An introduction to Big Data Integration

Letizia Tanca
Politecnico di Milano
Trends in Data Integration

✓ Pre-history: focused on challenges that occur within enterprises
✓ The Web era:
  • scaling to a much larger number of sources
  • Handling sources with less structure.
✓ Nowadays:
  • large scientific experiments rely on data management for progress.
  • People create data fragments (breadcrumbs) by interacting with services on the Web.
  • Massive use of social media and smart devices
  • User-generated content merges with the Internet of Things
  • Users as sensors and actuators

How do we make sense of this mass of data?

L. Tanca
The Data Integration problem is:

Combining data coming from different data sources, providing the user with a unified vision of the data

➔ Detecting correspondences between similar concepts that come from different sources, and conflict solving
The four V’s in data integration

- **Volume:** Not only can each data source contain a huge volume of data, but also the number of data sources has grown to be in the millions.

- **Velocity:** As a direct consequence of the rate at which data is being collected and continuously made available, many of the data sources are very dynamic.

- **Variety:** Data sources (even in the same domain) are extremely heterogeneous both at the schema level, regarding how they structure their data, and at the instance level, regarding how they describe the same real world entity, exhibiting considerable variety even for substantially similar entities.

- **Veracity:** Data sources (even in the same domain) are of widely differing qualities, with significant differences in the coverage, accuracy and timeliness of data provided. This is consistent with the observation that “1 in 3 business leaders do not trust the information they use to make decisions.”

(Xin Luna Dong, Divesh Srivastava, VLDB2013 tutorial)
The steps of Data Integration: how is it done?

- **Schema Reconciliation**: mapping the data structure

- **Record Linkage**: data matching based on the same content

- **Data Fusion**: reconciliation of non-identical content
Relevant Ways of Integrating Database Systems

1. Use a materialized data base (data are merged in a new database) $\rightarrow$ Extract-Transform-Load Systems
   $\rightarrow$ Data Warehouses: Materialized integrated data sources

2. Use a virtual non-materialized data base (data remain at sources) $\rightarrow$
   - Enterprise Information Integration (EII) (or Data Integration) Systems (common front-end to the various datasources)
   - Data Exchange (source-to-target)
Materialized
Virtual

Global schema (view)

Query

Relational DB

OO DB

Excel sheet

on-line

On-line
Data Exchange
(e.g. IBM CLIO)

Query

“Target” DB

answer

Reformulated query

“Source” DB
Views

• Also called external schemata
• Sintax:
  ```sql
  create view ViewName [ (AttList) ] as SQLquery
  [ with [ local | cascaded ] check option ]
  ```
Recall: the Data Integration problem is:

Combining data coming from different data sources, providing the user with a unified vision of the data

→ Detecting correspondences between similar concepts that come from different sources, and conflict solving
The possible situations

Data integration problems arise even in the simplest situation: unique, centralized DB...

...and it becomes more and more complex

...up to the extreme case of transient, dynamic, initially unknown data sources...
An integrated centralized DB
Distributed DB

It is the simplest case of non-centralized DB: NOT a matter of data integration

- Often, data for the same organization
- Integrated a-priori: same design pattern as in the centralized situation, indeed we have homogeneous technology, data model, thus the schema integration problems are as above
- At distribution design time, design decisions on:
  - Fragmentation:
    - Vertical
    - Horizontal
  - Allocation
    - Replication
Data Integration with various data sources

query    answer

QUERY FRONT-END

DATA SOURCE 1 (RDBMS)
DATA SOURCE 2 (XML)
DATA SOURCE 3 (WWW)
The steps of Data Integration: how is it done?

- Schema Reconciliation: mapping the data structure
- Record Linkage: data matching based on the same content
- Data Fusion: reconciliation of non-identical content
If the sources have a schema

1. Global schema (view) conceptual design: **conflict resolution** and **restructuring**
2. Conceptual-to-logical translation (of the global schema, of the single subschemata)
3. **Reconciliation** of the global logical schema with the single schemata (logical view definition)
View integration and restructuring

a. Related concept identification
b. Conflict analysis and resolution
c. Conceptual Schema integration
Related Concepts’ identification

• Ex:
  o employee, clerk
  o exam, course
  o code, num

• Not too difficult if manual
• Very difficult if automatic – this is the extreme case
Conflict analysis

NAME CONFLICTS

• HOMONYMS

• SYNONIMS

Product  price (production price)

Product  price (sale price)

