DATA FUSION –
a stage in data integration

For the Course on Information Extraction and Integration from Big Data

Lectures on Data Fusion: an Outline

1st Lecture

1. Data Integration – an Introduction
2. Data Fusion as a stage of Database Integration
3. Conflict resolution strategies and functions at the fusion stage
4. Data fusion and conflict resolution operators
5. Data Fusion: Join-based Approaches
6. Data Fusion: Union-based Approaches

2nd Lecture

1. Data Fusion Operator
2. Illustration of various data fusion operators use
3. Another approach for data fusion
4. Data fusion in existing integration systems
5. Truth-discovery techniques
6. Open Problems and Challenges
7. References

Sources (origins) of this part of the course

The approaches selected as the basis for this part of the course are given in accordance with the research made:

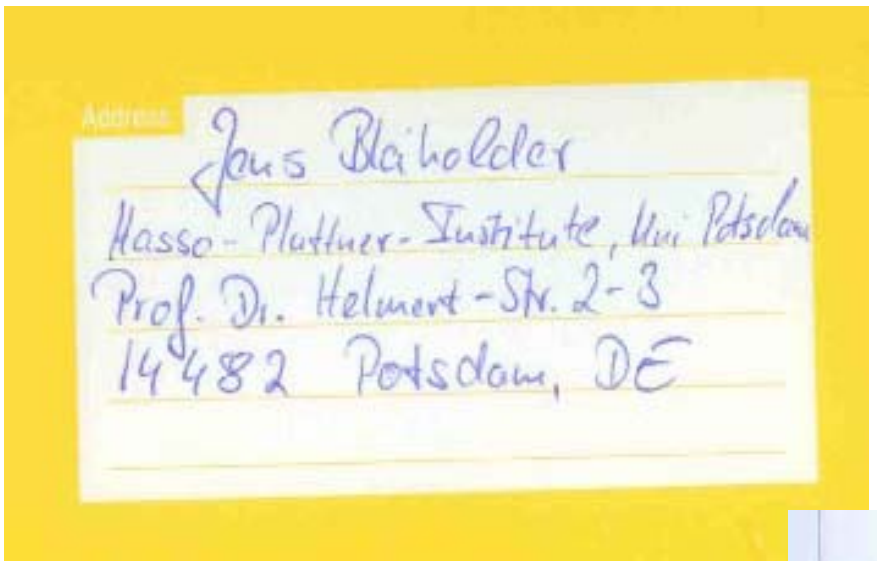
- in Potsdam University (e.g., in the dissertation of Jens Bleiholder «**Data Fusion and Conflict Resolution in Integrated Information Systems»),**
- **at the IBM Almaden Center (Midas project, HIL language),**
- **and other research centers**

Data Integration – an Introduction

Data fusion – what this part is about

- This part of the course follows **its entity resolution** part that included such issues as **data matching, data deduplication, data similarity, entity resolution, …).**
- **Data scientis**t – самая востребованная специальность (в США). В связи с необходимостью работы с большими данными, проблемы процедур эффективной работы с данными, их анализа, становятся все более острыми.
- **Курс посвящен интеграции разнообраных источников данных (multi – structured data**), прежде всего в Hadoop (включая Web, текстовые данные, изображения, социальные сети, log'и, результаты наблюдений (измерений), сенсоры, мобильные данные; потоковые данные в реальном масштабе времени, …)
- **В этой части курса мы будем выяснять, что такое data fusion, каково место этих функций в процессе интеграции баз данных, включая ETL and mediation**. В курсе мы будем ближе к ETL, реляционной модели данных, для приведения данных, полученных от разно-структурированных источников. Платформа – в принципе любая, но мы в этом курсе больше ориентированы на HADOOP в составе Big Insights IBM.
- **Напоминание: по понедельникам читается курс «Методы и платформы интеграции данных и организации хранилищ больших данных», тесно связанный с настоящим курсом**.

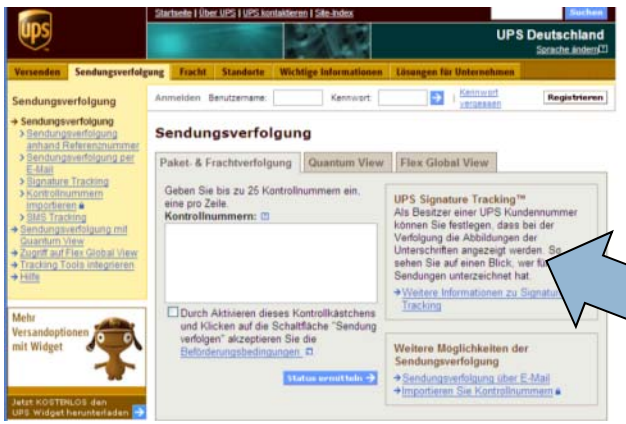
Origins of Data Conflicts



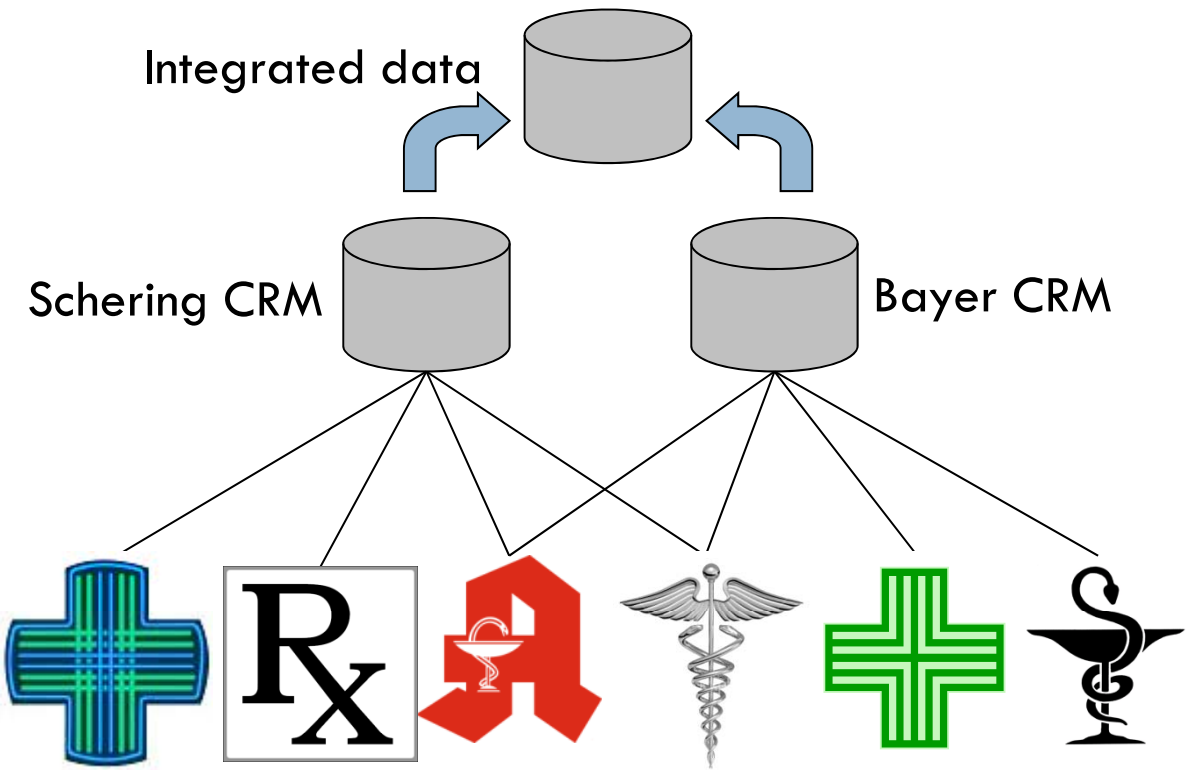
Original

ACM Computing Survey [BN08]

Origins of Data Conflicts



Origins of Data Conflicts



Origins of Data Conflicts: German Names

Origins of Data Conflicts: Difficult Names

488941 britney spears	29 britent spears	9 brinttany spears	5 brney spears	3 britiy spears
40134 brittany spears	29 brittnany spears	9 britanay spears	5 broitney spears	3 britmeny spear
36315 brittney spears	29 britttany spears	9 britinany spears	5 brotny spears	3 britneeey spea
24342 britany spears	29 btiney spears	9 britn spears	5 bruteny spears	3 britnehy spear
7331 britny spears	26 birttney spears	9 britnew spears	5 btiyney spears	3 britnely spear
6633 briteny spears	26 breitney spears	9 britneyn spears	5 btrittney spears	3 britnesy spear
2696 britteny spears	26 brinity spears	9 britrney spears	5 gritney spears	3 britnetty spea
1807 briney spears	26 britenay spears	9 brtiny spears	5 spritney spears	3 britnex spears
1635 brittny spears	26 britneyt spears	9 brtittney spears	4 bittny spears	3 britneyxxx spe
1479 brintey spears	26 brittan spears	9 brtny spears	4 bnritney spears	3 britnity spear
1479 britanny spears	26 brittne spears	9 brytny spears	4 brandy spears	3 britntey spear
1338 britiny spears	26 btittany spears	9 rbitney spears	4 brbritney spears	3 britnyey spear
1211 britnet spears	24 beitney spears	8 birtiny spears	4 breatiny spears	3 britterny spea
1096 britiney spears	24 birteny spears	8 bithney spears	4 breetney spears	3 brittneey spea
991 britaney spears	24 brightney spears	8 brattany spears	4 bretiney spears	3 brittnney spea
991 britnay spears	24 brintiny spears	8 breitny spears	4 brfitney spears	3 brittnyey spea
811 brithney spears	24 britanty spears	8 breteny spears	4 briattany spears	3 brityen spears
811 brtiney spears	24 britenny spears	8 brightny spears	4 brieteny spears	3 briytney spear
664 birtney spears	24 britini spears	8 brintay spears	4 briety spears	3 brltney spears
664 brintney spears	24 britnwy spears	8 brinttey spears	4 briitny spears	3 broteny spears
664 briteney spears	24 brittni spears	8 briotney spears	4 briittany spears	3 brtaney spears
601 bitney spears	24 brittnie spears	8 britanys spears	4 brinie spears	3 brtiiany spear
601 brinty spears	21 biritney spears	8 britley spears	4 brinteney spears	3 brtinay spears
544 brittaney spears	21 birtany spears	8 britneyb spears	4 brintne spears	3 brtinney spear
544 brittnay spears	21 biteny spears	8 britnrey spears	4 britaby spears	3 brtitany spear
364 britey spears	21 bratney spears	8 britnty spears	4 britaey spears	3 brtiteny spear
364 brittiny spears	21 britani spears	8 brittner spears	4 britainey spears	3 brtnet spears
329 brtney spears	21 britanie spears	8 brottany spears	4 britinie spears	3 brytiny spears
269 bretney spears	21 briteany spears	7 baritney spears	4 britinney spears	3 btney spears
269 britneys spears	21 brittay spears	7 birntey spears	4 britmney spears	3 drittney spear
244 britne spears	21 brittinay spears	7 biteney spears	4 britnear spears	3 pretney spears
244 brytney spears	21 brtany spears	7 bitiny spears	4 britnel spears	3 rbritney spear
220 breatney spears	21 brtiany spears	7 breateny spears	4 britneuy spears	2 barittany spea
220 britiany spears	19 birney spears	7 brianty spears	4 britnewy spears	2 bbbritney spea
199 britnney spears	19 brirtney spears	7 brintye spears	4 britnmey spears	2 bbitney spears
163 britnry spears	19 britnaey spears	7 britianny spears	4 brittaby spears	2 bbritny spears

Origins of Intra-Source Conflicts

- No integrity or consistency checks
- Redundant schemata
- Typos, transmission errors, incorrect calculations
- Variants
  - Kantstr. / Kantstrasse / Kant Str. / Kant Strasse
  - Kolmogorov / Kolmogoroff / Kolmogorow
- Typical confusion (OCR)
  - U<->V, 0<->o, 1<->l, etc.
- Obsolete values
  - Different update frequencies, forgotten update

