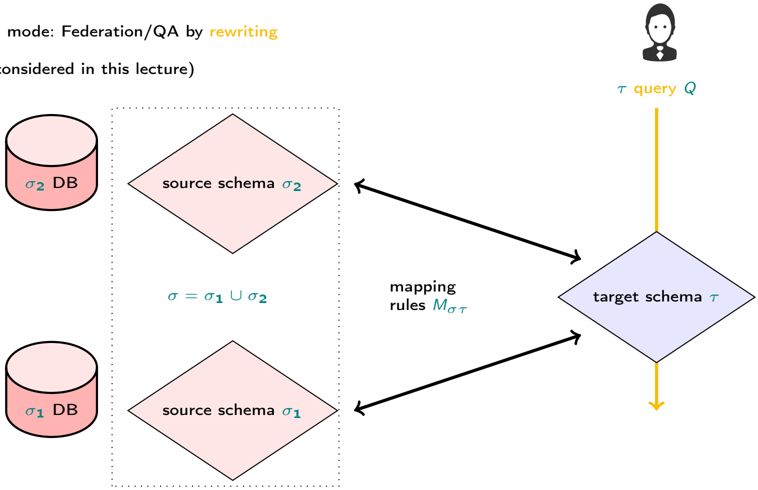
(not in the focus here in this lecture)



Data Integration (DI): Main Setting

2. mode: Federation/QA by rewriting

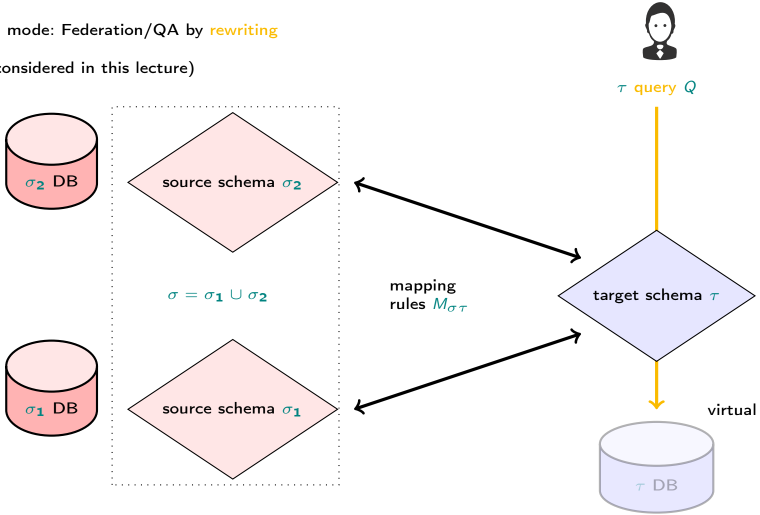
(considered in this lecture)



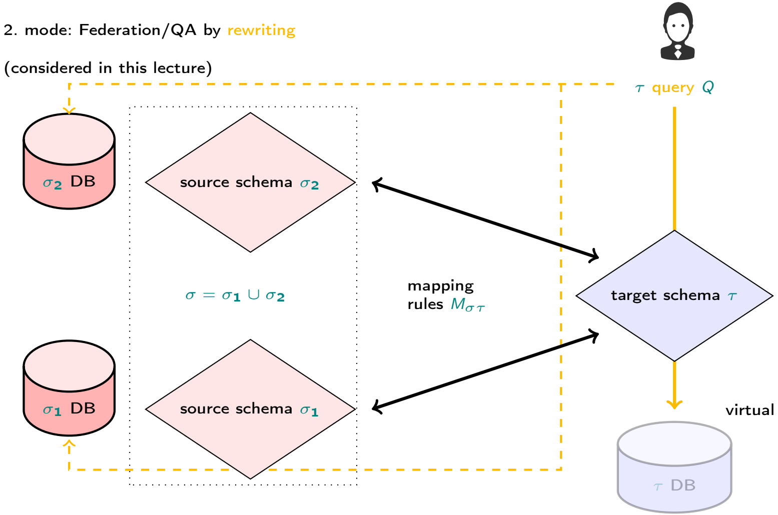
Data Integration (DI): Main Setting

2. mode: Federation/QA by rewriting

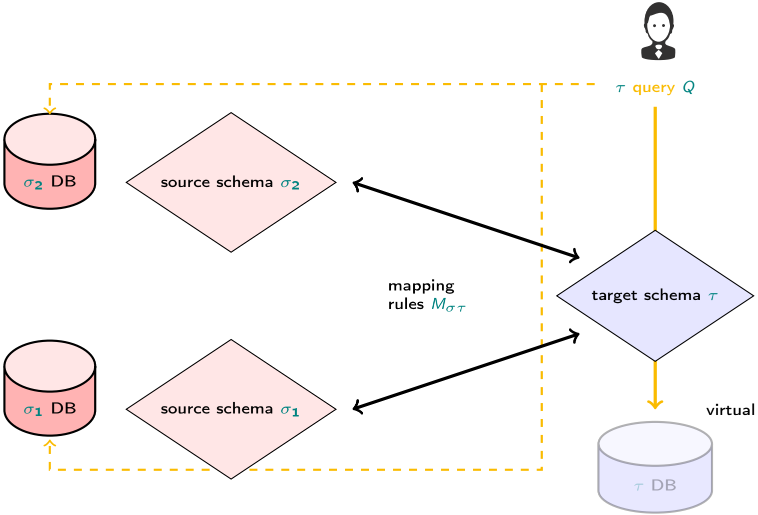
(considered in this lecture)