Department

Division
Conflict analysis

TYPE CONFLICTS

• **in a single attribute** (e.g. NUMERIC, ALPHANUMERIC, ...)
  
  e.g. the attribute “gender”:
  
  o Male/Female
  o M/F
  o 0/1
  o In Italy, it is implicit in the “codice fiscale” (SSN)

• **in an entity type**
  
  different abstractions of the same real world concept produce different sets of properties (attributes)
Conflict analysis

DATA SEMANTICS

• different currencies (euros, US dollars, etc.)
• different measure systems (kilos vs pounds, centigrades vs. Farhenheit.)
• different granularities (grams, kilos, etc.)
Conflict analysis

STRUCTURE CONFLICTS

Person

GENDER

MAN
WOMAN

EMPLOYEE

DEPARTMENT

PROJECT

BOOK

PUBLISHER

EMPLOYEE

PROJECT

BOOK

PUBLISHER
Conflict analysis

- DEPENDENCY (OR CARDINALITY) CONFLICTS

![Entity-Relationship Diagram]

- EMPLOYEE
  - 1:1
  - 1:n
  - DEPARTMENT
  - 1:1
  - PROJECT
  - 1:n

- EMPLOYEE
  - 1:n
  - PROJECT
  - 1:n
Conflict analysis

• KEY CONFLICTS
Schema Integration

• Conflict resolution
• Production of a new conceptual schema which expresses (as much as possible) the same semantics as the schemata we wanted to integrate
• Production of the transformations between the original schemata and the integrated one: $V_1(DB), V_2(DB), \ldots, V_3(DB)$
A first case

• The data sources have the same data model
• Adoption of a global schema
• The global schema will provide a
  • Reconciled
  • Integrated
  • Virtual

view of the data sources
Architecture with a uniform data model
Mapping between the global logical (mediated) schema and the single source schemata (logical view definition)

- Two basic approaches
  - **GAV** (Global As View)
  - **LAV** (Local As View)
- The same approach can be used also in case of different data models
- In that case also a **model transformation** is required (we’ll see it later)
GAV (Global As View)

- Up to now we supposed that the global schema be derived from the integration process of the data source schemata
- Thus the global schema is expressed in terms of the data source schemata
- Such approach is called the Global As View approach
The other possible ways

LAV (Local As View)

• The global schema has been designed independently of the data source schemata
• The relationship (mapping) between sources and global schema is obtained by defining each data source as a view over the global schema

GLAV (Global and Local As View)

• The relationship (mapping) between sources and global schema is obtained by defining a set of views, some over the global schema and some over the data sources
GAV example

**SOURCE 1**

Product\((\text{Code}, \text{Name}, \text{Description}, \text{Warnings}, \text{Notes}, \text{CatID})\)

Category\((\text{ID}, \text{Name}, \text{Description})\)

Version\((\text{ProductCode}, \text{VersionCode}, \text{Size}, \text{Color}, \text{Name}, \text{Description}, \text{Stock}, \text{Price})\)

**SOURCE 2**

Product\((\text{Code}, \text{Name}, \text{Size}, \text{Color}, \text{Description}, \text{Type}, \text{Price}, \text{Q.ty})\)

Tipe\((\text{TypeCode}, \text{Name}, \text{Description})\)

**note:** here we do not care about data types…
SOURCE 1

Product (Code, Name, Description, Warnings, Notes, CatID)
Version (ProductCode, VersionCode, Size, Color, Name, Description, Stock, Price)

SOURCE 2

Product (Code, Name, Size, Color, Description, Type, Price, Q.ty)

GLOBAL SCHEMA

CREATE VIEW GLOB-PROD AS
FROM SOURCE1.Product, SOURCE1.Version
WHERE Code = ProductCode
UNION
SELECT Code AS PCode, null as VCode, Name, Size, Color, Description, Type as CatID, Price, Q.ty AS Stock
FROM SOURCE2.Product
• Mapping quality depends on how well we have compiled the sources into the global schema through the mapping
• Whenever a source changes or a new one is added, the global schema needs to be reconsidered
Query processing in GAV: Unfolding

Select R1.A
From R1
Where R1.B in
(Select R2.B From R2)

Query over DB2:
Select R2.B into X
From R2

Query over DB1:
Select R1.A
From R1
Where R1.B in X
Query processing in GAV

**QUERY OVER THE GLOBAL SCHEMA**

```
SELECT PCode, VCode, Price, Stock
FROM GLOB-PROD
WHERE Size = "V" AND Color = "Red"
```

The transformation is easy, since the combination operator is a UNION → push selections through union!!

```
FROM SOURCE1.Product, SOURCE1.Version
WHERE Code = ProductCode AND Size = "V" AND Color = "Red"
UNION
SELECT Code, null, Price, Q.ty
FROM SOURCE2.Product
WHERE Size = "V" AND Color = "Red"
```
The steps of Data Integration: how is it done?