Origins of Inter-Source Conflicts

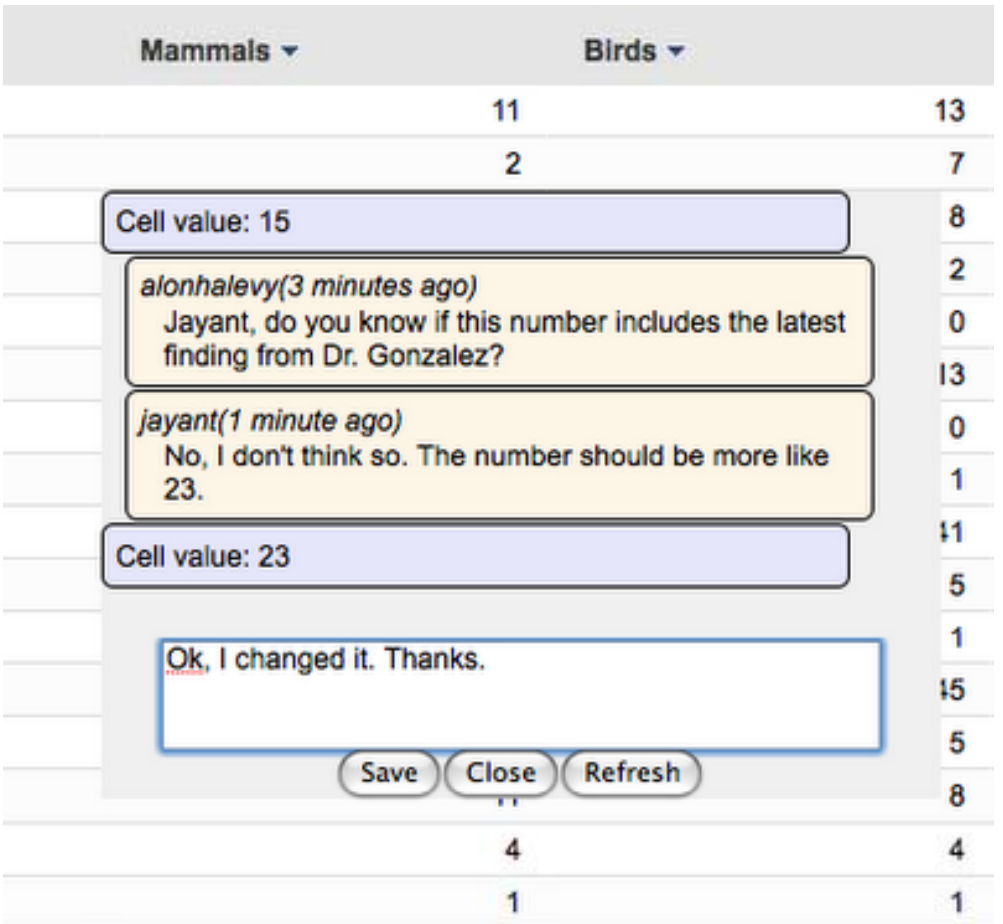
- Locally consistent but globally inconsistent
- Different data types
- Local spelling variations and conventions
  - Addresses
    - St → Street, Ave → Avenue, etc.
    - R.-Breitscheid-Str. 72 a → Rudolf-Breitscheid.-Str. 72A
    - 128 spellings for Frankfurt am Main
      - Frankfurt a.M., Frankfurt/M, Frankfurt, Frankfurt a. Main, …
  - Names
    - Dr. Ing. h.c. F. Porsche AG
    - Hewlett-Packard Development Company, L.P.
  - Numerical data
    - 10.000 € = 10T EURO = 10k EUR = 10.000,00€ = 10,000.- €
  - Phone numbers, birth dates, etc.

Resolution of Data Conflicts?

Google fusion tables labs
tables.googlelabs.com

- "… focus is on fusing data management and collaboration: **merging** multiple data sources, discussion of the data, querying, visualization, and Web publishing."
- "The power of data is truly harnessed when you combine data from multiple sources. Fusion Tables enables you to fuse multiple sets of data when they are about the **same entities**. In database speak, we call this a **join** on a primary key but the data originates from multiple independent sources."

Web Integration—Google Fusion Tables



- Allows discussion of values between users

Solving Data Conflicts

- Conflict ignoring
  - Not aware of conflicts
  - No resolution
  - Produce inconsistent results
- Conflict avoiding
  - Aware of conflicts
  - Avoid conflicts rather than performing resolutions for each conflict
- Conflict resolving
  - Individual fusion decisions for each conflict
    - Decisions are instance-or metadata based

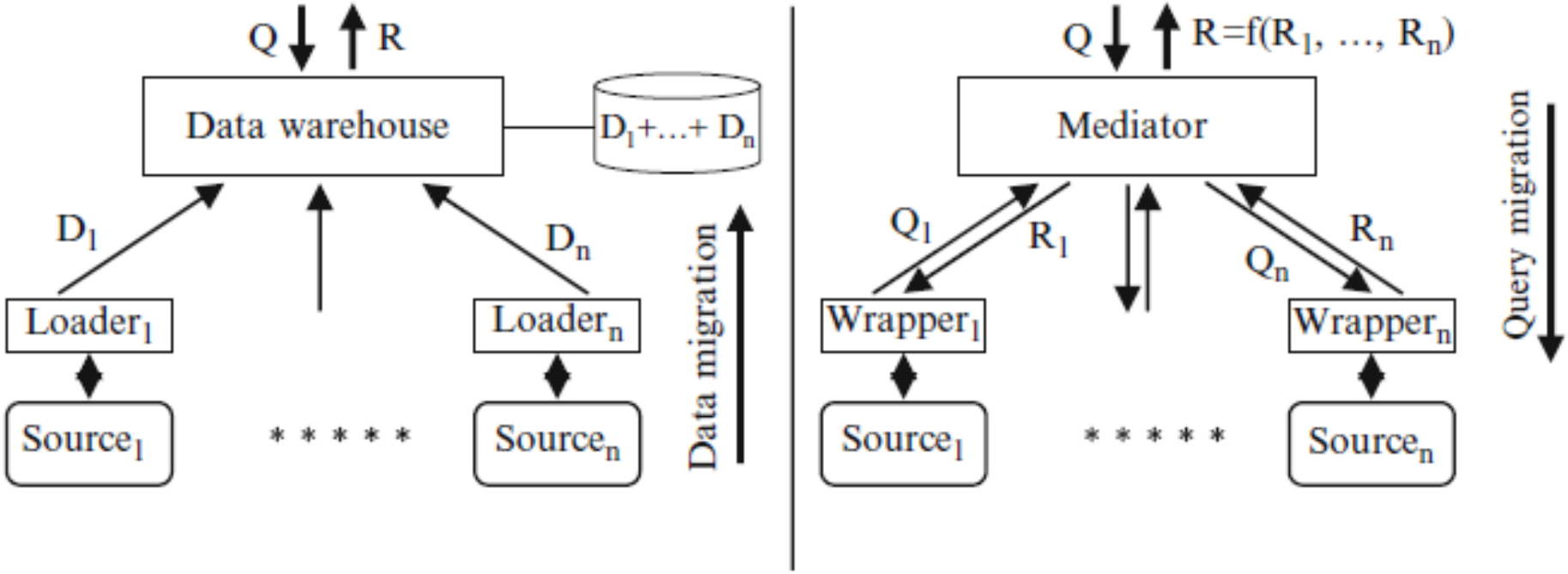
Data Conflict Elimination

- Error correction
  - Reference tables
    - Cities, countries, products ...
    - Similarity measures
  - Standardization and transformation
  - Domain-knowledge (meta data)
    - Conventions (country/region-specific spelling)
    - Ontologies
    - Thesauri, dictionaries for homonyms, synonyms, ...
  - Outlier detection and elimination
- And data fusion…

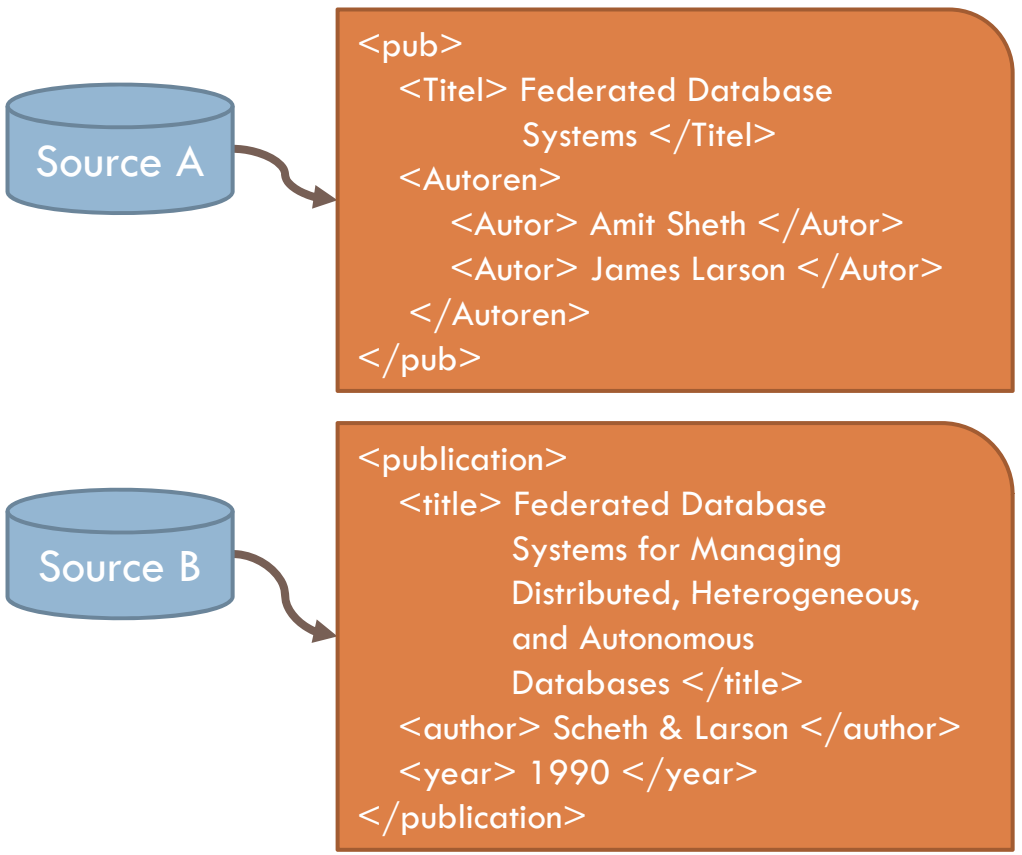
Stages of Data Integration

- **Four stages in data integration**: Data models unification; Schema mapping; Entity resolution including entity matching; **Data fusion**
- **Problems**: semantic preserving mappings of the heterogeneous data models; semantics of the sources: (e.g., conflicts – naming, representation, identity (representing the same real world entity), …); evolution of the autonomic source schemas; automatic schema mapping (ontology based); providing completeness, conciseness (uniqueness of entity representations), correctness of the integrated data, …
- **Schema mapping approaches**: GAV, LAV, GLAV (GAV is not scalable for large applications, LAV scales better)
- **Virtual vs Materialized** Integration
- In the course we follow **the ETL integration flow for the materialized data integration in the Hadoop infrastructure** (though usually the methods and algorithms considered are reusable also for the general case including **virtual (mediation based) integration**)

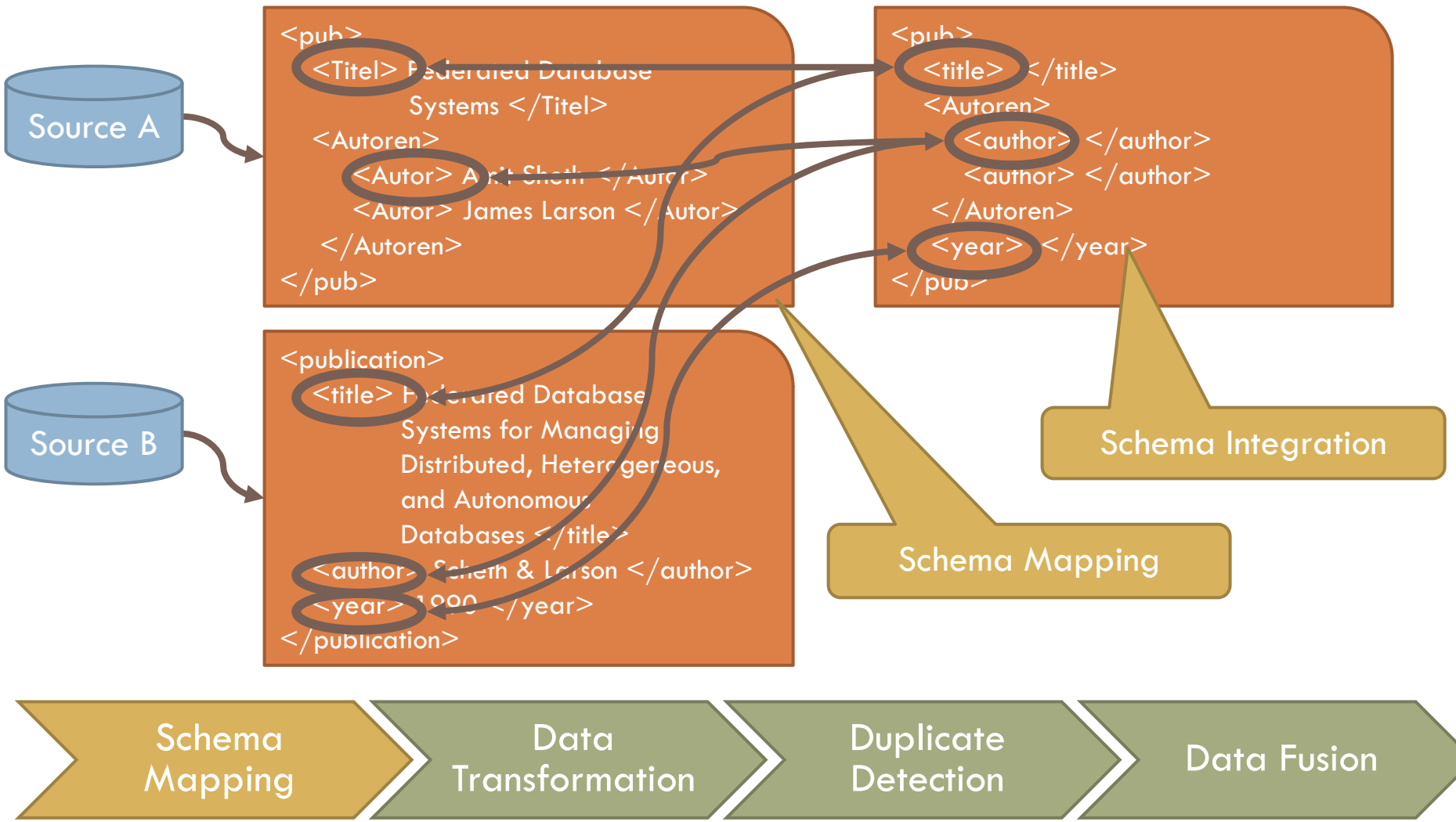
Materialized vs Virtual Data Integration