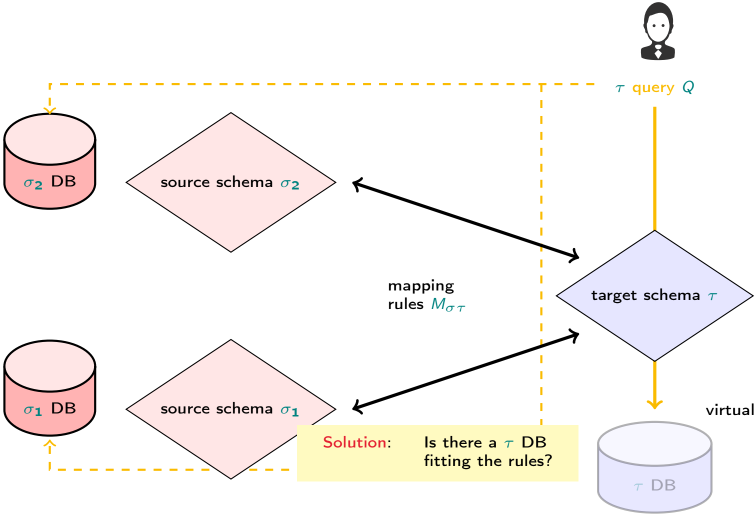
Data Integration (DI): Main Setting



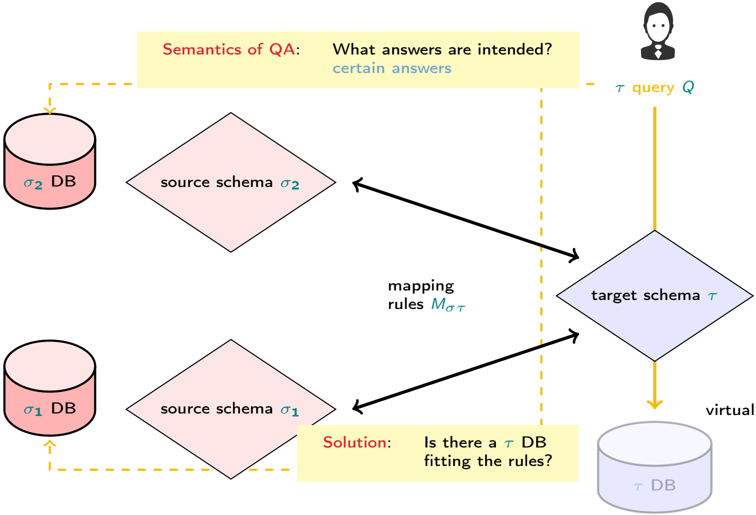
Data Integration (DI): Challenges



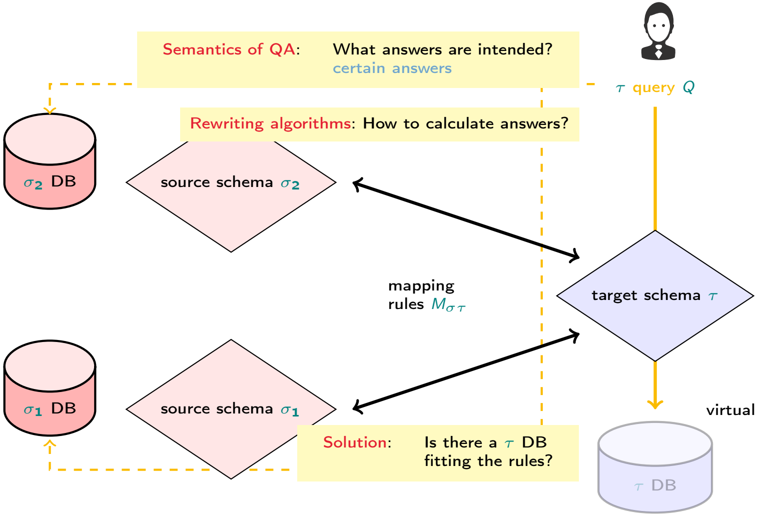
Data Integration (DI): Challenges



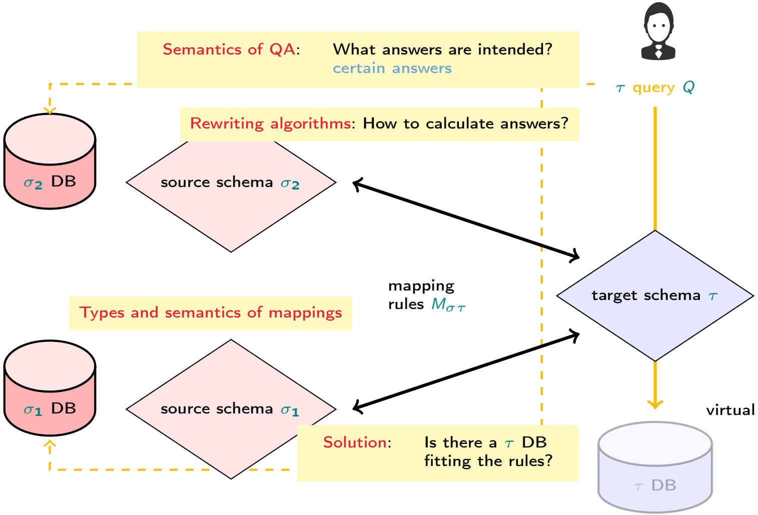
Data Integration (DI): Challenges



Data Integration (DI): Challenges



Data Integration (DI): Challenges



Short notice on method 1

- Usually built bottom-up (from sources) to global schema
- Used in data warehousing
- Still the most used approach in industry
- But usually: transformation ad hoc (not even w.r.t. declarative mappings)
- No well-founded theory in industry
- In contrast: Data exchange (see next two lectures)

Formalization and Basic Notions

Formalization

Definition (Lenzerini 2002)

A vector $(\tau, \sigma, M_{\sigma\tau}, M_{\tau})$ consisting of

- a global (alias target) schema τ
- a source (alias local) schema σ
- $M_{\sigma\tau} = \{$ source-to-target rules $\}$
- $M_{\tau} = \{$ target constraints $\}$

is called a data integration system $\mathcal{DI}$

- Lenzerini calls $M_{\sigma\tau}$ a mapping.
- Source schema σ is the union of the local schemas

  Logically: consider a single σ-DB consisting of disjoint unions of local σ_i DBs
- Some federation aspects dealt under theme complex: view rewriting

Convention

For ease of exposition, we will neglect target dependencies M_τ, i.e. let $M_\tau = \{\}$ in this lecture. Will deal with them in next lecture. (Makes definitions easier and lets us focus on rewriting aspects)

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

- Source-Target-Dependencies may be arbitrary FOL formula
- Usually they have a simple 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

- ϕ_σ is a conjunction of atoms over source schema σ
- ψ_τ is a conjunction of atoms over target schema τ

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