- **Schema Reconciliation:** schema reconciliation: mapping the data structure
- **Record Linkage:** record linkage: data matching based on the same content
- **Data Fusion:** data fusion: reconciliation of non-identical content
Be there a schema or not, we may have inconsistencies in the data

- At query processing time, when a real world object is represented by instances in different databases, they may have different values

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>AGE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>234567891</td>
<td>Ketty</td>
<td>48</td>
<td>18k</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>AGE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>234567891</td>
<td>Ketty</td>
<td>48</td>
<td>25k</td>
</tr>
</tbody>
</table>
Some data in these two tables clearly represent the same people.
Resolution function

Inconsistency may depend on different reasons:

- One (or both) of the sources are incorrect
- Each source has a correct but partial view, e.g. databases from different workplaces $\rightarrow$ the full salary is the sum of the two
- In general, the correct value may be obtained as a function of the original ones
  (maybe: $1*\text{value}_1 + 0*\text{value}_2$ as in a previous slide)
**RESOLUTION FUNCTION: EXAMPLE**

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>AGE</th>
<th>SALARY</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>JOHN</td>
<td>34</td>
<td>$30K</td>
<td>ENGINEER</td>
</tr>
<tr>
<td>234567891</td>
<td>KETTY</td>
<td>27</td>
<td>$25K</td>
<td>ENGINEER</td>
</tr>
<tr>
<td>345678912</td>
<td>WANG</td>
<td>39</td>
<td>$32K</td>
<td>MANAGER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>NAME</th>
<th>AGE</th>
<th>SALARY</th>
<th>PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>234567891</td>
<td>KETTY</td>
<td>25</td>
<td>$20K</td>
<td>1234567</td>
</tr>
<tr>
<td>345678912</td>
<td>WANG</td>
<td>38</td>
<td>$22K</td>
<td>2345678</td>
</tr>
<tr>
<td>456789123</td>
<td>MARY</td>
<td>42</td>
<td>$34K</td>
<td>3456789</td>
</tr>
</tbody>
</table>

\[ R = \text{MAX}_\text{AGE}, \text{SUM}_\text{SALARY} \ (R1 \text{ OuterJoin} \ R2) \]
The new application context
(recall)

• A (possibly large) number of data sources
  ➢ Heterogeneous data sources
• Different levels of data structure
  o Databases (relational, OO...)
  o Semi-structured data sources (XML, HTML, more markups ...)
  o Unstructured data (text, multimedia etc...)
  o Different terminologies and different operational contexts
• Different terminologies and different operational contexts
• Time-variant data (e.g. WEB)
• Mobile, transient data sources
Levels of source heterogeneity

- Schemata (already seen)
- **Data Models**
- Systems (not our concern here)
Data integration in the MULTIDATABASE

[Diagram of data integration process]

Letizia Tanca
A new element in this figure: WRAPPERS (translators)

• They convert queries into queries/commands which are understandable for the specific data source
  o they can even extend the query possibilities of a data source
• They convert query results from the source format to a format which is understandable for the application
The new application context (recall)

- A (possibly large) number of data sources
- Heterogeneous data sources
  - Different levels of data structure
    - Databases (relational, OO...)
    - Semi-structured data sources (XML, HTML, more markups ...)
    - Unstructured data (text, multimedia etc...)
    - Different terminologies and different operational contexts
- Different terminologies and different operational contexts
- Time-variant data (e.g. WEB)
- Mobile, transient data sources
SEMISTRUCTURED DATA

FOR THESE DATA THERE IS SOME FORM OF STRUCTURE, BUT IT IS NOT AS

– PRESCRIPTIVE
– REGULAR
– COMPLETE

AS IN TRADITIONAL DBMSs

EXAMPLES

– WEB DATA
– XML DATA
– BUT ALSO DATA DERIVED FROM THE INTEGRATION OF HETEROGENEOUS DATASOURCES
EXAMPLE OF SEMISTRUCTURED DATA
SEMISTRUCTURED DATA
a page produced from a database
SEMISTRUCTURED DATA MODELS

• BASED ON
  – TEXT
  – TREES
  – GRAPHS
    • LABELED NODES
    • LABELED ARCS
    • BOTH

• THEY ARE ALL DIFFERENT AND DO NOT LEND THEMSELVES TO EASY INTEGRATION
GRAPH-BASED REPRESENTATION: THE IRREGULAR DATA STRUCTURE APPEARS VERY CLEARLY

NAME

PROFESSOR

AGE

TEACHES

COURSE

YEARS

NAME

LAST NAME

KING

37

ABSTRACT OBJECTS

DATABASES

CONCRETE VALUES

NAME?