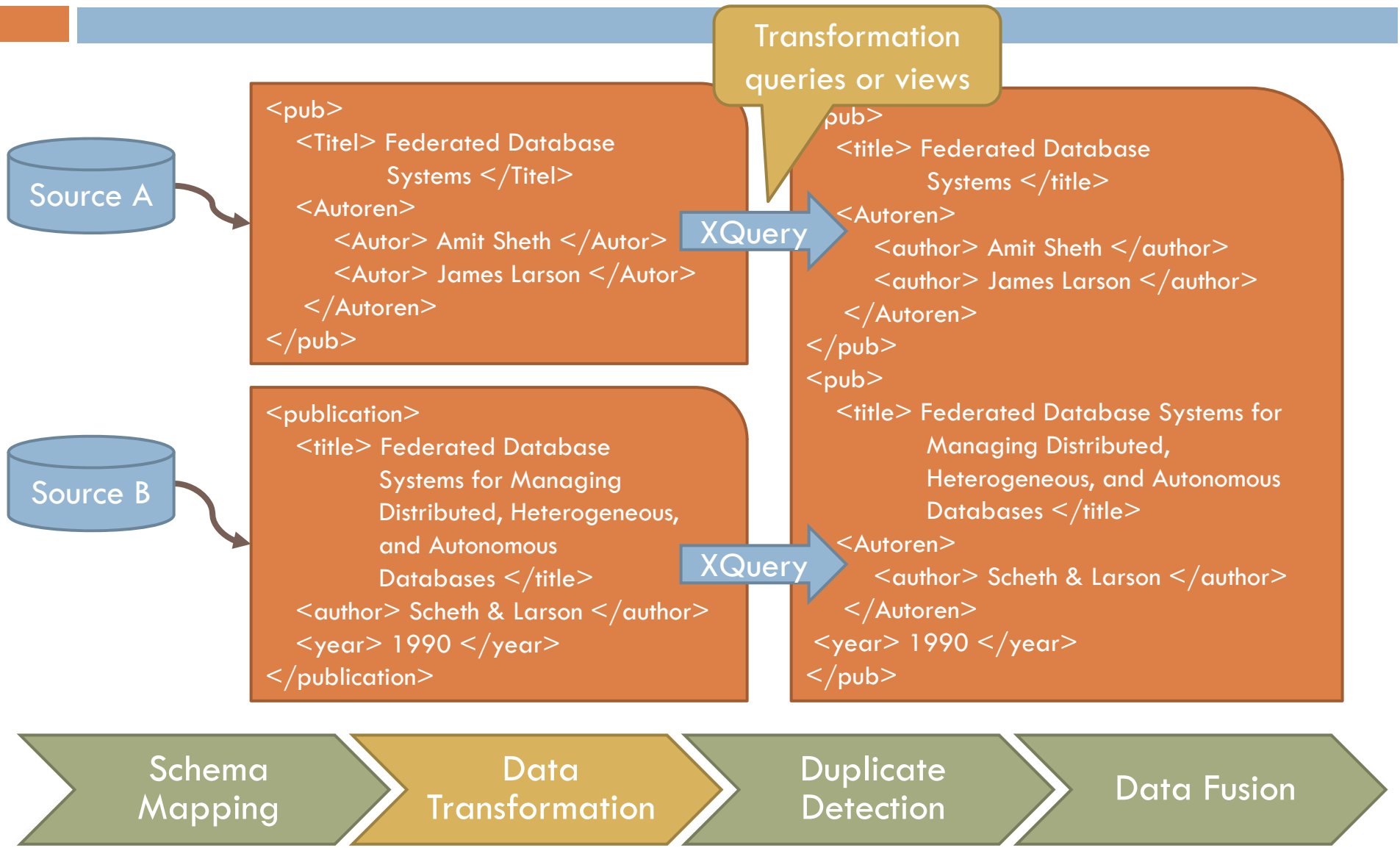
Information Integration



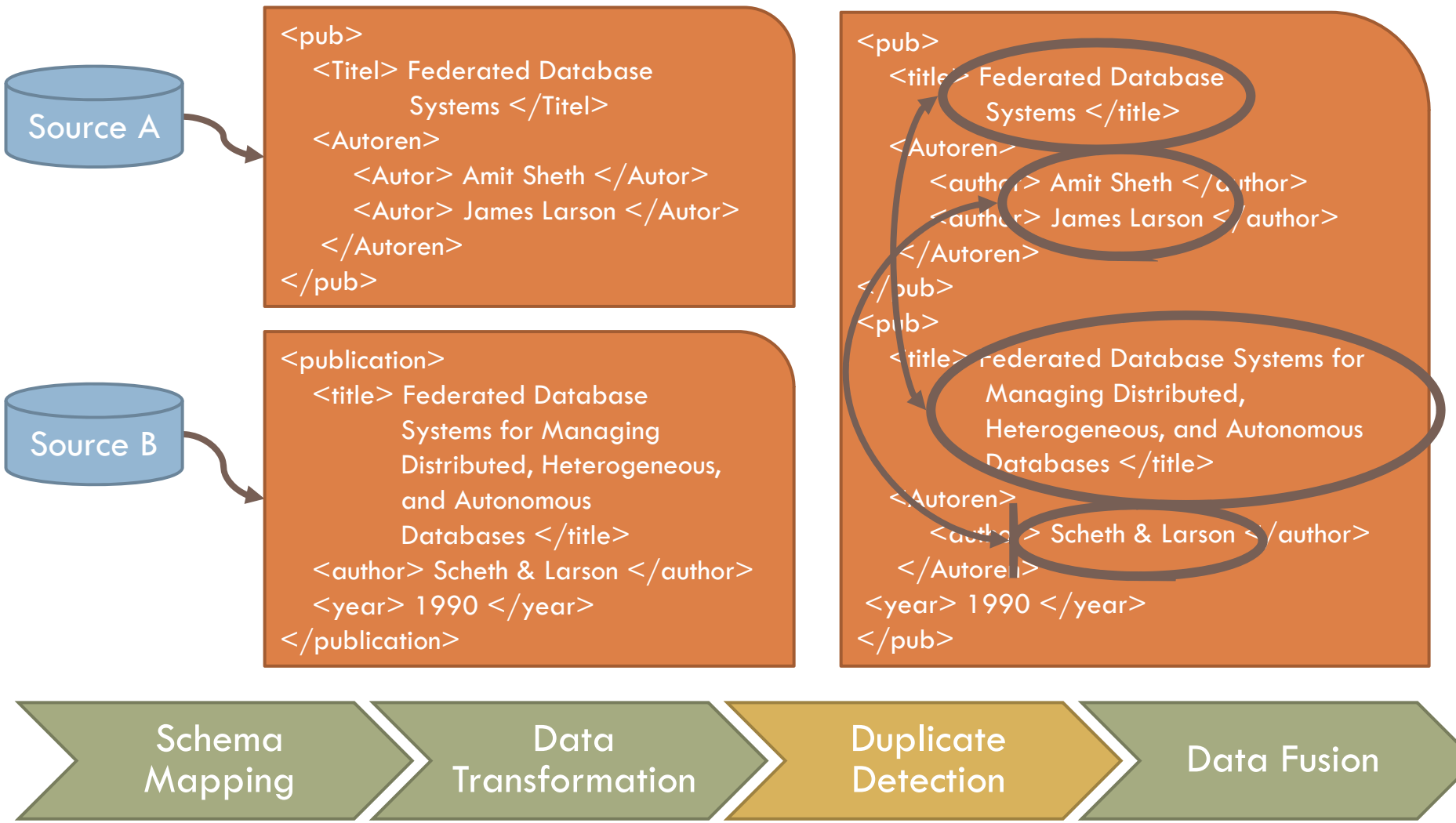
Information Integration



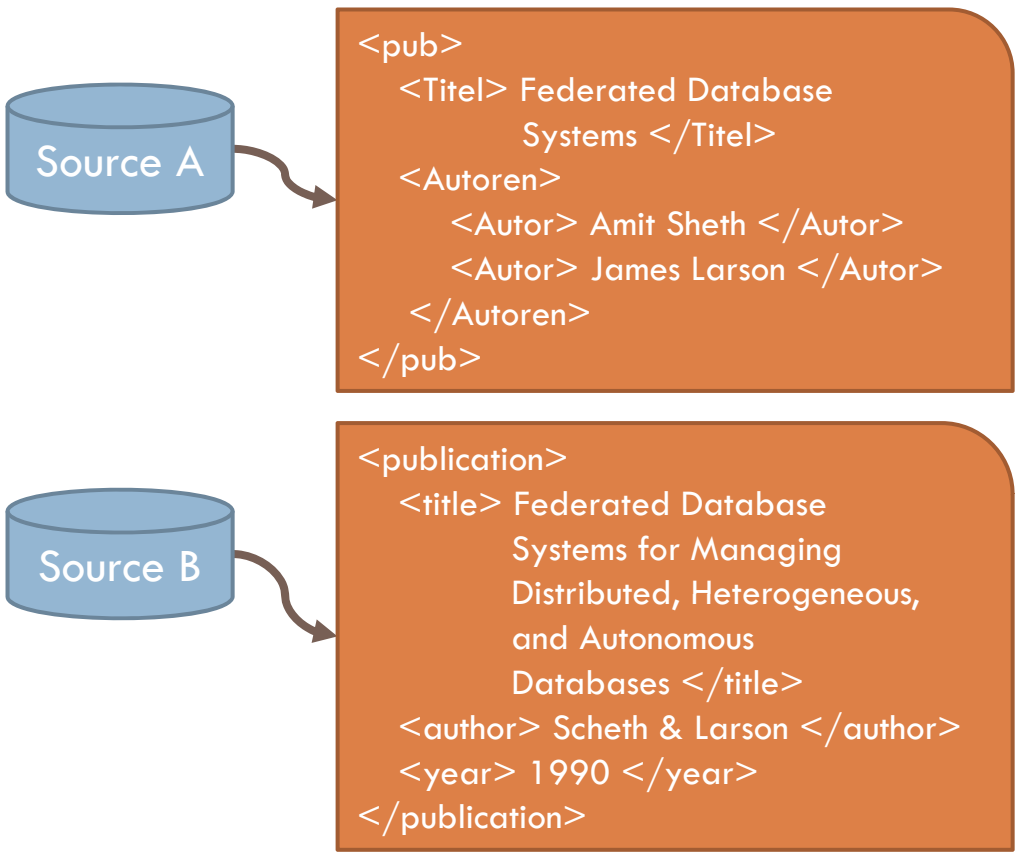
Information Integration



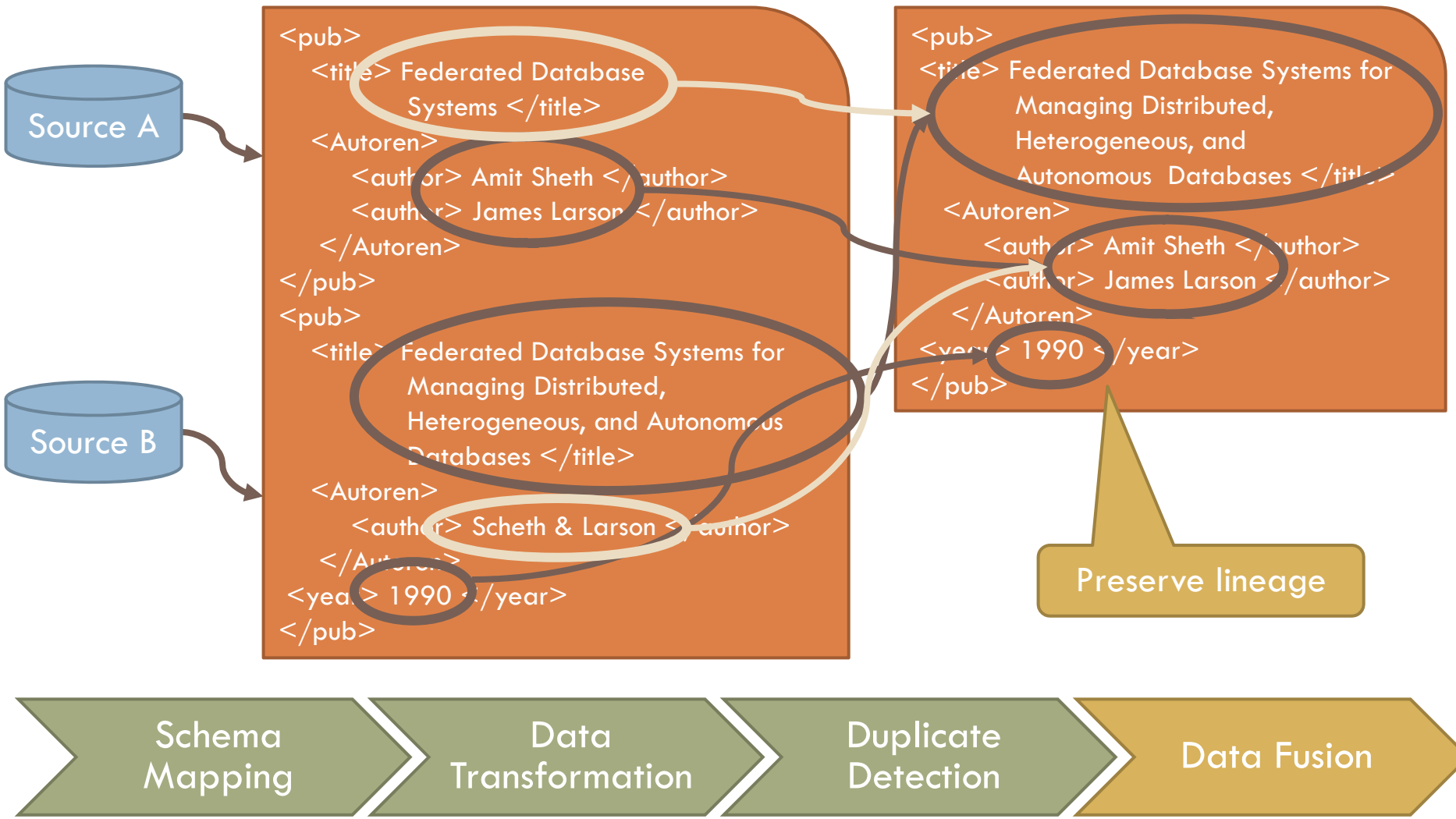
Information Integration



Information Integration



Information Integration



Data fusion in ETL and mediation techniques

- Main difference is that in ETL data fusion is a static process that ends with the materialization of the data fusion results
- Mediation is oriented on data integration on-the-fly (so called virtual integration) – sometimes provided in real time

