Shuffling Data Around...

An introduction to the keywords in Data Integration, Exchange and Sharing

Dr. Anastasios Kementsietsidis

Special thanks to Prof. Renée J. Miller

The Cause and Effect Principle

Cause:

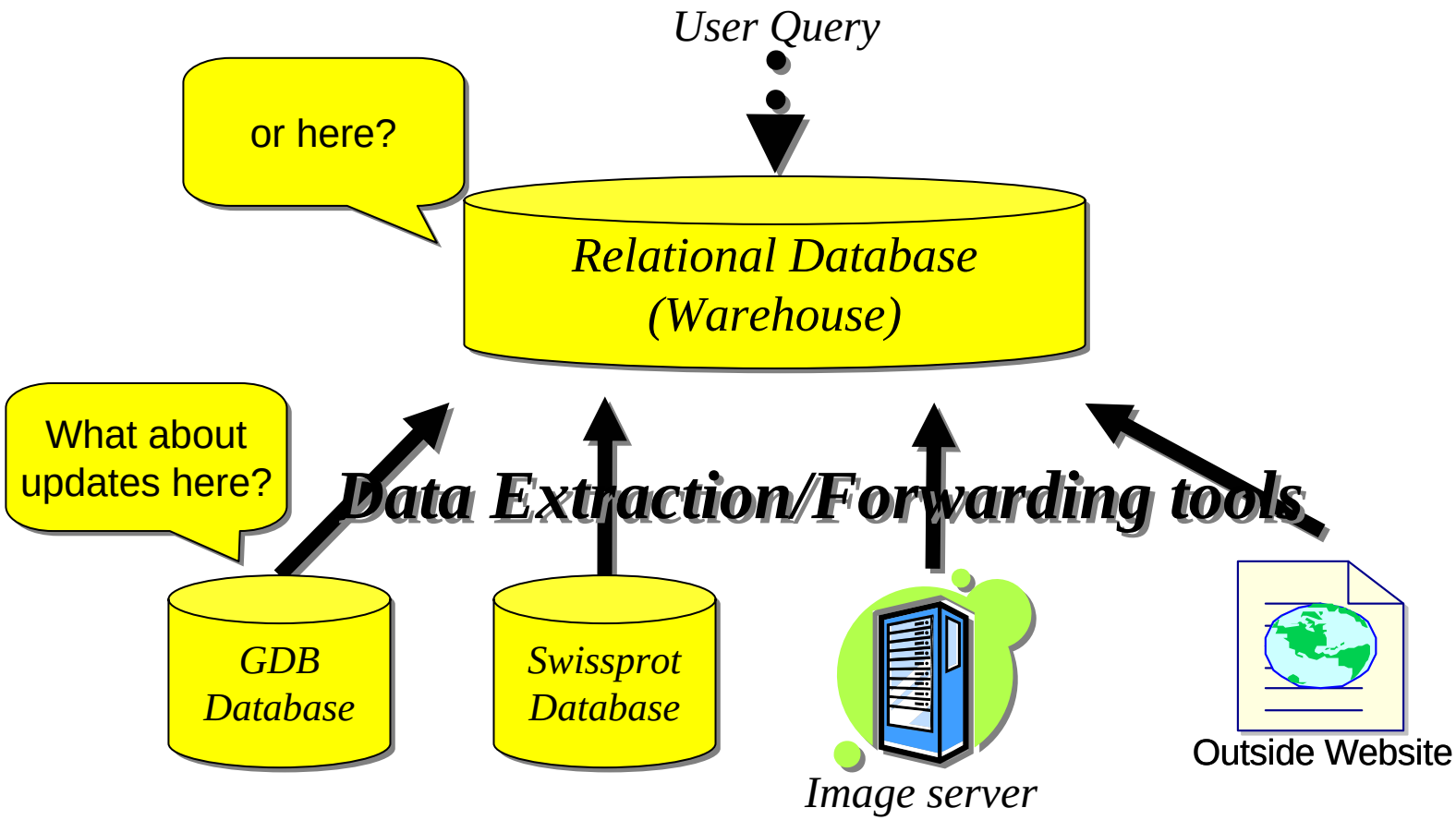
Data sources are **autonomous**, **heterogeneous**