45
MEDIATORS

The term mediation includes:

• the processing needed to make the interfaces work

• the knowledge structures that drive the transformations needed to transform data to information

• any intermediate storage that is needed (Wiederhold)
Mediator-based approach

IN TSIMMIS:

• UNIQUE, GRAPH-BASED DATA MODEL
• DATA MODEL MANAGED BY THE MEDIATOR
• WRAPPERS FOR THE MODEL-TO-MODEL TRANSLATIONS
• QUERY POSED TO THE MEDIATOR
• MEDIATOR “KNOWS” THE SEMANTICS OF THE APPLICATION DOMAIN
OEM (Object Exchange Model) (TSIMMIS)

- Graph-based
- Does not represent the schema
- Directly represents data: self-descriptive

<temp-in-farenheit,int,80>
Object structure

\[(\text{Object-id}), \text{label}, \text{type}, \text{value}\]

Nested structure

\[
\text{set-of-temps, set, } \{\text{cmp}_1, \text{cmp}_2\}
\]
\[
\quad \text{cmp}_1: \text{temp-in-Fahrenheit, int, 80}
\]
\[
\quad \text{cmp}_2: \text{temp-in-Celsius, int, 20}
\]
OEM (Object Exchange Model) (TSIMMIS)
Typical complications when integrating semi- or un-structured data

• Each mediator is specialized into a certain domain (e.g. weather forecast), thus
• Each mediator must know domain metadata, which convey the data semantics
• On-line duplicate recognition and removal (no designer to solve conflicts at design time here)
• If data source changes a little, the wrapper has to be modified ➔ automatic wrapper generation
The new application context (recall)

- A (possibly large) number of data sources
- Heterogeneous data sources
- Different levels of data structure
  - Databases (relational, OO…)
  - Semi-structured data sources (XML, HTML, more markups …)
  - Unstructured data (text, multimedia etc…)
- Time-variant data (e.g. WEB)
- Different terminologies and different operational contexts
- Mobile, transient data sources
Ontologies

• A formal and shared definition of a vocabulary of terms and their inter-relationships
• Predefined relations:
  – synonymy
  – omonimy
  – hyponimy
  – etc..
• More complex, designer-defined relationships, whose semantics depends on the domain

  e.g.  enrolled(student,course)

→ an ER diagram, a class diagram, any conceptual schema is an ontology!
Definitions

• Ontology = **formal specification of a conceptualization of a shared knowledge domain.**

• An ontology is a **controlled vocabulary** that describes objects and the relationships between them in a formal way.

• It has a grammar for using the terms to express something meaningful **within a specified domain of interest.**

• The vocabulary is used to express **queries and assertions.**

• **Ontological commitments** are agreements to use the vocabulary in a consistent way for knowledge sharing and **semantic interoperability** → semantic Web.
Ontology types

• **Taxonomic ontologies**
  o Definition of concepts through terms, their hierarchical organization, and additional *(pre-defined)* relationships (synonymy, composition, …)
  o To provide a reference vocabulary

• **Descriptive ontologies**
  o Definition of concepts through data structures and their interrelationships
  o Provide information for “aligning” existing data structures or to design new, specialized ontologies *(domain ontologies)*
  o Closer to the database area techniques
horse, *Equus caballus*:
a solid-hoofed herbivorous quadruped domesticated since prehistoric times
Ontologies and integration problems

• Discovery of “equivalent” concepts (mapping)
  o What does equivalent mean?
• Formal representation of these mappings
  o How are these mappings represented?
• Reasoning on these mappings
  o How do we use the mappings within our reasoning and query-answering process?
How can ontologies support integration?