- See more in the slides of lectures by Sergey Stupnikov:

- Методы и платформы интеграции данных и организации хранилищ больших данных (весенний семестр)

Entity resolution and integration - Heterogeneity

Different sources are heterogeneous, multi-structured

At instance level

Solutions: record linkage (also called “entity resolution”, “object matching”, …)

Different approaches: Rule-based / Probabilistic / …

2 steps: Blocking (find group of similar records) and Scoring (calculate how similar the records are)

Other solution: string similarity comparison

Using different measures for different types of data (Hemming distance, Jaccard similarity, …)

Data Fusion as a stage of Database Integration

Data Fusion

- Focus on **data fusion**
  - Addresses second challenge (conflicting data)
- Fusing records of same real-world entity into single record
- Resolving possible conflicts from different data sources
- Detecting and removing of dirty data

Data Fusion Stage (General View)

- Transform the extracted (and possibly linked) records into aggregated entities (**to join and map the source data into the target schema:** group, aggregate, and merge data into the proper, potentially nested output structures).
- Since data is collected from multiple sources, **duplicate va**lues for certain fields are inevitable and must be solved to determine which value survives.
- One of the serious problems is **how to satisfy the target dependencies**, constraints (e.g., key, tgd, egd) treating them as the ETL constraints imposed on the created collections (probably, as a part of entity resolution process).
- Another problem is **resolving identity conflicts**. Using duplicate detection techniques, multiple, possibly inconsistent representations of same real-world objects are found and an object identifier is assigned (proper link can be provided)
- An obvious next step **after detecting duplicates is to combine or merge them and thus produce a single, possibly more complete representation of that real-world object.** During this step, possible **data conflicts** among the multiple representations must somehow be resolved. **This second step is called data fusion**

Data Fusion Application Scenarios (examples)

- **Company Mergers:** An intuitive example where data fusion techniques are needed because of different representations of the same real-world entities coming from different sources are company mergers. There is a need to integrate the data (***materialized integration*** is possible) and, in case of conflicts, to handle them.
- **Catalog Integration:** Another example scenario where ***fuzzy duplicates*** exists and data fusion techniques are helpful is the integration of product catalogs. To guarantee the most recent prices, ***virtual, on-the-fly data integration is required*** (***federated search, mediation based search***). ***Duplicates among partial results obtained from the sources and their fusion into one representation should be done on-the-fly***.
- **Disaster Prediction :** In the disaster management scenario, different highly heterogeneous sources need to be connected to each other in order to allow for integration of all available data in the region together with the similar data on the former precedents. An integrated view on all available data then can help organize police, firefighters, and relief organizations.
- **Data Cleansing: ty**pical data cleansing scenarios where one single source or table is cleansed, e.g., by detecting and removing different representations of same real-world entities (fuzzy duplicates).

Goals of Data Integration

Increasing:

- **Completeness**
  - Measures amount of data
    - Number of tuples
    - Number of attributes
- **Conciseness**
  - Measures uniqueness of object representations
    - Number of unique objects
    - Number of unique attributes of the objects
- **Correctness**
  - Measures if the data is conform to the real world.

Data fusion: resolving conflicts and increasing **correctness** of data in process of data integration

Completeness and Conciseness

- **In analogy to *recall*, *completeness* of a dataset**, such as a query result or a source table, measures the amount of data in that set. The amount is measured relative to all data present in the universe of discourse.
- **Extensional completeness** is the number of unique object representations in a data set in relation to the overall number of unique objects in the real-world, e.g., in all the sources of an integrated system.
- ***Intensional completeness*** is the number of unique attributes in a data set in relation to the overall number of unique attributes available. In analogy to precision, conciseness measures the uniqueness of object representations in a dataset.
- **Four degrees of integration combining two tables** (1: no mapping of attributes and no entity keys are used; 2: mapping of attributes is used but no entity key used; 3: partial attribute mapping and entity keys are used; 4: full attribute mapping and entity keys are used). The 4^{th} variant provides the most integrated case.
- Most integration techniques and integrated information systems **try to increase extensional as well as intensional completeness**

Four Degrees of Integration

S.A	S.B	S.ID	T.ID	T.B	T.C
x	y	1	⊥	⊥	⊥
z	⊥	2	⊥	⊥	⊥
⊥	x	3	⊥	⊥	⊥
⊥	⊥	⊥	2	n	m
⊥	⊥	⊥	3	k	⊥
⊥	⊥	⊥	4	⊥	m

(a) No mapping and no object identifier used

S.A	B	ID	T.C
x	y	1	⊥
z	⊥	2	⊥
⊥	x	3	⊥
⊥	n	2	m
⊥	k	3	⊥
⊥	⊥	4	m

(b) Mapping (S.B↔T.B, S.ID↔T.ID) used, but no object identifier used

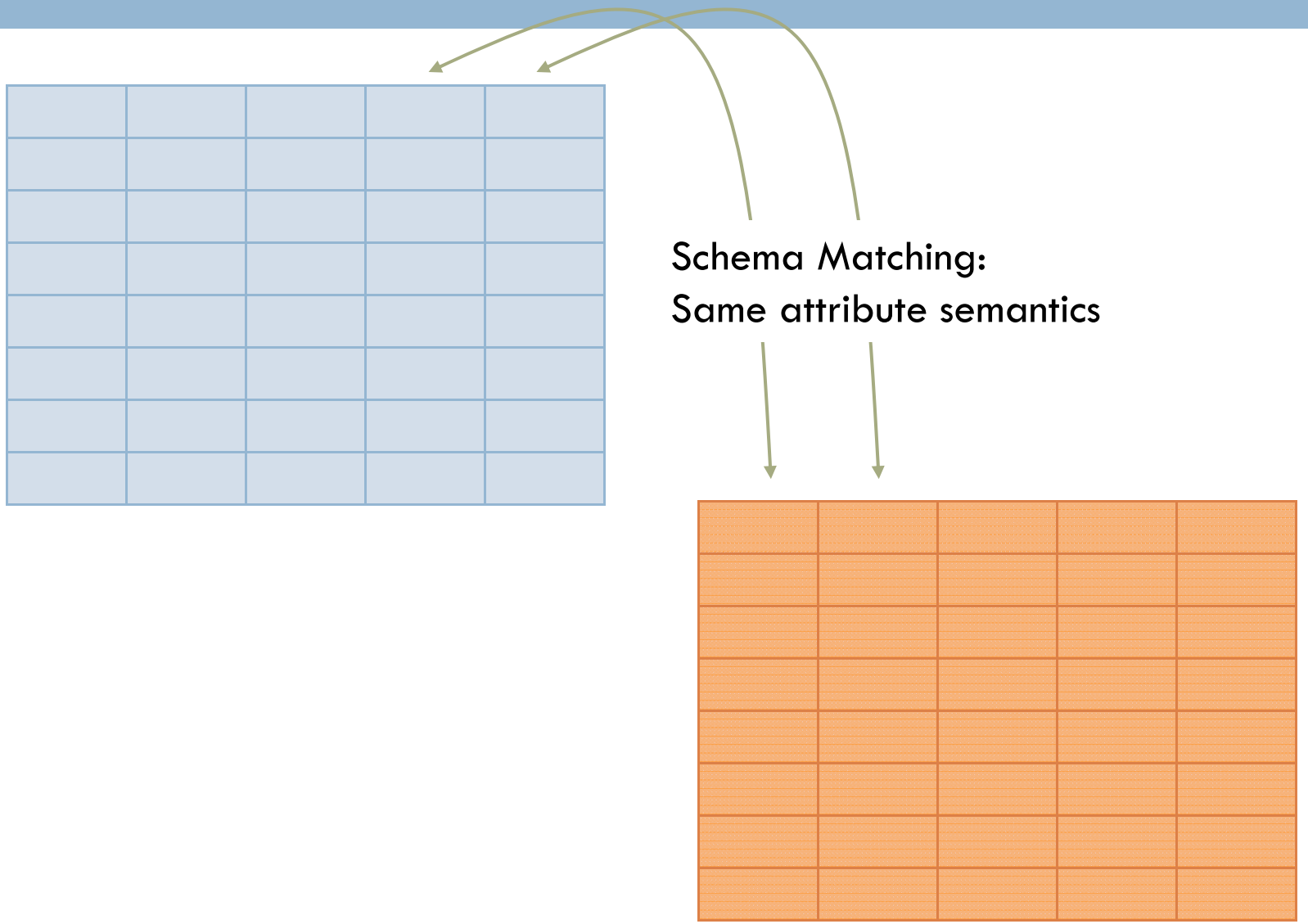
S.A	S.B	ID	T.B	T.C
x	y	1	⊥	⊥
z	⊥	2	n	m
⊥	x	3	k	⊥
⊥	⊥	4	⊥	m

(c) Partial mapping (S.ID↔T.ID) used, object identifier (S.ID, T.ID) used

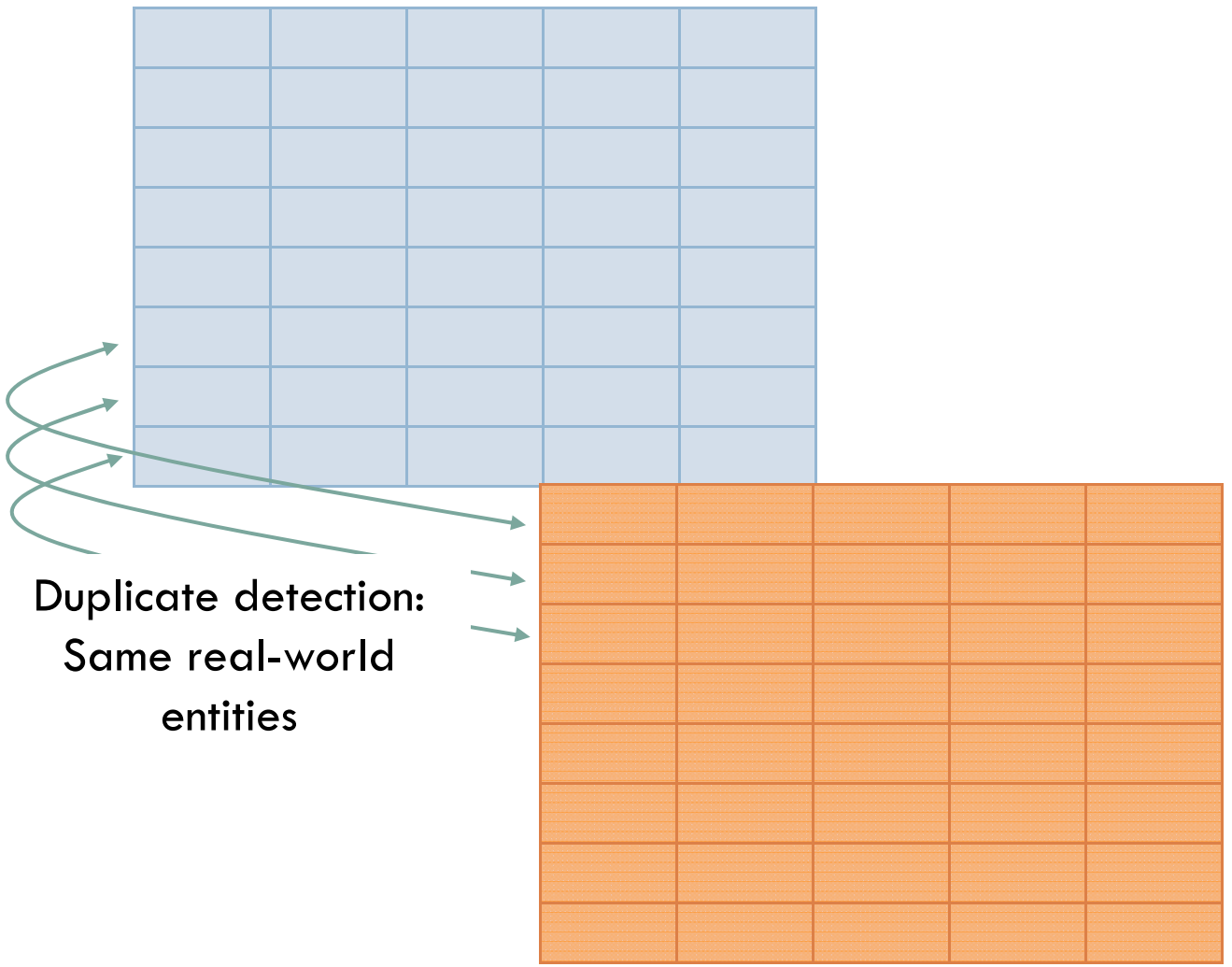
S.A	B	ID	T.C
x	y	1	⊥
z	n	2	m
⊥	x	3	⊥
⊥	⊥	4	m