- Different data models, types and schemas
- Different vocabularies (in data and schemas)
- Different requirements for what/how data is shared

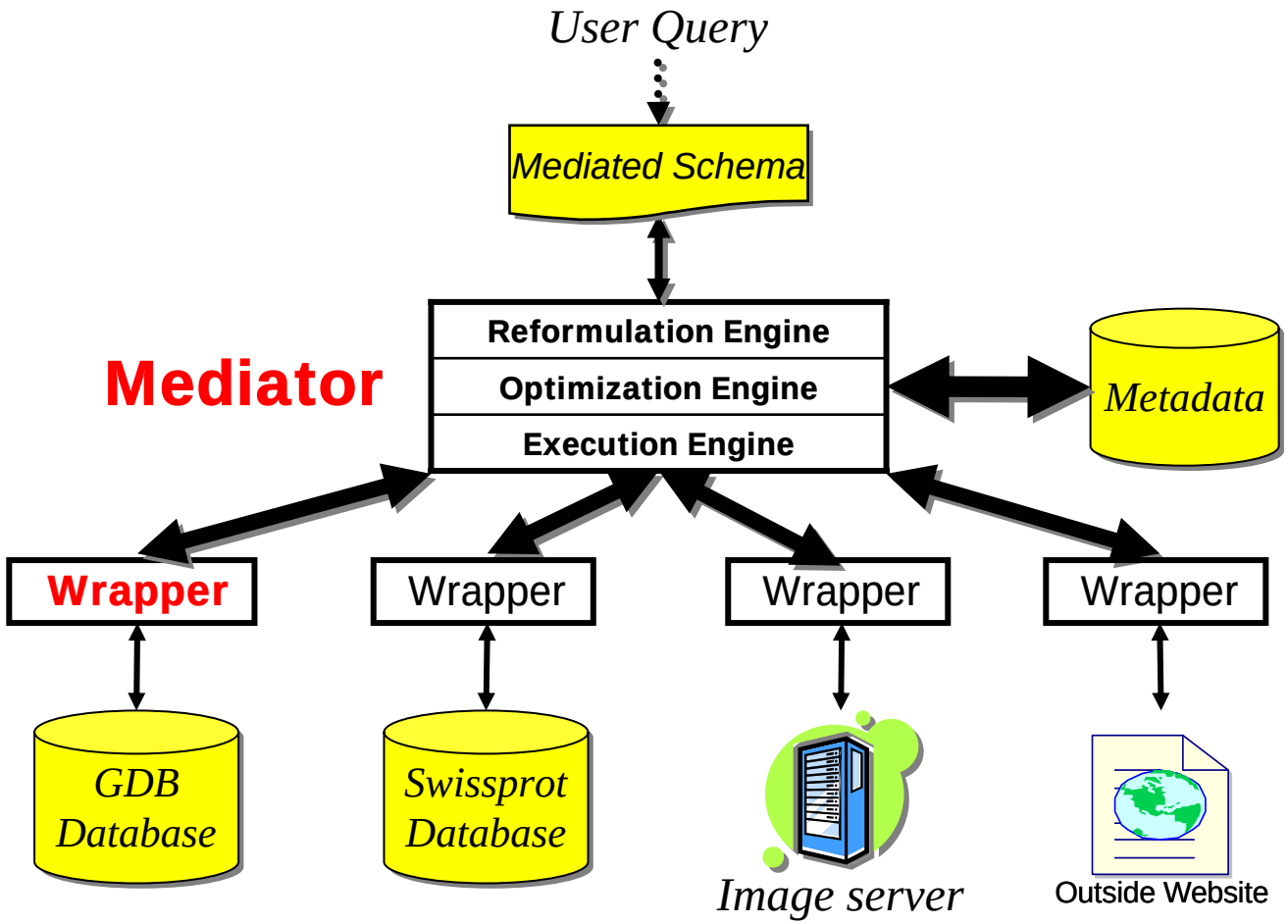
Effect:

- **Integration:** Provide *uniform* access to heterogeneous data
- **Exchange**: Move data between heterogeneous sources
- **Sharing:** Provide *non-uniform* access to data through each source's schema and vocabulary