- Source-Target-Dependencies may be arbitrary FOL formula
- Usually they have a simple 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

- ϕ_σ is a conjunction of atoms over source schema σ
- ψ_τ is a conjunction of atoms over target schema τ

- So in particular, antecedens and succedens conjunctive queries (CQ)
- CQs “well-behaved”

Reminder: Conjunctive Queries (CQs)

- Class of sufficiently expressive and feasible FOL queries of form

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

  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

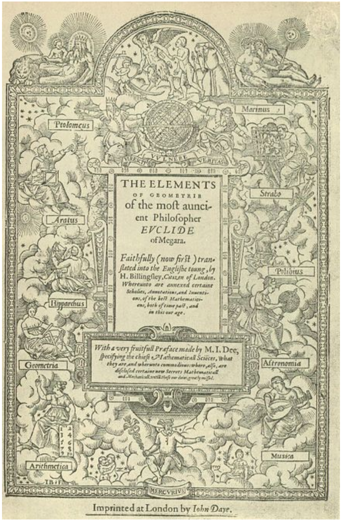
Are st-tgds Datalog rules?

- 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

Semantics for Data Integration Systems: Solutions

Definition

Given: A data integration system $\mathcal{DI}$ with mapping rules $M_{\sigma\tau}$ and a σ-instance $\mathfrak{S}$

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

Semantics for Data Integration Systems: Solutions

Definition

Given: A data integration system $\mathcal{DI}$ with mapping rules $M_{\sigma\tau}$ and a σ-instance $\mathfrak{S}$

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

- $(\mathfrak{S}, \mathfrak{T}) \models M_{\sigma\tau}$ 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
- $Sol_{\mathcal{DI}}(\mathfrak{S})$: Set of solutions for $\mathfrak{S}$ under $\mathcal{DI}$

Certain answering

- There may be more than one solution.
- What then is the semantics for query answering?

Definition (Certain answers (informally))

$\mathit{cert}_{\mathcal{DI}}(Q, \mathfrak{S}) =$ intersection of answers over all possible solutions

Certain answering

- There may be more than one solution.
- What then is the semantics for query answering?

Definition (Certain answers (formally))

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

Certain answering

- There may be more than one solution.
- What then is the semantics for query answering?