(d) Full mapping (S.B↔T.B, S.ID↔T.ID) and object identifier (S.ID, T.ID) used

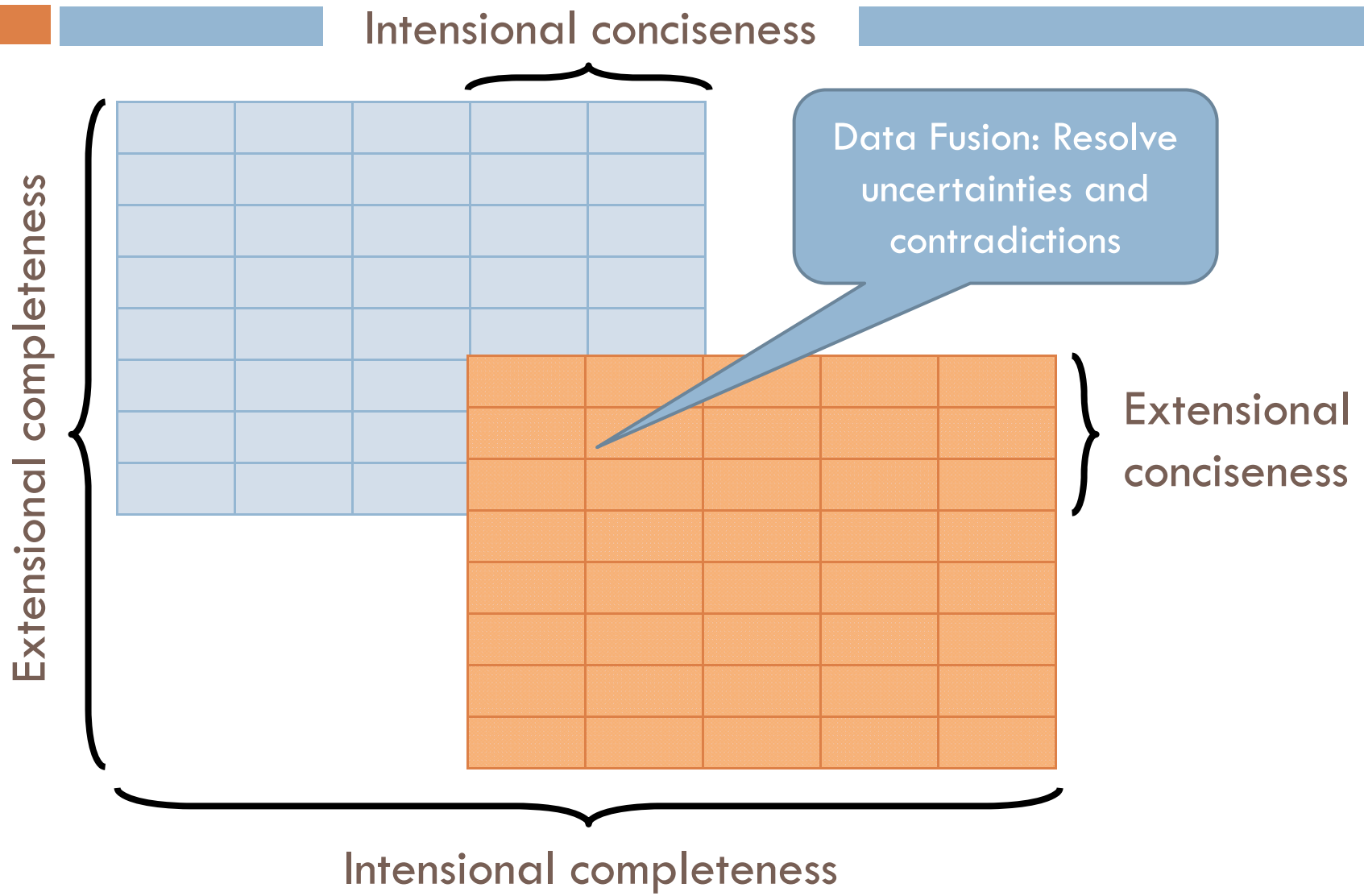
Completeness, Conciseness, and Correctness



Completeness, Conciseness, and Correctness

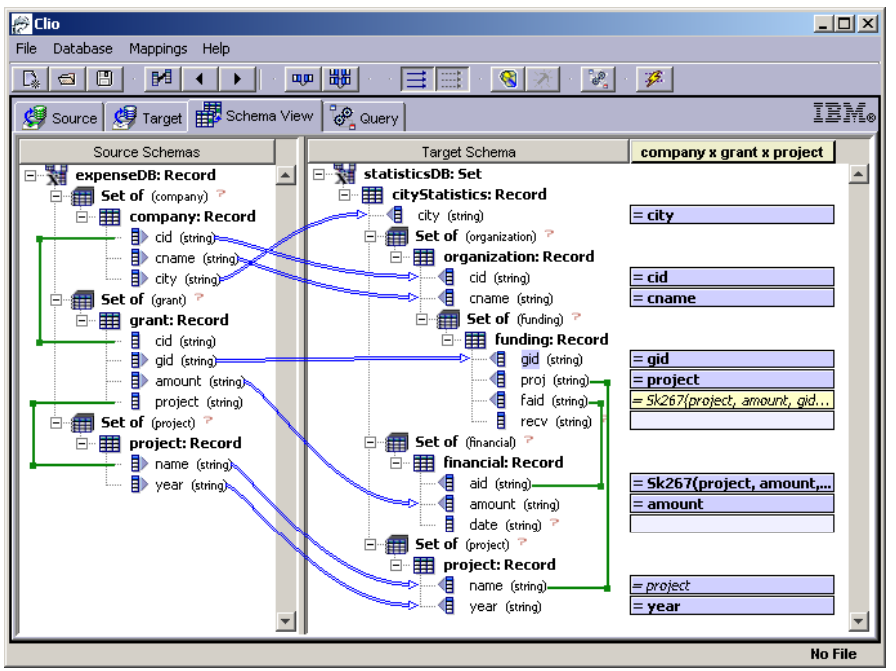


Completeness, Conciseness, and Correctness



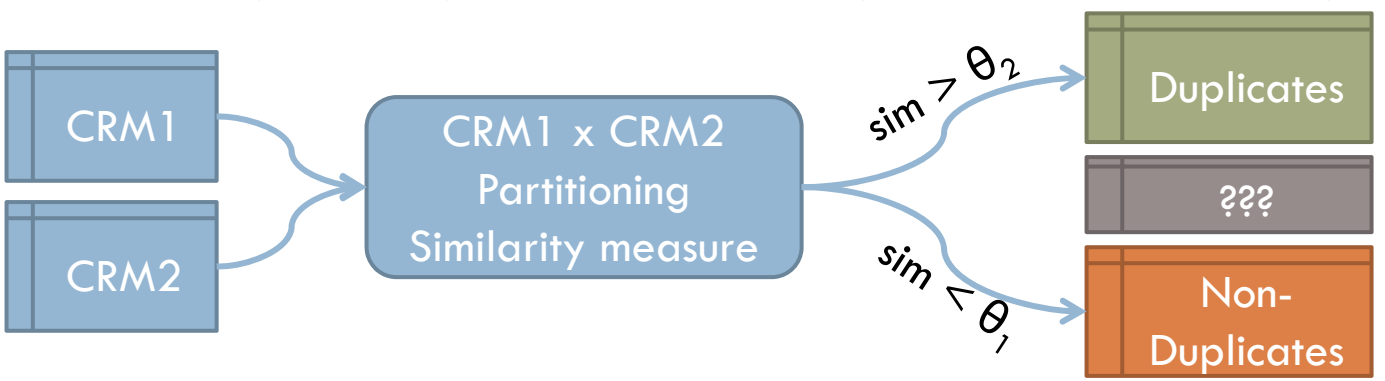
Schema Matching

- Problem
  - Given two schemata, find all correspondences between their attributes
- Difficulties
  - Schematic heterogeneity (synonyms & homonyms)
  - Data heterogeneity
  - n:m mappings
  - Transformation functions
  - User interaction
- Then: Derive a schema mapping



Duplicate Detection

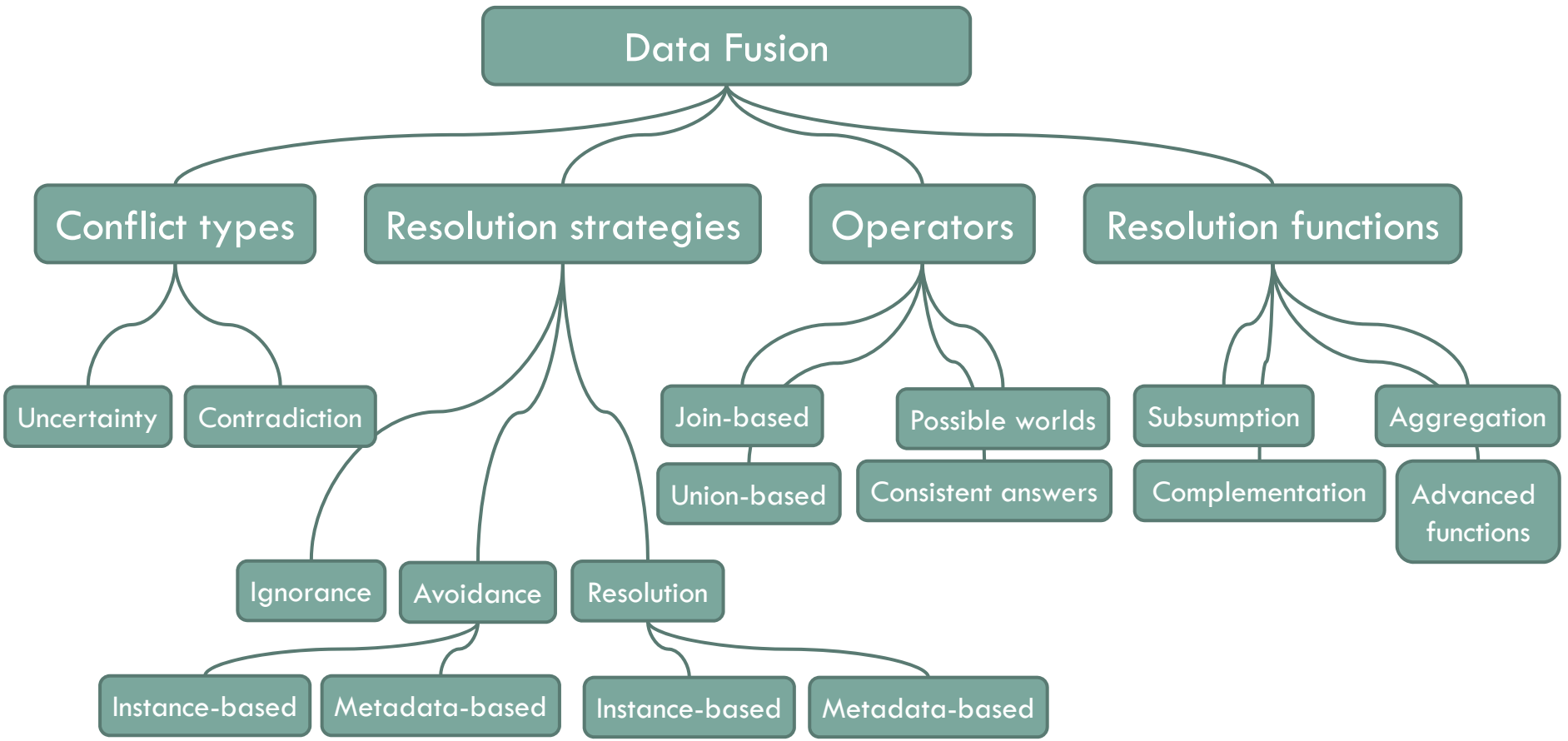
- Problem
  - Given one or more data sets, find all sets of objects that represent the same real-world entity.
- Difficulties
  - Duplicates are not identical
    - Similarity measures – Levenshtein, Soundex, Jaccard, etc.
  - Large volume, cannot compare all pairs
    - Partitioning strategies – Sorted neighborhood, Blocking, etc.



Data Fusion

- Problem
  - Given a duplicate, create a single object representation while resolving conflicting data values.
- Difficulties
  - Null values: Subsumption and complementation
  - Contradictions in data values
  - Uncertainty & truth: Discover the true value and model uncertainty in this process
  - Metadata: Preferences, recency, correctness
  - Lineage: Keep original values and their origin
  - Implementation in DBMS: SQL, extended SQL, UDFs, etc.