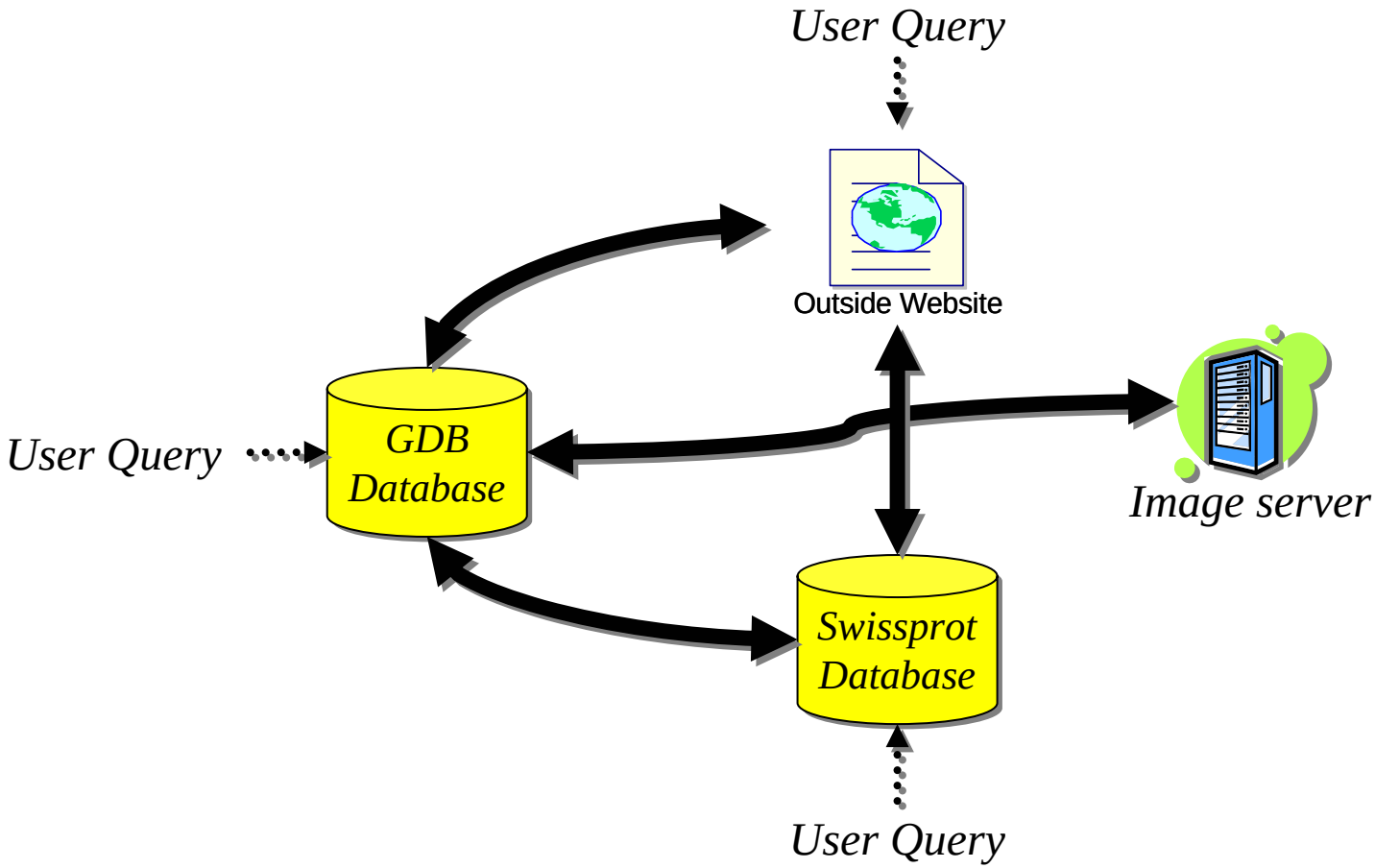
Data Warehousing Architecture



Virtual Integration Architecture



Peer-to-Peer Architecture



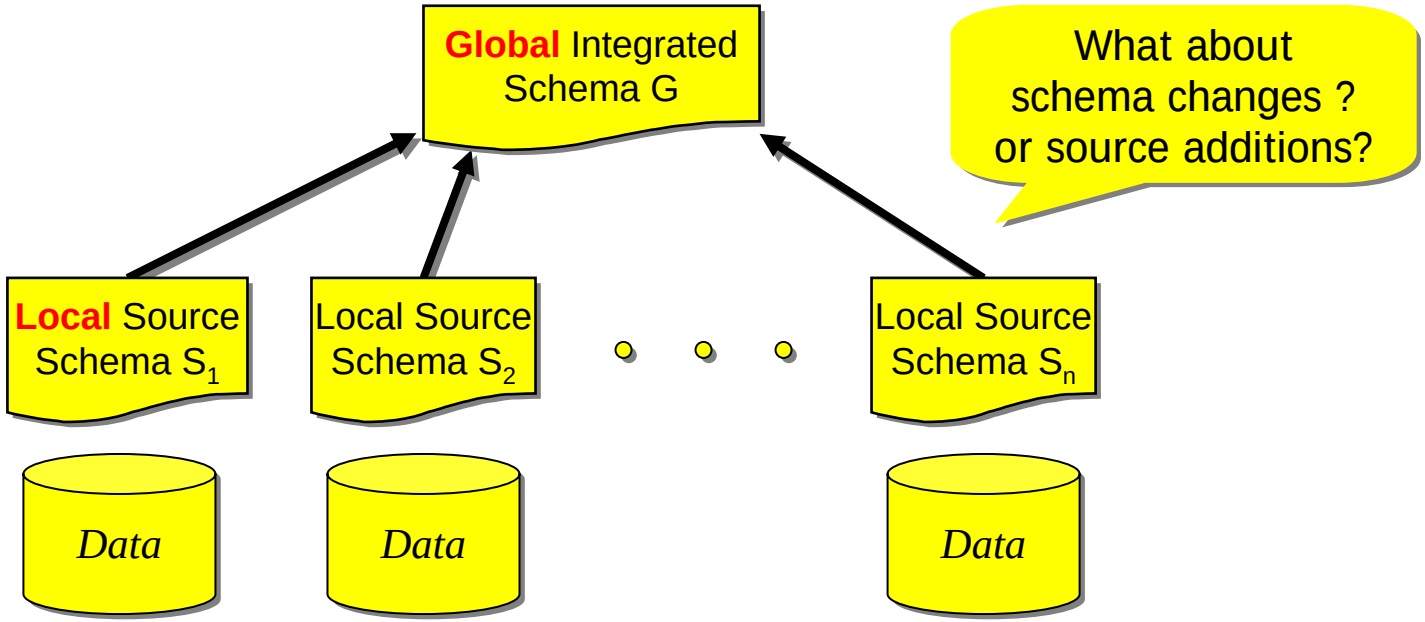
What are the Metadata?

- **Schemas** (models of data)
  - Structured or Semi-structured
    - e.g., relational, object-oriented, XML data, ...
  - Talk will not cover unstructured data
    - e.g., documents, images, audio files,...
  - Data(base) is an instance of a schema
- **Mappings**
  - Model relationship between schemas or data
    - Schema mapping (e.g., views)
    - Data mapping (e.g., aliases)
  - Requirements for mapping specifications

Metadata Lifecycle

- Creation
  - Automatic discovery or creation
  - Design tools facilitating creation
- Maintenance
  - Maintain (integrated) schemas as sources change
  - Maintain mappings as schemas change
- Use
  - Query answering
  - Data exchange (materialization), updates, etc...