Definition (Certain answers (formally))

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

- General approach for dealing with incomplete information
  - Certain answers on incomplete DBs (see DE lecture)
  - Certain answers in inconsistent DBs (see Database Repairs lecture)
  - Certain answers in OBDA
  - Partial (full meet) revision (See belief revision lecture)

Types of Integration: LAV and GAV

Various Approaches for Virtual Data Integration

Form of rules in $M_{\sigma\tau}$ leads to different approaches

- Source-centric/local-as-view (LAV): sources "defined" in terms of global schema
- Global-schema-centric/global-as-view (GAV): global schema defined in terms of sources
- Mixed approach: GLAV
- Peer-to-peer/P2P: mapping without global schema

Various Approaches for Virtual Data Integration

Form of rules in $M_{\sigma\tau}$ leads to different approaches

- Source-centric/local-as-view (LAV): sources "defined" in terms of global schema
- Global-schema-centric/global-as-view (GAV): global schema defined in terms of sources
- Mixed approach: GLAV
- Peer-to-peer/P2P: mapping without global schema

Focus of this lecture

Example (Movie Scenario)

- τ: $movie(Title, Year, Director)$
  $european(Director)$
  $review(Title, Critique)$

- σ_1: $r_1(Title, Year, Director)$ european directors since 1960
- σ_1: $r_2(Title, Critique)$ critiques since 1990

- Q: Title and critique of movies since 1998

$$\exists D.movie(T, 1998, D) \wedge review(T, R)$$

GAV Approach

Definition

A GAV-$\mathcal{DI}$ has rules in $M_{\sigma\tau}$ for all relations $R_\tau \in \tau$ of the form (called GAV rules):

- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z}\, R_\tau(\vec{x}, \vec{z}))$ (sound)
- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z}\, R_\tau(\vec{x}, \vec{z}))$ (exact)

GAV Approach

Definition

A GAV-$\mathcal{DI}$ has rules in $M_{\sigma\tau}$ for all relations $R_\tau \in \tau$ of the form (called GAV rules):

- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z}\, R_\tau(\vec{x}, \vec{z}))$ (sound)
- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z}\, R_\tau(\vec{x}, \vec{z}))$ (exact)

- Given a source database, $M_{\sigma\tau}$ provides direct information about which data satisfy the elements of the global schema.

GAV Approach

Definition

A GAV-$\mathcal{DI}$ has rules in $M_{\sigma\tau}$ for all relations $R_\tau \in \tau$ of the form (called GAV rules):

- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, R_\tau(\vec{x},\vec{z}))$ (sound)
- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x},\vec{y}) \longleftrightarrow \exists \vec{z}\, R_\tau(\vec{x},\vec{z}))$ (exact)

- Given a source database, $M_{\sigma\tau}$ provides direct information about which data satisfy the elements of the global schema.
- Relations in τ are views, and queries are expressed over the views.

GAV Approach

Definition

A GAV-$\mathcal{DI}$ has rules in $M_{\sigma\tau}$ for all relations $R_\tau \in \tau$ of the form (called GAV rules):

- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, R_\tau(\vec{x},\vec{z}))$ (sound)
- $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x},\vec{y}) \longleftrightarrow \exists \vec{z}\, R_\tau(\vec{x},\vec{z}))$ (exact)

- Given a source database, $M_{\sigma\tau}$ provides direct information about which data satisfy the elements of the global schema.
- Relations in τ are views, and queries are expressed over the views.
- Simple evaluation by unfolding and running query over the data satisfying the global relations (as if single DB)

Example (GAV example)

- τ: $movie(Title, Year, Director)$
  $european(Director)$
  $review(Title, Critique)$
- σ_1: $r_1(Title, Year, Director)$ european directors since 1960
- σ_2: $r_2(Title, Critique)$ critiques since 1990
- Q: $\exists D.movie(T, 1998, D) \wedge review(T, R)$
- GAV rules

$$
\begin{aligned}
r_1(T, Y, D) &\rightarrow movie(T, Y, D) \\
r_1(T, Y, D) &\rightarrow european(D) \\
r_2(T, R) &\rightarrow review(T, R)
\end{aligned}
$$

- Lhss are views over the source
- Note: In second rule only attribute D projected
  $\forall X, \forall Y A(X, Y) \rightarrow B(X) \equiv \forall X(\exists Y A(X, Y) \rightarrow B(X)$.

Example (GAV example (continued))

- τ: $movie(Title, Year, Director)$
  $european(Director)$
  $review(Title, Critique)$
- σ_1: $r_1(Title, Year, Director)$ european directors since 1960
- σ_2: $r_2(Title, Critique)$ critiques since 1990
- Q: $\exists D.movie(T, 1998, D) \wedge review(T, R)$
- GAV rules

$$\begin{aligned} r_1(T, Y, D) &\rightarrow movie(T, Y, D) \\ r_1(T, Y, D) &\rightarrow european(D) \\ r_2(T, R) &\rightarrow review(T, R) \end{aligned}$$

- Q unfolded into

$$\exists D.r_1(T, 1998, D) \wedge r_2(T, R)$$

LAV Approach

Definition

A LAV-$\mathcal{DI}$ has rules in $M_{\sigma\tau}$ for all relations $R_\sigma \in \sigma$ of the form