An ontology as a *schema integration support* tool

- Ontologies used to represent the semantics of schema elements (if the schema exists)
- Similarities between the source ontologies guide conflict resolution
  - At the schema level (if the schemata exist)
  - At the instance level (record linkage)

An ontology *instead of a global schema*:

- Schema-level representation only in terms of ontologies
- Ontology mapping, merging, etc. instead of schema integration
- Integrated ontology used as a schema for querying
An ontology instead of a global schema

Global Schema: Domain Ontology (at design-time)

Data-source ontologies are mapped to the Domain Ontology
The new application context (recall)

- A (possibly large) number of data sources
- Heterogeneous data sources
- Different levels of data structure
  - Databases (relational, OO...)
  - Semi-structured data sources (XML, HTML, more markups...)
  - Unstructured data (text, multimedia etc...)
- Different terminologies and different operational contexts
  - Time-variant data (e.g. WEB)
  - Mobile, transient data sources
The term mashup is widely used today:
A mashup is an application that integrates two or more mashup components at any of the application layers (data, application logic, presentation layer) possibly putting them into communication with each other.
The housingmaps.com mashup

Provides for the synchronized exploration of housing offers from craigslist.com and maps by Google Maps

Integration is the added value provided by the mashup
Other definitions

“Web-based resources consisting of dynamic networks of interacting components” (Abiteboul et Al., 2008)

“API enablers” (Ogrinz, 2009), to create an own API where there is none

“Combination of content from more than one source into an integrated experience” (Yee, 2008)
Mashup positioning in relation to other integration practices

Mashups introduce integration at the presentation layer and typically focus on non-mission-critical applications.
The problems for mashup composers

- Mashup development is non-trivial

- Luckily, mashups typically work on the “surface”
  - Reuse of existing components - neglecting the complexity hidden behind the service's external interface
  - Composition of the outputs of (much more complex) software systems

- The work of developers can be facilitated by suitable abstractions, component technologies, development paradigms and enabling tools
Types of mashup

- **Data mashups**
  - Fetch data from different resources, process them, and return an integrated result set

- **Logic mashups**
  - Integrate functionality published by logic or data components

- **User Interface (UI) mashups**
  - Combine the component’s native UIs into an integrated UI; the components’ UIs are possibly synchronized with each other

- **Hybrid mashups**
  - Span multiple layers of the application stack, bringing together different types of components inside one and a same application;

  integration happens at more than one layer
Future trends in (big) data Integration

reference:

Principles of Data Integration
by A. Doan, A. Halevy, and Z. Ives
Morgan Kaufmann
Uncertain Data

- Databases are assumed to represent *certain data*: a tuple in the database is true, any tuple NOT in the database is false (Closed World Assumption)
- Real life is not as certain! Examples:
  - Observations (e.g. sensor readings) may be *unreliable*
  - A person might be still *undecided* as to a certain choice
- Uncertain databases of various kinds attempt to model uncertain data and to answer queries in an uncertain world
Uncertainty in Data Integration

- Data itself may be uncertain (e.g. extracted from an unreliable source)
- Mappings might be approximate (e.g. created by relying on automatic ontology matching)
- Reconciliation is approximate
- Approximate mediated schema
- Imprecise queries, such as keyword-search queries, are approximate
Uncertain Databases

• Whatever the semantics of uncertainty (e.g. fuzzy, or probabilistic...) an uncertain database describes a set of possible worlds.

• **What is uncertain?**
  
  o A value?
  
  o A tuple?
  
  o A table?

• Example: Assign each tuple a probability. Then the probability of a possible world is the product of the probabilities for the tuples.

• This does not represent correlations between tuples.
Data Provenance

• Also called data lineage or data pedigree. Sometimes knowing where the data have come from and how they were produced is critical.

• Provenance of a data item records “where it comes from”:
  o Who created it
  o When it was created
  o How it was created - as a value in a database, as the result of a computation, coming from a sensor, etc...

• E.g. an information extractor might be unreliable, or one data source is more authoritative than others

• The database community models provenance in terms of how the datum was derived from the original source databases, the rest is left to the application (it is assumed to be domain dependent)
Two Viewpoints on Provenance

- **Provenance as Annotations on Data:** models provenance as a series of annotations describing how each data item was produced. These annotations can be associated with tuples or values.

- **Provenance as a Graph of Data Relationships:** models provenance as a (hyper)graph, with tuples as vertices. Each possible direct derivation of a tuple from a set of source tuples is a hyper-edge connecting the source and derived tuples.