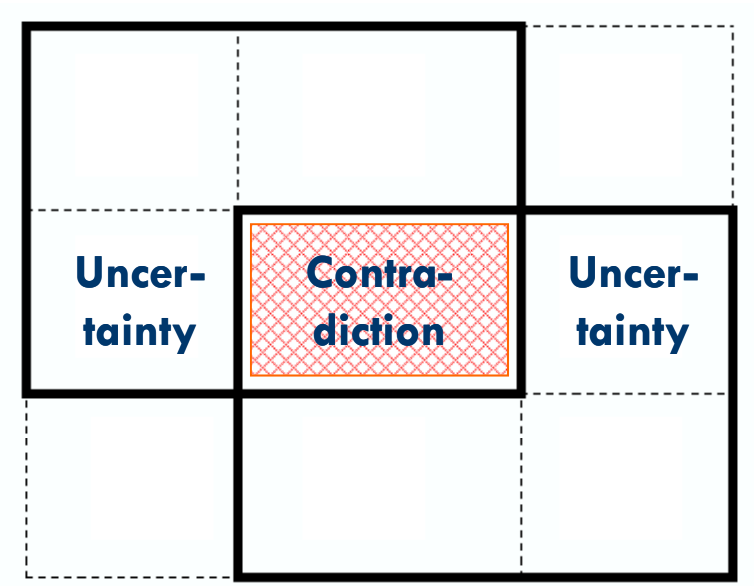
The Field of Data Fusion



Conflict resolution strategies and functions at the fusion stage

Uncertainty and Contradiction

- Uncertainty
  - NULL value vs. non-NULL value
  - "Easy" case
- Contradiction
  - Non-NULL value vs. (different) non-NULL value



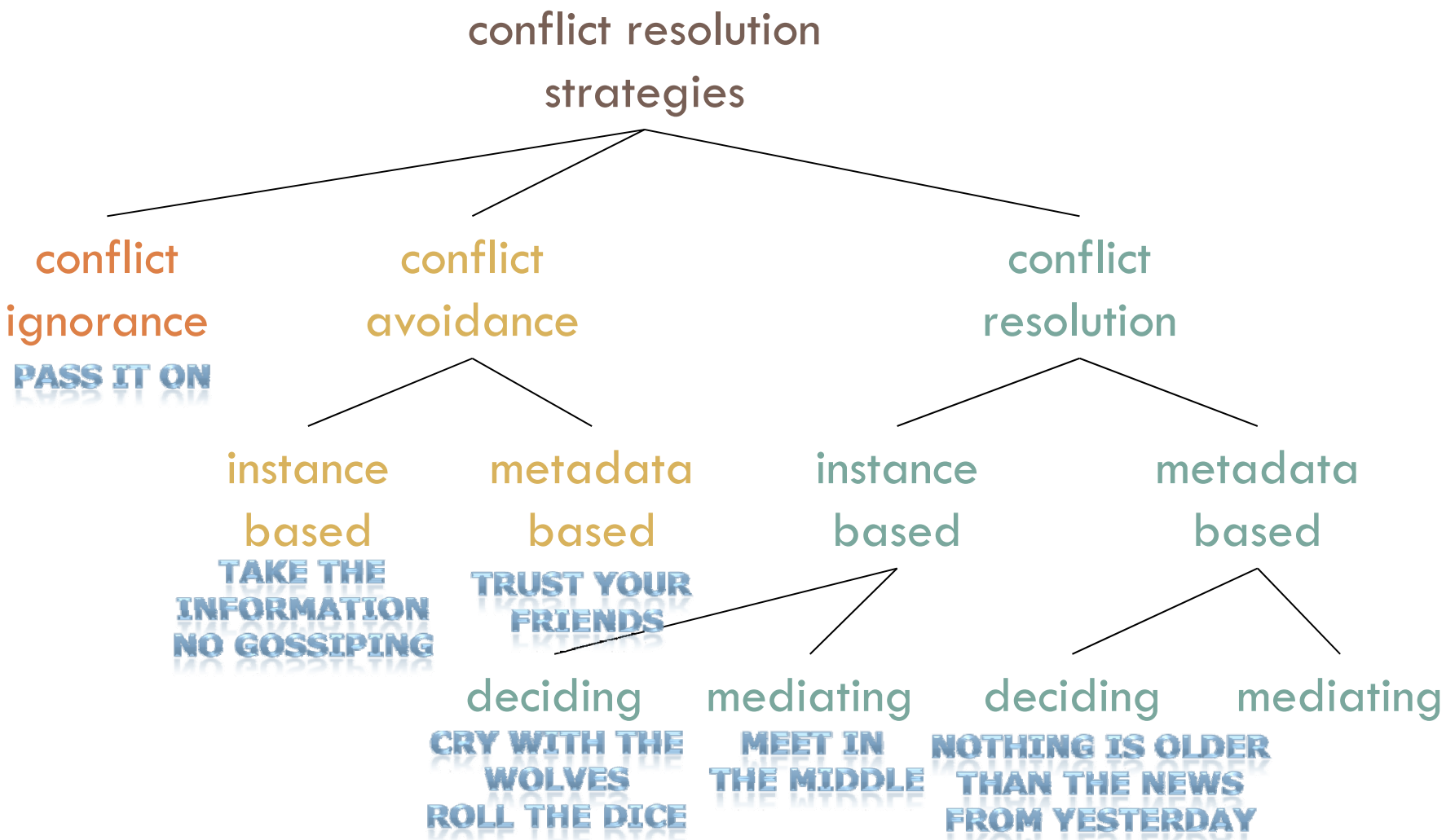
Semantics of NULL

- "unknown"
  - There is a value, but I do not know it.
  - E.g.: Unknown date-of-birth
- "not applicable"
  - There is no meaningful value.
  - E.g.: Spouse for singles
- "withheld"
  - There is a value, but we are not authorized to see it.
  - E.g.: Private phone line

Conflict handling strategies

- **Data conflicts**: missing data (value unknown or inapplicable, or withheld), uncertainty (Null vs Non-Null values), contradiction (at least two different Non-Niull values)
- **Three classes of conflict handling strategies**: *conflict ignorance*, *conflict avoidance*, and *conflict resolution*.
- **Conflict ignorance**: Pass it On (to the user), Consider All Possibilities.
- **Conflict avoidance**: Take the information (leave aside Null values), No Gossiping (only consistent answers are included into the result of a query), Trust your friends (e.g., prefer data fro one source over data from another sources)
- **Conflict resolution** (deciding and mediating strategies): Depending on the class, the choice of a value depends only on the data values (*instance based*) or takes some other information (*metadata based*) into account as well. Mediating strategies on the other hand may choose a value that does not necessarily exist among the conflicting values. Such strategies may decide on a new value, that has not been present among the conflicting values.

Classification of Strategies



What are conflict resolution functions

- Conflict resolution is defined as the application of a function ***fcr*** on a set of (possibly) conflicting values c resulting in a single resolved value ***s***.

- Aggregation functions known from SQL are examples for **single-column, parameter-free functions**, whereas CHOOSE DEPENDING is an example of a **multi-column parameter function.**

- A **function is mediating**, if it also allows other values as result than only the conflicting values. Such a function (e.g., AVERAGE, CONCAT) may create a new value that is from the same domain as the conflicting values.

- Other properties of conflicting functions:
  - monotonicity (MIN, MAX, FIRST, CONCAT are examples),
  - symmetry (MAX or VOTE are symmetric examples),
  - idempotence (COUNT, IGNORE, CONSTANT functions are not idempotent),
  - associativity (RANDOM, VOTE are not associative), NULL tolerant, etc.

Conflict Resolution Functions

Function	Description	Examples
Min, Max, Sum, Count, Avg	Standard aggregation	NumChildren, Salary, Height
Random	Random choice	Shoe size
Longest, Shortest	Longest/shortest value	First_name
Choose(source)	Value from a particular source	DoB (DMV), CEO (SEC)
ChooseDepending(val, col)	Value depends on value chosen in other column	city & zip, e-mail & employer
Vote	Majority decision	Rating
Coalesce	First non-null value	First_name
Group, Concat	Group or concatenate all values	Book_reviews
MostRecent	Most recent (up-to-date) value	Address
MostAbstract, MostSpecific, CommonAncestor	Use a taxonomy / ontology	Location
Escalate	Export conflicting values	gender
…	…	…

Example of resolution functions use (ADBIS 2005)

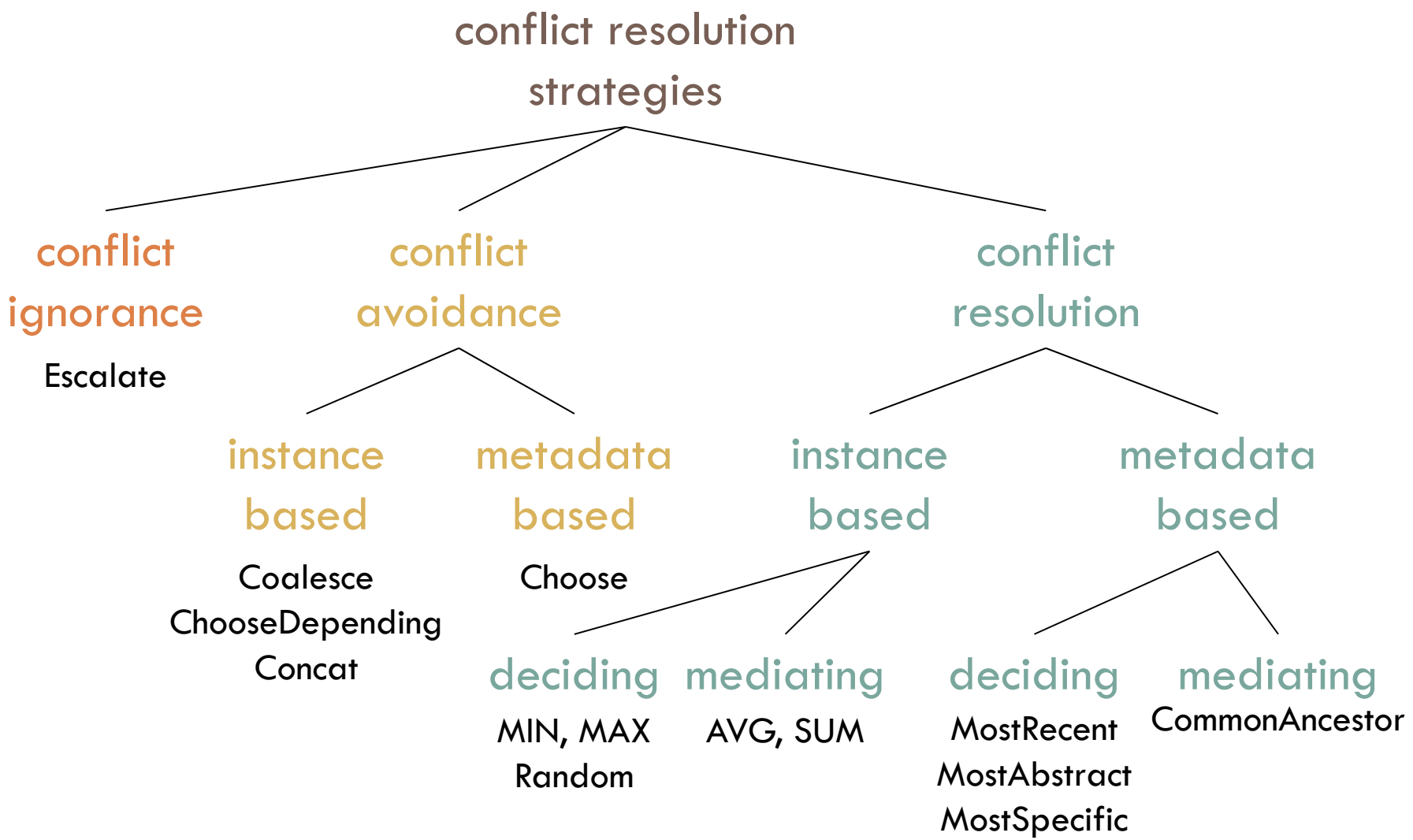
- SELECT ID,
  - RESOLVE (TITLE, Longest),
  - RESOLVE (YEAR, Vote),
  - RESOLVE (DIRECTOR, Choose(I)),
  - RESOLVE (PROD_COMP, MostComplete),
  - RESOLVE (PROD_COUNTRY, ChooseCorresponding(PROD_COMP)),
  - RESOLVE (GENRE, MostSpecific),
  - RESOLVE (RELEASE, Earliest),
  - RESOLVE (COLOR, Vote),
  - RESOLVE (DISTRIBUTOR, ChooseCorresponding(RELEASE))
- FUSE FROM I,M,C FUSE BY (ID)

Movie data is fused from the three sources (**I, M, and C**). FUSE FROM indicates combining the given tables by outer union instead of cross product. Equal movies are identified by the attribute ID. FUSE BY clause defines which objects are considered as the same real world objects, and are therefore fused into one single tuple.

Conflict resolution functions, choosing strategies and functions

- Conflict resolution functions are the **building blocks on the conceptual level** to implement conflict resolution strategies and are for use in systems or applications.
- **The functions can be used together with a variety of approaches, including the *data fusion operator.***
- ***Conflict resolution*** *is defined as the application of a function* f_{cr} *on a set of (possibly) conflicting values c resulting in a single resolved value s.*
- ***Kinds of functions****: single column; multi-column; metadata based; mediating.*
- ***Examples:*** *aggregation functions; First/Last; Highest quality; Most recent; Token union/Token intersection; Coalesce (returns the* 1^{st} *Non-Null value appearing).*
- Some **strategies have a direct equivalent among the functions** (e.g., the Coalesce function can be used to implement Take the information strategy).
- **Choosing a specific conflict handling strategy** for a certain problem is left to an expert user, as a high-level decision, which is then carried out by the integration system.