- $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z}\, \psi_\tau(\vec{x}, \vec{z}))$ (sound)
- $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z}\, \psi_\tau(\vec{x}, \vec{z}))$ (exact)

LAV Approach

Definition

A LAV-$\mathcal{DI}$ has rules in $M_{\sigma\tau}$ for all relations $R_\sigma \in \sigma$ of the form

- $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z}\, \psi_\tau(\vec{x}, \vec{z}))$ (sound)
- $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z}\, \psi_\tau(\vec{x}, \vec{z}))$ (exact)

- Mapping $M_{\sigma\tau}$ and the source database $\mathfrak{S}$ do not provide direct information about which data satisfy the global schema.

LAV Approach

Definition

A LAV-$\mathcal{DI}$ has rules in $M_{\sigma\tau}$ for all relations $R_\sigma \in \sigma$ of the form

- $\forall \vec{x}\vec{y}(R_\sigma(\vec{x},\vec{y}) \longrightarrow \exists \vec{z}\, \psi_\tau(\vec{x},\vec{z}))$ (sound)
- $\forall \vec{x}\vec{y}(R_\sigma(\vec{x},\vec{y}) \longleftrightarrow \exists \vec{z}\, \psi_\tau(\vec{x},\vec{z}))$ (exact)

- Mapping $M_{\sigma\tau}$ and the source database $\mathfrak{S}$ do not provide direct information about which data satisfy the global schema.
- Sources are views, and we have to answer queries on the basis of the available data in the views. $\Longrightarrow$ view rewriting

Example (LAV example)

- τ: $movie(Title, Year, Director)$
  $european(Director)$
  $review(Title, Critique)$
- σ_1: $r_1(Title, Year, Director)$ european directors since 1960
- σ_2: $r_2(Title, Critique)$ critiques since 1990
- Q: $\exists D.movie(T, 1998, D) \wedge review(T, R)$
- LAV rules

$$r_1(T, Y, D) \rightarrow movie(T, Y, D) \wedge european(D) \wedge Y \geq 1960$$
$$r_2(T, R) \rightarrow movie(T, Y, D) \wedge review(T, R) \wedge Y \geq 1990$$

- Q has to be rewritten in order to re-express target atoms with source atoms

$$Q_{rew}(T, R) : \exists D.r_2(T, R) \wedge r_1(T, 1998, D)$$

GAV-LAV Comparison

- GAV
  - Quality depends on how well sources represented in target schema by mapping
  - Have to reconsider global schema when sources changed/added
  - Unfolding-based query processing

GAV-LAV Comparison

- GAV
  - Quality depends on how well sources represented in target schema by mapping
  - Have to reconsider global schema when sources changed/added
  - Unfolding-based query processing
- LAV
  - Quality depends on how well sources represented
  - High modularity and extensibility (reconsider only the definition the source relation which may have changed)
  - Query processing needs reasoning (rewriting)

The Full Picture of Different Integration Scenarios

- Parameters
  - MT: Mapping Type (GAV or LAV)
  - E: Exactness (sound or complete)
  - TC: Target-Constraints? (yes or no)
  - τ Query language (not mentioned)
- Outcomes
  - INCM: Incomplete answers (yes, no)
  - INCN: Inconsistency of target DBs (yes, no)

TC	MT	E	INCM	INCN
no	GAV	exact	no	no
no	GAV	sound	yes[a]/no	no
no	LAV	sound	yes	no
no	LAV	exact	yes	yes
yes	GAV	exact	no	yes
yes	GAV	sound	yes	yes
yes	LAV	sound	yes	yes
yes	LAV	exact	yes	yes

[a] For τ queries without negation

View Rewriting

View Rewriting

- We saw that QA under LAV requires rewriting
- But even under GAV may be needed if sources only accessible via views
- General field of rewriting w.r.t. views
- Relevant problem in
  - Data warehousing
  - Query optimization
  - Physical independence
  - Accessibility restriction/privacy aspects
- Problem: Generally (for arbitrary FOL queries) not decidable whether (exact/sound) rewriting possible

  $\Longrightarrow$ Consider CQs

Why undecidable?

- Let Q be arbitrary FOL query
- Consider exact LAV
- τ is primed copy of σ
- $M_{\sigma\tau}$ identify relations in σ with primed copies in τ
- No views at all given.
- Answering in this setting means answering independently of data in source.
- Can only happen if $D_1^Q = D_2^Q$ for all τ-DBs D_1, D_2
- But one knows that deciding $D_1^Q = D_2^Q$ for arbitrary FOL is undecidable

Definition (Query rewriting w.r.t views (intuition))

- Given: Query Q,
  View definitions $V_1, \ldots, V_n$
- a rewriting Q_{rew} is a query that refers to the views only (and possibly interpreted relations)
  An equivalent rewriting additionally fulfils $Q \equiv Q_{rew}$.