The two views are equivalent, and we can convert from one to the other as convenient.
Uses of provenance information

- Explanations
- Scoring of sources and data quality
- Influence of sources on one another
Uncertainty, Provenance, and Cleaning

- Utilize data usage, user feedback and data quality info to assess uncertainty and automate cleaning
- Develop formalisms, data models and query primitives to capture the semantics of uncertainty propagation
Crowdsourcing

- Some checks are very simple for humans but hard for a computer
  - image contents
  - Web content extraction
  - ....
- Amazon Mechanical Turk
- Wikipedia is also a kind of crowdsourcing, collecting information from "unknown" humans
- Can provide powerful solutions to traditionally hard data integration problems (e.g. wrapping, as above, check correctness of schema mappings, etc.)
Building Large-Scale Structured Web Databases

Examples:
• Google Scholar integrates bibliographic sources to build a citation database for millions of publications.
• Physicists want to integrate the results of their experiments.
• Doctors and biologists want to integrate sources such as PubMed, Grants, and Clinical Trials to build biomedical databases.
• Companies integrate product information for searching, analysis, and marketing purposes.

New challenge besides the ones discussed today: develop methodologies for designing a fully integrated database coming from heterogeneous data sources:
• Do we clean each source first (first acting on the data), then merge them?
• Do we merge the source schemas (when they are present) and then clean the data?
• How do we manage the IDs of the identified entities, and how do we navigate through them? No more uniform query languages to access and manipulate well-identified objects!
• Incremental source update and use human feedback during the update process.
Many data integration tasks are transient:
• We may need to integrate data from multiple sources to answer a question asked once or twice. The integration needs to be done quickly and by people without technical expertise (e.g. a disaster response situation in which reports are coming from multiple data sources in the field, and the goal is to corroborate them and quickly share them with the affected public)
• Problems typical of lightweight data integration:
  o locating relevant data sources
  o assessing source quality
  o helping the user understand the semantics
  o supporting the process of integration.
• Ideally, machine learning and other techniques can be used to amplify the effects of human input, through semi-supervised learning, where small amounts of human data classification, plus large amounts of additional raw (“unlabeled”) data, are used to train the system.
• Mash-up is an example of lightweight integration
Pay-as-You-Go Data Management

- Data integration on the Web is an extreme example motivating pay-as-you-go data management.
- In contrast to the other kinds of data integration systems, pay-as-you-go systems try to avoid the need for the initial setup phase that includes creating the mediated schema and source descriptions.
- The goal is to offer a system that provides useful services on a collection of heterogeneous data with very little up-front effort.
Dataspaces

Two basic principles:

• keyword search over a collection of data coupled with effective data visualization. This can be enriched with some of the techniques for automatic schema or instance matching, automatic extraction of metadata.

• Improving the metadata in the system to the end to support and validate schema mappings, instance and reference reconciliation, or improve the results of information extraction.
Many people are trying to access the same data sources and perform similar tasks. Examples:

- When someone extracts a structured data set (excel or DB) from semistructured data it means that data is “easily structurable”, and we might also copy the names of the columns.
- When someone manually combines two data sets, we can infer that the two sources are related to each other and think about making a join between “similar” columns.

What is needed is to record actions of tons of users (problems with privacy!)

Cloud-based tools for data management might provide such services since they log the activities of many users.
Visualizing Integrated Data

• Visualize important patterns in the data instead of an infinite number of rows
• Immediately see discrepancies between the data in the sources.
• During the integration process, show a subset of the data that was not reconciled correctly
• When browsing different collections of data to be integrated, visually show the search results and evaluate their relevance to the specific integration task
• Visualization of the data provenance.
Integrating Social Media

• Social media data are often noisy (spam and low-quality data)
• Transient nature of such data and users: identifying quality data and influential users is difficult
• The data lack context: hard to interpret
• data often arrive as high-speed streams that require very fast processing
Cluster- and Cloud-Based Parallel Processing and Caching

• Most query engines, schema matchers, storage systems, query optimizers, have been developed for operation on a single server or a few servers.

• Most algorithms (schema matching, entity resolution, data cleaning, indexing, etc.) are based on assumptions of limited scale (sometimes main memory):
  o need to be tackled in a much more parallelizable and scalable way
  o need to be redesigned to exploit the power of large clusters
Conclusion

• Data Integration: a fascinating and ever-challenging problem, lending itself to a multitude of solutions

• Making sense of Big Data: still the BIG problem
Bibliography

• A. Doan, A. Halevy and Z. Ives, Principles of Data Integration, Morgan Kaufmann, 2012