Classification of Functions



Conflict ignorance strategies

- Conflict ignorance describes a class of strategies that do not include a direct decision at all concerning data conflicts. When employing such a strategy one not even needs to look for, detect or be aware of data conflicts in the data, as this information is not needed and not used.

- **Pass it on.** This strategy simply takes all (possibly conflicting) representations and their attribute values, passes them on to the user or another application without changing them and lets the user or application decide how to handle possible conflicts.

- **Consider all possibilities.** This strategy tries to be as complete as possible by enumerating all eventualities and giving the user the choice among all possible combinations of attribute values.

- After applying these strategies, data conflicts remain in the data.

Conflict Avoidance strategies

- These strategies take a quick decision on whether to handle **inconsistencies** at all and if yes, which attribute values to use in that case. Because the decision is often done before regarding the actual values, these strategies are not always aware of a conflict. Therefore the name of conflict is **avoidance.**

- **Take the information. Th**is strategy takes existing information and leaves aside values. It is the natural way of dealing with uncertainties (***minimal union*** operator is a good example)

- **No gossiping**. Only consistent answers are included in the result of a query, leaving aside all inconsistent ones.

- **Trust your friends. Th**e intuition behind this strategy is to trust somebody else (can be determined automatically) to either provide the correct value or the correct strategy.

Conflict resolution strategies

- **Cry with the Wolves**: choosing the most common value among the conflicting values
- **Roll the Dice**: picks one at random
- **Better Bend than Break**: tries to invent a value that is as close as possible to all present values (mediating strategy)
- **Keep up to date**: uses the most recent value (based on metadata strategy)
- **Oops, I did it again**: a value is chosen if it has been chosen before (metadata based)

Strategy	Implementing the strategy: possible functions or reference
PASS IT ON	GROUP, CONCAT
CONSIDER ALL POSSIBILITIES	(Burdick et al., 2005; Yan and Özsu, 1999)
TAKE THE INFORMATION	COALESCE, LONGEST
NO GOSSIPING	(Arenas et al., 1999; Fuxman et al., 2005a)
TRUST YOUR FRIENDS	CHOOSE, CHOOSE DEPENDING, HIGHEST QUALITY, FIRST, MOST COMPLETE, CHOOSE CORRESPONDING
CRY WITH THE WOLVES	VOTE
ROLL THE DICE	RANDOM
BETTER BEND THAN BREAK	Average
MEET IN THE MIDDLE	AVERAGE, MEDIAN, MOST GENERAL
KEEP UP TO DATE	MOST RECENT, FIRST
IGNORANCE IS BLISS	LOOKUP(*source*), (Gigerenzer and Todd, 1999)
OUPS I DID IT AGAIN	LOOKUP(*source*), (Gigerenzer and Todd, 1999)
BETTER BEND THAN BREAK II	MOST GENERAL CONCEPT

Strategies and functions that can be used to realize them.

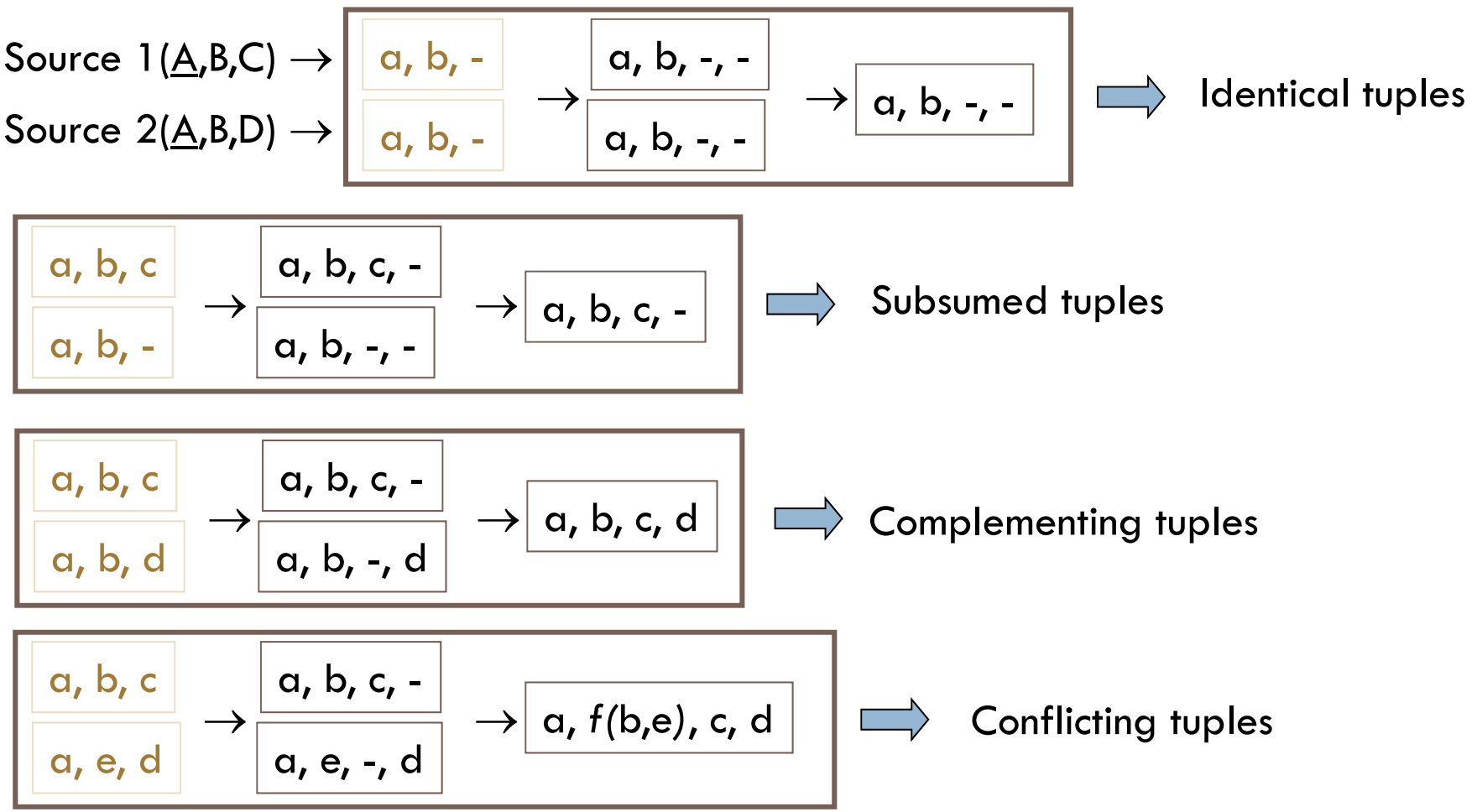
Data fusion and conflict resolution operators

Terminology: information preserving operators

- ***Value preservation*** for an operator: to be fully *value preserving*, exactly all values for all attributes of all described objects need to remain in the result.

  E.g., bag union is value preserving; set union or Cartesian product are *nonvalue preserving.*
- ***Object preservation***: values of attributes describing objects may get lost, but at least all objects that have been described in the sources are also included in the result.
- ***Uniqueness-preserving*** operators should preserve in the result the unique values in the sources
- ***Uniqueness-enforcing*** operators create a result that contains an attribute that is unique.
- The ideal data fusion operator, creating **complete and consistent** answers, should **preserve as many values and objects** as possible while at the same time **enforcing or preserving uniqueness** and resolving conflicts

Data Fusion Goals



Data fusion strategies and operations to be considered

- Survey of the data fusion techniques and information integration systems developed so far and comparison of their abilities and limitations are presented. We show how the different strategies can be implemented as part of an information integration system.
- *Data fusion* is considered as a third main step in the general information integration process after *schema mapping, entity resolution (including duplicate detection).*
- An exampls of two special case operators for handling missing information: *subsumption* and *complementation* will be introduced .
- The *subsumption/minimum union* operation is one possible way of implementing a specific data fusion strategy. An efficient implementation of the operation will be given.
- Similarly to *subsumption* and *minimum union*, *complement union* is the combination of *complementation* and *outer union.*

Relational Operators – Overview

- Identical tuples
  - UNION, OUTER UNION
- Subsumed tuples (uncertainty)
  - MINIMUM UNION
- Complementing tuples (uncertainty)
  - COMPLEMENT UNION, MERGE
- Conflicting tuples (contradiction)
  - Relational approaches: Match, Group, Fuse, …
- Other approaches
  - Possible worlds, probabilistic answers, consistent answers

Operator	Notation	Description
selection	σ_c (R)	Selects tuples from R that fulfill condition *c*
projection/renaming	π_L (R)	Extended projection as in (Ullman et al., 2001), including renaming and basic computations, given in *L* and applied on relation R
sort	τ_o (R)	Sorts the tuples in a specified order *o*
distinct	δ (R)	Removes exact duplicates from relation R
grouping/aggregation	γ_L (R)	Grouping and aggregation as given in attribute list *L* and applied on relation R
natural join	R ⋈ S	Joins two relations R and S on attributes with same names
join	R $\bowtie_c$ S	Joins two relations R and S using join condition *c*
cartesian product	R × S	Computes the cartesian product (cross product) of two relations R and S
left outer join	R $⟕_c$ S	Computes a left outer join of R and S using join condition *c*
right outer join	R $⟖_c$ S	Computes a right outer join of R and S using join condition *c*
full outer join	R $⟗_c$ S	Computes a full outer join of R and S using join condition *c*
union	R ∪ S	Union of relations R and S

Notation used for relational operators, based on the notation used in (Ullman et al., 2001)

Reminder on ***outer join***

- An **outer join** does not require each record in the two joined tables to have a matching record. The joined table retains each record—even if no other matching record exists. Outer joins subdivide further into ***left outer joins, right outer joins, and full outer joins,*** depending on which table's rows are retained (left, right, or both).

- The result of a ***left outer join*** for tables A and B always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B). This means that if the `ON` clause matches 0 (zero) records in B (for a given record in A), the join will still return a row in the result (for that record)—but with NULL in each column from B.

- A ***right outer join*** closely resembles a left outer join, except with the treatment of the tables reversed. Every row from the "right" table (B) will appear in the joined table at least once. If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in B.

Definition (Outer Union)

The binary outer union *operator* ($\uplus$) *combines two relations and is an extension of the* union *operator to tables with arbitrary schemata. The output schema is the union of the input schemata, where duplicate attributes are removed according to given equivalences (same name, if nothing else is given). Likewise, the resulting tuple set is the union of all input tuple sets. Given two relations R and S and implicit modeling of missing information, then the* outer union $T = R \uplus S$ *is given by*

$$T = (S_R \cup S_S, T_R \cup T_S)$$
$$= \left((newname, \{a_i^r\} \cup \{a_j^s\}, \{dom(a_i^r)\} \cup \{dom(a_j^s)\}), \{\alpha(t_i^r)\} \cup \{\alpha(t_j^s)\}\right)$$

□

The two relations R and S with attribute sets $A_R = \{A, B, C, D\}$ and $A_S = \{C, D, E, F\}$ result in $T = (R \uplus S)$ having attribute set $A_T = \{A, B, C, D, E, F\}$. Combining more than two relations is easily possible as *outer union* is commutative. If we employ explicit modeling of missing information, tuples must be padded by NULL values in the attributes that are contained in the attribute set of the respective other relation.

Comment on ***outer union***

As far as the *outer union* is **not a part of the SQL standard**, everywhere it is assume that it should be rewritten as follows:

- **Outer union combines two nonunion compatible relations** by first padding attributes that are not in common with null values (to make the relations union compatible) and then performing a regular union [Galindo-Legaria 1994]. **Like union, outer union is not value- but object preserving. It is not uniqueness preserving, and conflicts are ignored.** Example:
- (SELECT Name, Age, Status, Address, Field, Library, NULL as Phone FROM U1)

UNION

(SELECT Name, Age, Status, Address, Field, NULL as Library, Phone FROM U2)

Definition (Tuple Subsumption (Galindo-Legaria, 1994))

A tuple $t_1 \in T$ subsumes *another tuple $t_2 \in T$ ($t_1 \sqsupset t_2$), if (1) t_1 and t_2 have the same schema, (2) t_2 contains more* NULL *values than t_1, and (3) t_2 coincides in all* NON-NULL *attribute values with t_1.* □

Under implicit modeling of missing information, tuple subsumption is equal to set containment. A tuple t_1 *subsumes* another tuple t_2 ($t_1 \sqsupset t_2$), if $t_1 \supset t_2$. Tuple subsumption is a transitive relationship, so if $t_1 \sqsupset t_2$ and $t_2 \sqsupset t_3$, then also $t_1 \sqsupset t_3$. Tuple subsumption is neither symmetric nor reflexive.

Definition (Subsumption Operator)

We use the unary subsumption operator β to denote the removal of subsumed tuples from a relation R:

$$\beta(R) = \left\{ t \in R \mid \neg \exists t' \in R : t' \sqsupset t \right\}$$

Data Fusion: Union-based Approaches

Union approaches for fusion

- **Union approaches**: all information from the source tables are retained and **extensional as well as intensional completeness are easily reached**
- Only unnecessary information (**i.e., duplicate tuples or subsumed tuples**) is deleted, increasing conciseness.
- However, in most cases, too many tuples and therefore too much information is kept, requiring an additional step towards a more satisfying level of conciseness. This finally **results in one single tuple per real-world object**.
- **The intuitive way of increasing conciseness together with the union approach is to apply grouping and aggregation**.