Definition (Query rewriting w.r.t views (formal))

- Schema σ of relations with arities n_i
- Queries $V_1, \ldots, V_k$ over schema τ with arities n_i
  Assume those views are CQs
- Input query Q over τ

Definition (Query rewriting w.r.t views (formal))

- Schema σ of relations with arities n_i
- Queries $V_1, \ldots, V_k$ over schema τ with arities n_i
  Assume those views are CQs
- Input query Q over τ

- Query Q_{rew} over σ is an equivalent rewriting of Q iff for all τ-DBs:

$$D^Q = Q_{rew}(V_1(D) \ldots V_n(D))$$

Definition (Query rewriting w.r.t views (formal))

- Schema σ of relations with arities n_i
- Queries $V_1, \ldots, V_k$ over schema τ with arities n_i
  Assume those views are CQs
- Input query Q over τ

- Query Q_{rew} over σ is an equivalent rewriting of Q iff for all τ-DBs:

$$D^Q = Q_{rew}(V_1(D) \ldots V_n(D))$$

- Query Q_{rew} over σ is a maximally contained rewriting of Q iff for all τ-DBs
  - $D^Q \supseteq Q_{rew}(V_1(D) \ldots V_n(D))$ and
  - Q_{rew} is maximal in this property, i.e.
    if $D^Q \supseteq Q'_{rew}(V_1(D) \ldots V_n(D))$ then $Q'_{rew} \subseteq Q_{rew}$.

Example (Equivalent rewriting 1)

- DB: $\{Movie(ID, title, year, genre), Director(ID, director), Actor(ID, actor)\}$
- $Q(T, Y, D) : Movie(I, T, Y, G) \wedge Y \geq 1950 \wedge G = comedy \wedge Director(I, D) \wedge Actor(I, D)$
- $V_1(T, Y, D) : Movie(I, T, Y, G) \wedge Y \geq 1940 \wedge G = comedy \wedge Director(I, D) \wedge Actor(I, D)$
- Because $V_1 \supseteq Q$ we get equivalent rewriting

  $Q_{rew} : V_1(T, Y, D) \wedge Y \geq 1950$

Example (Equivalent Rewriting 2)

- DB: $\{Movie(ID, title, year, genre), Director(ID, director), Actor(ID, actor)\}$
- $Q(T, Y, D): Movie(I, T, Y, G) \wedge Y \geq 1950 \wedge G = comedy \wedge Director(I, D) \wedge Actor(I, D)$
- $V_2(I, T, Y): Movie(I, T, Y, G) \wedge Y \geq 1950 \wedge G = comedy$
- $V_3(I, D): Director(I, D) \wedge Actor(I, D)$
- No subsumption relation for views but nonetheless equivalent rewriting

  $Q_{rew}: V_2(I, T, Y) \wedge V_3(I, D)$

Example (Maximally Contained Rewriting)

- DB: $\{Movie(ID, title, year, genre), Director(ID, director), Actor(ID, actor)\}$
- $Q(T, Y, D)$: $Movie(I, T, Y, G) \wedge Y \geq 1950 \wedge G = comedy \wedge Director(I, D) \wedge Actor(I, D)$
- $V_4(I, T, Y)$: $Movie(I, T, Y, G) \wedge Y \geq 1960 \wedge G = comedy$
- $V_3(I, D)$: $Director(I, D) \wedge Actor(I, D)$
- Only maximally-contained rewriting possible

  $Q_{rew} : V_4(I, T, Y) \wedge V_3(I, D)$

Naive query rewriting algorithm

- Input
  - Conjunctive queries $V_1, \ldots, V_n$ over τ
  - Query Q over τ
- Output: equivalent or maximally contained Q_{rew}

- Procedure
  - Guess Q' over views
  - Unfold Q' in terms of views
  - Check if one unfolding contained in Q
- If one unfolding U equivalent with Q, then $Q_{rew} = U$
- Otherwise $Q_{rew} = \bigvee Unfoldings(Q')$

Naive query rewriting algorithm

- Input
  - Conjunctive queries $V_1, \ldots, V_n$ over τ
  - Query Q over τ
- Output: equivalent or maximally contained Q_{rew}

- Procedure
  - Guess Q' over views
  - Unfold Q' in terms of views
  - Check if one unfolding contained in Q
- If one unfolding U equivalent with Q, then $Q_{rew} = U$
- Otherwise $Q_{rew} = \bigvee Unfoldings(Q')$

- Need to test equivalence: Feasible for CQs

Naive query rewriting algorithm

- Input
  - Conjunctive queries $V_1, \ldots, V_n$ over τ
  - Query Q over τ
- Output: equivalent or maximally contained Q_{rew}

- Procedure
  - Guess Q' over views
  - Unfold Q' in terms of views
  - Check if one unfolding contained in Q
- If one unfolding U equivalent with Q, then $Q_{rew} = U$
- Otherwise $Q_{rew} = \bigvee Unfoldings(Q')$

