Information Integration – Exercise 2 – Gabriel Glaser

Task 1: Heterogeneity

a)
- *Syntactic* (Hardware, Software, Interface): No, only one source.
- *Structural* (Data model, schematic): No, only one source
- *Semantic*:
  - Name conflict: synonym labels “year” and “YEAR”.
  - Identity: Some of the first results likely refer to the same album, but couldn’t be differentiated, because different or incomplete representations.
  - Value conflict: Track titles: *You know you’re right* vs. *You Know You’re Right* among first Nirvana results.

b) Extensions to scenario to satisfy missing heterogeneity types:
- *Syntactic*: Consider a CD data source from America which likely has a longer response time. (different hardware)
- *Structural*: Consider a CD data source which returns an XML file. (different data model)

Task 3: Distributed DBMS

a) Distribution:
- *Physical*: Servers are located in different physical/geographical locations. Therefore, they don’t share hardware components (except network).
- *Logical*: Result of application requirements, e.g., store data on different servers to handle network failure or implement caching for more speed.

b) Contained autonomy types:
- *Design*: Choose to store DB in 3NF (3$^{\text{rd}}$ normal form) or as a “BigTable” with less normalization.
- *Communication*: Decision to communicate with SQL and web form (*decision on specific query languages*). Also, they decide to allow write access to older table but not on newer table (*decision on what query capabilities to support*).

c) Execution autonomy is the last type of autonomy. For instance, a system could block queries from some countries.

d) Syntactic heterogeneity:

- *Hardware*: This type of heterogeneity can be noticed when some datasources process a query faster than others (different CPU/bandwidth).
- *Software*: Different datasources could be stored using different operation systems, e.g., need to use different file separators ("/" for Linux, "\" for Windows).
- *Interface*: For instance, access to datasource, i.e., pass a query, via URL containing various parameters vs. access via SQL query given to an URL as string.

e) a) Value vs. relation heterogeneity:

Employee(ID, Firstname, Lastname, isManager, department, gender)

vs.

MaleEmployee(ID, Firstname, Lastname, isManager, department)
FemaleEmployee(ID, Firstname, Lastname, isManager, department)

b) Value vs. attribute heterogeneity:

Employee(ID, Firstname, Lastname, isManager, department, gender)

vs.

Employee(ID, Firstname, Lastname, isManager, department, isFemale, isMale)

c) Different labels of attributes (in comparison with the original):

Employee(ID, Firstname, Lastname, isManager, *workArea*)

d) Normalized vs non-normalized schema (original is normalized):

Employee(ID, Firstname, Lastname, department)
Managers(MID, ID → Employee)

Task 4: Integrating publication data

a)
- **Syntactic**: Likely a hardware heterogeneity, because Amazon is a much bigger cooperation than ACM and DBLP. For instance, Amazon is likely more reliable.
- **Structural**: Different data model between ACM and DBLP (XML vs. BibTex).
- **Semantic**: Value conflict of attribute *publisher* between ACM and DBLP results ("ideaGroup" vs. "ACM").

b) **DBLP**:

Publications(P-ID, *publicationType*, *key*, *mdate*, title, publisher, year, isbn, url)
Editors(E-ID, name, P-ID → Publications)

ACM:

Publications(P-ID, *publicationType*, label, author, title, journal, issue-date, volume,
number, month, year, issn, pages, numpages, url, doi, acmid,
publisher, address)

Amazon:

Items(Item-ID, name, Prize-ID → Prizes, description, isbn-10, isbn-13, publisher,
publishDate, language, size, weight, numberOfPages, review,
itemType)
Authors(Author-ID, firstname, lastname, Item-ID → Items)
Prizes(Prize-ID, Item-ID → Items, description [*e.g., new or used*], prizeInDollar)

c) Attribute matches from target schema attributes to matched attributes:

- **DBLP**: Match title → title, publisher → publisher, pubType → publicationType, pubDate → mdate, editon → *not given*

  Extract authors from *Editor*-table, then match firstname → firstname, lastname → lastname, but authorType → *not given.*

- **ACM**: Match title → title, publisher → publisher, pubType → publicationType, pubDate → *only month, year given*, editon → volume

  Extract single names by splitting on appropriately, then match firstname → *only first letter given*, lastname → lastname, authorType → *not given.*

- **Amazon**: Match title → name, publisher → publisher, pubType → itemType, pubDate → publishDate, editon → *not given*

  Extract authors from *Authors*-table of Amazon, then match firstname → firstname, lastname → *lastname*, authorType → *not given.*