Union-based operators (1)

- *1) **Union, Outer Union, and Minimum Union**.* The **union** operator (with set semantics) combines the tuples of two union-compatible relations and **removes exact duplicates** (tuples that coincide in all attribute values). **Outer union** overcomes one restriction of union and combines two nonunion-compatible relations by first padding attributes that are not in common with null values. **Union and outer union are** not value- but **object preserving**. They are **not uniqueness preserving, and conflicts are ignored.** The **minimum union operator** is defined by as the result of an **outer union from which subsumed tuples have been removed.**

1) *2) **Merge, Prioritized Merge**.* The merge operator can be described as **the union of two outer joins**. There exists a variant, namely **prioritized merge**, that is used to prioritize among the sources and **give preference values** from one source. Merge is well suited in scenarios where **many null values could be complemented**, such as when **data sources are sparsely filled** or only slightly overlap in their schemata

Union-based operators (2)

1) *3) **Increasing Conciseness in Union Results**.* As semantically equivalent attributes have already been matched, different representations reside in different rows of a union result. Therefore, the obvious way of **removing possible data conflicts is by grouping representations of a single real-world object together and aggregating the values of the attributes**. The grouping part of this operation is **uniqueness enforcing in the grouping attribute(s)**. In general, the result is, after grouping, similar to the union result, extensionally complete. **Increasing conciseness in this way requires one or more attributes that serve as ID** to identify same real-world objects.

- SELECT Name, max(Age), vote(Status), most_recent(Address), concat(Field),
- coalesce(Library), coalesce(Phone)
- FROM *union(U1,U2)*
- GROUP BY Name

Subsumption and Minimum Union

- ***Minimum union*** is one of two alternative operators that both aim at resolving a special type of data conflict called uncertainty, , realizing the *Take the information* strategy.
- ***Minimum union* combines an *outer union* with the removal of tuples** that are contained in other tuples, so called *subsumed* tuples
- **As far as the *outer union* is not a part of the SQL standard, everywhere it is assume that it should be rewritten**
- **Tuple subsumption**: A tuple t1 subsumes another tuple t2, if t1 and t2 have the same schema, t2 contains more Null values than t1 and t2 coincides in all Non Null attribute values with t1.
- **Unary subsumption** operator denotes the removal of subsumed tuples from a relation.
- **The binary minimum union operator** is the combination of outer union and subsumption where subsumed tuples are removed from the result of the outer union of the two input relations.

Minimum Union

- (Normal) union: eliminate exact duplicates
- Minimum union: eliminate subsumed tuples
  - A tuple t1 subsumes a tuple t2 if it
    - uses the same schema
    - has less NULL values
    - coincides in all non-NULL values
- Two steps
  - Outer union
  - Subsumption

Definition (Minimum Union)

The binary minimum union *operator* ($\oplus$) *is the combination of* outer union *and* subsumption *where subsumed tuples are removed from the result of the* outer union *of the two input relations:*

$$A \oplus B = \beta\,(A \uplus B)$$

□

The *minimum union* operator is commutative and associative. The following example shows tuples from two tables that are combined by *outer union* and where subsumed tuples are removed subsequently:

Relation R

tuple	A	B	C	D
1	a	b	c	d
2	a	⊥	c	d
3	a	b	⊥	d

Relation S

tuple	C	D	E	F
4	c	⊥	⊥	⊥
5	c	d	⊥	⊥
6	d	⊥	e	f

Relation $R \oplus S$

tuple	A	B	C	D	E	F
1	a	b	c	d	⊥	⊥
6	⊥	⊥	d	⊥	e	f

Relational Operators: Minimum Union

Name	Age	Country
Adam	20	England
Ben	23	USA
Chris	21	Canada

$\oplus$

Name	Age	Height
Adam	20	NULL
Ben	23	174
Chris	40	NULL

=

Name	Age	Country	Height
Adam	20	England	NULL
Adam	20	NULL	NULL
Ben	23	USA	NULL
Ben	23	NULL	174
Chris	21	England	NULL
Chris	40	NULL	NULL

$\longrightarrow$ subsmp

Name	Age	Country	Height
Adam	20	England	NULL
Ben	23	USA	NULL
Ben	23	NULL	174
Chris	21	England	NULL
Chris	40	NULL	NULL

Minimum Union

- Union: Elimination of exact duplicates
- Minimum Union: Elimination of subsumed tuples
  - Outer union
  - Subsumption
- Rewriting in SQL using DWH extensions (Windows) and assuming existence of favorable ordering [RPZ04]

A	B	C
a	b	c
e	f	g
m	n	o

⟗

A	B	D
a	b	⊥
e	f	h
m	p	⊥

=

A	B	C	D
a	b	c	⊥
a	b	⊥	⊥
e	f	g	⊥
e	f	⊥	h
m	n	o	⊥
m	p	⊥	⊥

A tuple t1 subsumes a tuple t2, if it has same schema, has less NULL-values, and coincides in all non-NULL-values.

R↓

A	B	C	D
a	b	c	⊥
e	f	g	⊥
e	f	⊥	h
m	n	o	⊥
m	p	⊥	⊥

Complementation and Complement Union

- ***Complement union*** is another operator that aims at resolving the uncertainty data conflict thus realizing the Take the information strategy . It combines an *outer union* with the combination of tuples that complement each other.
- ***A tuple t1 complements a tuple t2*** *if t1 and t2 comply to the same schema, values of corresponding attributes in t1 and t2 are equal or one of them is Null, t1 and t2 are neither equal nor do they subsume one another, and t1 and t2 have at least one attribute value combination in common.* Tuple complementation is symmetric rel.
- ***A complementing set CS*** *of tuples of a relation R* = (*S*, *T*) *is a subset of tuples from T where for each pair* t_i, *tj of tuples from CS it holds that* t_i complements t_j. *A complementing set S1 is a* maximal complementing set *(MCS) if there*
- *exist no other complementing set S2 s.t. S1 is a subset of S2.*
- The unary complementation operator replaces each existing maximal complementing set MCS of a relation R by the single complement tuple tci of all tuples in MCSi.
- Note that equal tuples (exact duplicates) do not complement each other and are therefore retained after complementation. Under set semantics, exact duplicates do not exist anyway. Under bag semantics, exact duplicates do exist.

Complementation example

The following examples demonstrates the application of *complementation*:

Relation *R*

tuple	A	B	C	D
1	a	b	c	d
2	a	b	c	⊥
3	a	⊥	c	d
4	a	b	⊥	d
5	a	⊥	⊥	e
6	b	⊥	⊥	d

Relation κ (R)

tuple	A	B	C	D
1	a	b	c	d
2+3+4	a	b	c	d
2+5	a	b	c	e
6	b	⊥	⊥	d

Complement Union

- Minimum union: eliminate subsumed tuples
- Complement union: eliminate complemented tuples
  - A tuple t1 complements a tuple t2 if
    - uses the same schema
    - coincides in all non-NULL values
  - Less strict than subsumption
- Two steps
  - Outer union
  - Complementation

Definition (Complement Union)

The binary complement union *operator* ($\boxplus$) *is the combination of* outer union *and* complementation. *Complementing tuples in the result of the* outer union *of the two input relations A and B are combined:*

$$A \boxplus B = \kappa\left(A \uplus B\right)$$

□

Relation *R*

tuple	A	B	C	D
1	a	⊥	c	d
2	⊥	b	c	⊥
3	⊥	c	c	d

Relation *S*

tuple	C	D	E	F
4	c	e	⊥	f
5	c	⊥	⊥	g
6	c	e	e	⊥

Relation $R \boxplus S$

tuple	A	B	C	D	E	F
1+2+6	a	b	c	d	⊥	g
1+3+6	a	c	c	d	⊥	g
2+4+6	⊥	b	c	e	e	f
2+5+6	⊥	b	c	e	e	g

Complement Union – Proposal

- Elimination of complementing tuples
  - Outer union
  - Complementation
- No known SQL rewriting

A	B	C
a	b	c
e	f	g
m	n	o

$\dot{\cup}$

A	B	D
a	b	⊥
e	f	h
m	p	⊥

=

A	B	C	D
a	b	c	⊥
a	b	⊥	⊥
e	f	g	⊥
e	f	⊥	h
m	n	o	⊥
m	p	⊥	⊥

R⇅

A	B	C	D
a	b	c	⊥
e	f	g	h
m	n	o	⊥
m	p	⊥	⊥

A tuple t1 complements a tuple t2, if it has same schema and coincides in all non-NULL-values.

Includes duplicate removal and subsumption

Relational Operators: Complement Union

Name	Age	Country
Adam	20	England
Ben	23	USA
Chris	21	Canada

$\uplus$

Name	Age	Height
Adam	20	NULL
Ben	23	174
Chris	40	NULL

=

Name	Age	Country	Height
Adam	20	England	NULL
Adam	20	NULL	NULL
Ben	23	USA	NULL
Ben	23	NULL	174
Chris	21	England	NULL
Chris	40	NULL	NULL

→ cmpl

Name	Age	Country	Height
Adam	20	England	NULL
Ben	23	USA	174
Chris	21	England	NULL
Chris	40	NULL	NULL

Merge & Prioritized Merge

- **Merge**
  - Combine Join & Union into new operator
  - Idea: make 2 versions for each common attribute, one *favoring* the first source, the other *favoring* the second source
  - NULL values are replaced by using COALESCE

    COALESCE is a function and returns the first non-NULL value
- **Prioritized Merge**
  - Same as merge, but contradictions get eliminated

Merge example

(**SELECT**R.Name, **COALESCE(**R.Age, S.Age**)**, R.Country, S.Height
FROM R **LEFT OUTER JOIN** S **ON**R.Name = S.Name
)
UNION
(**SELECT** S.Name, **COALESCE(**S.Age, R.Age**)**, R.Country, S.Height
FROM R **RIGHT OUTER JOIN**S **ON** R.Name = S.Name)

MERGE – more examples

```
( SELECT U1.Name, coalesce(U1.Age, U2.Age) AS Age,
    COALESCE(U1.Status, U2.Status) AS Status,
    COALESCE(U1.Address, U2.Address) as Address,
    COALESCE(U1.Field, U2.Field) as Field,
    U1.Library, U2.Phone
FROM U1 LEFT OUTER JOIN U2 ON U1.Name = U2.Name )
UNION
( SELECT U2.Name, COALESCE(U2.Age, U1.Age) AS Age,
    COALESCE(U2.Status, U1.Status) AS Status,
    COALESCE(U2.Address, U1.Address) as Address,
    COALESCE(U2.Field, U1.Field) as Field,
    U1.Library, U2.Phone
FROM U1 RIGHT OUTER JOIN U2 ON U1.Name = U2.Name )
```

The COALESCE function shows how to remove uncertainties: the missing AGE, FIELD, and STATUS values are replaced by the corresponding value from the other tuple

Relational Operators: Merge

Name	Age	Country
Adam	20	England
Ben	23	USA
Chris	21	Canada
Chris	21	NULL
David	33	Spain

$|\bowtie$

Name	Age	Height
Adam	20	NULL
Ben	23	174
Chris	19	NULL

=

Name	Age	Country	Height
Adam	20	England	NULL
Ben	23	USA	174
Chris	21	Canada	NULL
Chris	21	NULL	NULL
David	33	Spain	NULL

$\cup$

Name	Age	Country
Adam	20	England
Ben	23	USA
Chris	21	Canada
Chris	21	NULL
David	33	Spain

$\bowtie|$

Name	Age	Height
Adam	20	NULL
Ben	23	174
Chris	19	NULL

=

Name	Age	Country	Height
Adam	20	England	NULL
Ben	23	USA	174
Chris	19	Canada	NULL
Chris	19	NULL	NULL

Merge and Prioritized Merge

- Mixes Join and Union to a new operator [GPZ01]
  - Idea: Build two versions for each common attribute, one "favoring" S1, the other "favoring" S2.
  - Nulls in a source are replaced using COALESCE.
  - Fuses complementing tuples, but only for inter-source duplicates
  - Priorization possible: Removes conflicting tuples from right relation.

```
( SELECT R.A, COALESCE(R.B, S.B), R.C, S.D
  FROM R LEFT OUTER JOIN S ON R.A = S.A )
            UNION
( SELECT S.A, COALESCE(S.B, R.B), R.C, S.D
  FROM R RIGHT OUTER JOIN S ON R.A = S.A )
```

Merge and Prioritized Merge

A is real-world ID

A	B	C
a	b	c
e	f	g
m	n	o
m	n	⊥
q	r	s

$|\bowtie$

A	B	D
a	b	⊥
e	f	h
m	p	⊥

$=$

A	B	C	D
a	COAL(b,b)	c	⊥
e	COAL(f,f)	g	h
m	COAL(n,p)	o	⊥
m	COAL(n,p)	⊥	⊥
q	r	s	⊥

$=$

A	B	C	D
a	b	c	⊥
e	f	g	h
m	n	o	⊥
m	n	⊥	⊥
q	r	s	⊥

$\cup$

A	B	C	D
a	b	c	⊥
e	f	g	h
m	p	o	⊥
m	p	⊥	⊥

A	B	C
a	b	c
e	f	g
m	n	o
m	n	⊥
q	r	s

$\bowtie|$

A	B	D
a	b	⊥
e	f	h
m	p	⊥

$=$

A	B	C	D
a	COAL(b,b)	c	⊥
e	COAL(f,f)	g	h
m	COAL(p,n)	o	⊥
m	COAL(p,n)	⊥	⊥

$=$

A	B	C	D
~~a~~	~~b~~	~~c~~	~~⊥~~
~~e~~	~~f~~	~~g~~	~~h~~
m	p	o	⊥
m	p	⊥	⊥