- Need to test equivalence: Feasible for CQs
- Need to constrain search space: 1) theoretical bound, 2) prune search space (Bucket, MiniCon, inverse rules)

Theoretical Bound

Theorem

- *If there is an equivalent rewriting (maximally contained rewriting, resp.) of Q,*

  then there is one with at most n subgoals where n is number of atoms in Q (all CQs in disjunction have less than n atoms, resp.)
- *Finding rewriting is NP-complete*

Bucket Algorithm: Idea

1. Create a bucket for each subgoal g in query Q
   Bucket contains view atoms contributing to g

2. Create rewritings from the cartesian product of buckets.

An Example Run

Example

- $Q(ID, Dir)$: $Movie(ID, title, year, genre) \wedge Revenues(ID, amount) \wedge$
  $Director(ID, dir) \wedge amount \geq 100M$
- $V_1(I, Y)$: $Movie(I, T, Y, G) \wedge Revenues(I, A) \wedge I \geq 5000 \wedge A \geq 200M$
- $V_2(I, A)$: $Movie(I, T, Y, G) \wedge Revenues(I, A)$
- $V_3(I, A)$: $Revenues(I, A) \wedge A \leq 50M$
- $V_4(I, D, Y)$: $Movie(I, T, Y, G) \wedge Director(I, D) \wedge I \leq 3000$

An Example Run

Example

- $Q(ID, Dir)$: $Movie(ID, title, year, genre) \wedge Revenues(ID, amount) \wedge$
  $Director(ID, dir) \wedge amount \geq 100M$
- $V_1(I, Y)$: $Movie(I, T, Y, G) \wedge Revenues(I, A) \wedge I \geq 5000 \wedge A \geq 200M$
- $V_2(I, A)$: $Movie(I, T, Y, G) \wedge Revenues(I, A)$
- $V_3(I, A)$: $Revenues(I, A) \wedge A \leq 50M$
- $V_4(I, D, Y)$: $Movie(I, T, Y, G) \wedge Director(I, D) \wedge I \leq 3000$

- Atoms that can contribute to $Movie(ID, title, year, genre)$

$$V_1(ID, year), V_2(ID, A'), V_4(ID, D', year)$$

An Example Run

Example

- $Q(ID, Dir)$: $Movie(ID, title, year, genre) \wedge Revenues(ID, amount) \wedge$
  $Director(ID, dir) \wedge amount \geq 100M$
- $V_1(I, Y)$: $Movie(I, T, Y, G) \wedge Revenues(I, A) \wedge I \geq 5000 \wedge A \geq 200M$
- $V_2(I, A)$: $Movie(I, T, Y, G) \wedge Revenues(I, A)$
- $V_3(I, A)$: $Revenues(I, A) \wedge A \leq 50M$
- $V_4(I, D, Y)$: $Movie(I, T, Y, G) \wedge Director(I, D) \wedge I \leq 3000$

- Atoms that can contribute to $Movie(ID, title, year, genre)$

$$V_1(ID, year), V_2(ID, A'), V_4(ID, D', year)$$

- Similarly for $Revenues(ID, amount)$ and $Director(ID, dir)$

Example (First Candidate rewriting)

$Movie(ID, title, year, genre)$	$Revenues(ID, amount)$	$Director(ID, dir)$
$V_1(ID, year)$	$V_1(ID, Y')$	$V_4(ID, Dir, Y')$
$V_2(ID, A')$	$V_2(ID, amount)$	
$V_4(ID, D', year)$		

Consider first triple as potential rewriting

- first atom redundant,
- second and third exclusive (so can neglect rewriting Q_1')
- $Q_1'(ID, dir) = \cancel{V_1(ID, year)} \wedge V_1(ID, y') \wedge V_4(ID, dir, y')$

Example (Second candidate rewriting)

$Movie(ID, title, year, genre)$	$Revenues(ID, amount)$	$Director(ID, dir)$
$V_1(ID, year)$	$V_1(ID, Y')$	$V_4(ID, Dir, Y')$
$V_2(ID, A')$	$V_2(ID, amount)$	
$V_4(ID, D', year)$		

- $Q_2'(ID, dir) = V_2(ID, A') \ \wedge V_2(ID, amount) \wedge V_4(ID, dir, y')$

Example (Second candidate rewriting)

$Movie(ID, title, year, genre)$	$Revenues(ID, amount)$	$Director(ID, dir)$
$V_1(ID, year)$	$V_1(ID, Y')$	$V_4(ID, Dir, Y')$
$V_2(ID, A')$	$V_2(ID, amount)$	
$V_4(ID, D', year)$		

- $Q_2'(ID, dir) = \overbrace{V_2(ID, A')}^{\text{becomes redundant}} \wedge V_2(ID, amount) \wedge V_4(ID, dir, y')$
- $Q_2''(ID, dir) = V_2(ID, amount) \wedge V_4(ID, dir, y') \wedge amount \geq 100M;$

Step 1: Create Buckets Procedure

```
Input  : CQ Q(X⃗) = R_1(X⃗), ..., R_n(X⃗_n), c_1, ..., c_l; set 𝒱 of CQ views
Output: list of buckets
forall i ∈ [n] do
    Bucket_i := ∅
end
forall i ∈ [n] do
    foreach V ∈ 𝒱 do
        //Let V be of form V(Y⃗) = S_1(Y⃗_1), ..., S_m(Y⃗_m), d_1, ..., d_k
        forall j ∈ [m] do
            if R_i = S_j then
                // Let ψ be the mapping defined on the vars(V) as: Let y be the bth variable
                  in Y⃗_i and x be the bth variable in X⃗_i
                if x ∈ X⃗ and y ∉ Y⃗ then
                    ψ undefined, j++
                else if y ∈ Y⃗ then
                    ψ(y) = x
                else
                    ψ(y) is a new variable not in Q or V
                endif
                Q' := R_1(X⃗_1), ..., R_n(X⃗_n), c_1, ..., c_n,
                  S_1(ψ(Y⃗_1)), ..., S_m(ψ(Y⃗_m)), ψ(d_1), ... ψ(d_k)
                if Q' is satisfiable then
                    Add ψ(V) to Bucket_i
                endif
            endif
        end
    end
end
return Bucket_1, ..., Bucket_n
```

Step 2: Creating rewritings (exact or max. contained)

Consider each $Q' \in Bucket_1 \times \cdots \times Bucket_n$

- For equivalent rewriting
  - If $Q \equiv Q'$ or can add interpreted atoms C s.t. $Q \wedge C \equiv Q'$ then $Q_{rew} := Q$ is a potential rewriting
- For maximally contained rewriting construct UCQ Q_{rew} rewriting considering each conjunctive rewriting Q'
  - If $Q' \subseteq Q$, then $Q_{rew} := Q_{rew} \vee Q'$
  - If there is an interpreted atom C that can be added to Q' s.t. $Q' \wedge C \subseteq Q$ then $Q_{rew} := Q_{rew} \vee (Q' \wedge C)$
  - If $Q' \not\subseteq Q$ but there is homomorphism ψ on head variables of Q' s.t. $\psi(Q') \subseteq Q$, then $Q_{rew} := Q_{rew} \vee \psi(Q')$

Other Algorithms

- Bucket algorithms do not consider interaction of goals

$$\begin{aligned} Q(title, year, dir) &= Movie(ID, title, year, genre), \\ &\quad Director(ID, dir), Actor(ID, dir) \\ V_5(D, A) &= Director(I, D), Actor(I, A) \end{aligned}$$

- Variable I not in head of V_5 hence V_5 not usable in rewriting
  $\Longrightarrow$ mitigated in MiniCon

- Logical approach with inverse rules
- See **Lit:** A. Doan, A. Halevy, and Z. Ives. Principles of Data Integration. 2012. Chapter 2, p. 51 ff

Further Topics

Finding (logical) mappings I: Hierarchical approach

Topic on its own, here only some hints

- Find value correspondences by schema matching methods
- On top of them build logical mapping rules

Example

- $\tau : movie(Title, Year, Director), european(Director),$ $review(Title, Critique)$
- $\sigma : r_1(MTitle, Year, Director), r_2(MTitle, Critique)$
- Value correspondence: $r_1.MTitle \longrightarrow movie.Title$ etc. (found by, say, string-matching)
- Possible LAV-rule: $r_1(T, Year, Director) \longrightarrow movie(T, Year, Director)$

Lit: U. Leser and F. Naumann. Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. Chapter 5

Finding (Logical) Mappings II: Direct-Supervised

Learn directly logical mappings in supervised fashion

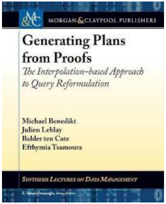
- Present pairs of sources DBs and potential target DB solutions
- See, e.g., **Lit:** G. Gottlob and P. Senellart. Schema mapping discovery from data instances. J. ACM, 57(2), Feb. 2010.

- Systemically studied using computational learning theory for GAV mappings in
  Lit: B. T. Cate, V. Dalmau, and P. G. Kolaitis. Learning schema mappings. ACM Trans. Database Syst., 38(4):28:1–28:31, Dec. 2013.

View Rewriting Advanced

- Consider view rewriting as application of Craig's interpolation theorem
- Benthem 2008: "last significant property of FOL that has come to light"

  Lit: J. van Benthem. The many faces of interpolation. Synthese, 164(3):451–460, 2008.
- Benedikt et al. generalize this w.r.t. accesibility methods (privacy)



Lit: M. Benedikt, J. Leblay, B. ten Cate, and E. Tsamoura. Generating Plans from Proofs: The interpolation-based Approach to Query Reformulation. Synthesis Lectures on Data Management, 2016.

Next Lecture: Data Exchange

- Main difference: requires materialization
- Usually considered without views (no access restrictions)
- Will consider in that lecture also target constraints
- Will consider mapping management