Schema Integration

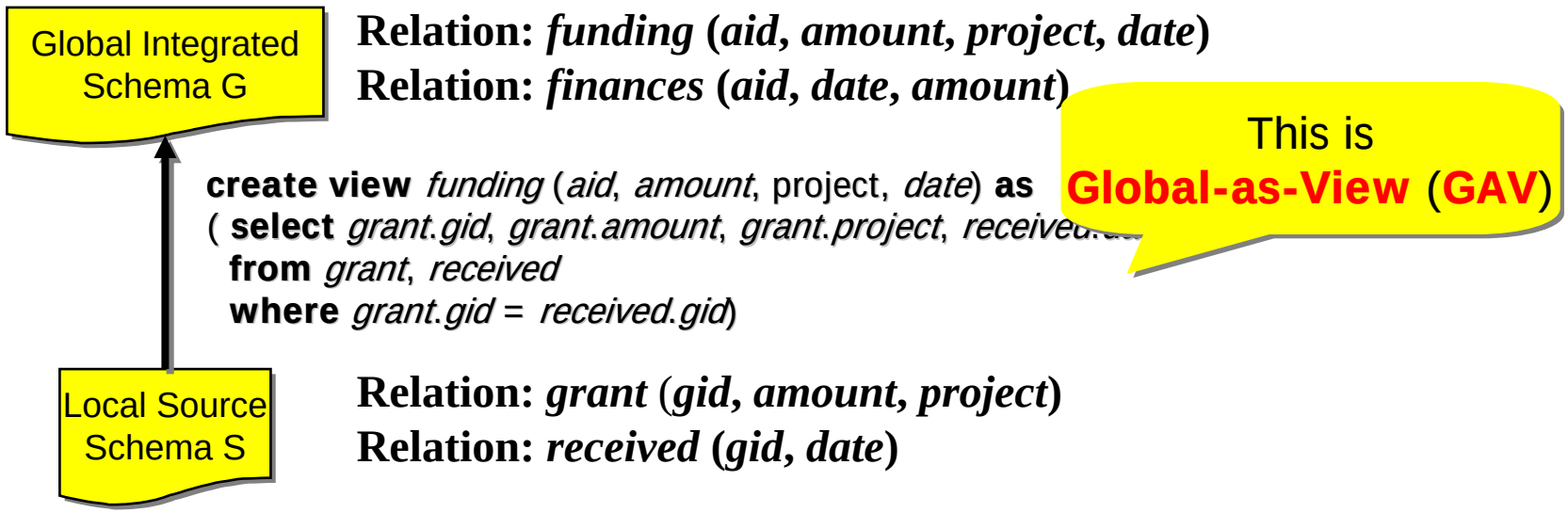


Schema Design Problem:
Create **global** integrated schema G (and mappings) for a set of independently designed **local** schemas S_i, $1 \leq i \leq n$

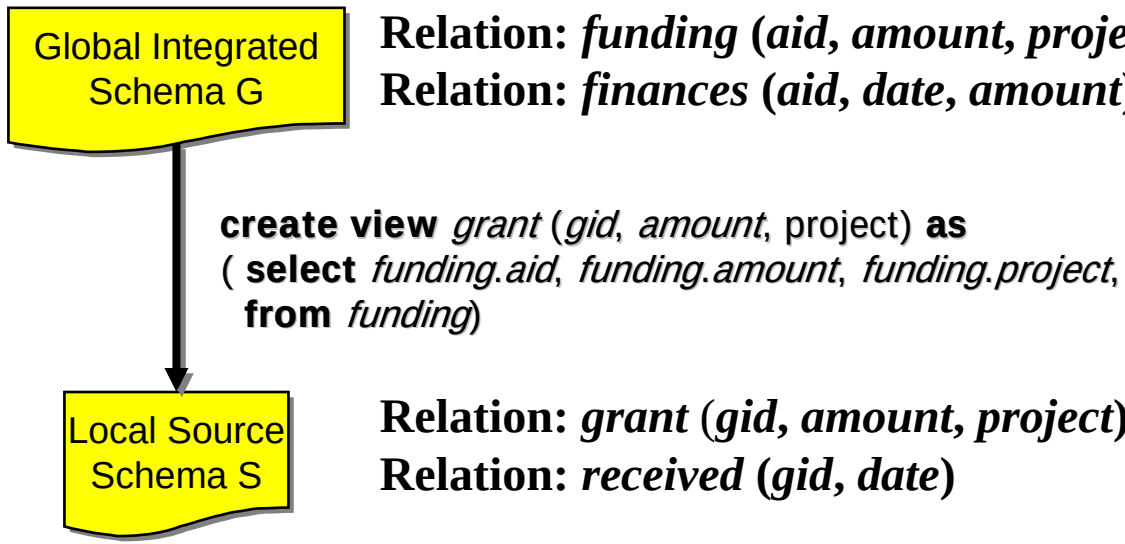
Mappings (a.k.a. views)

A view is just a query... works like a function....

- It accepts as input the local source instance(s)
- It outputs an instance of the global (target) schema



This is Local-as-View (LAV)

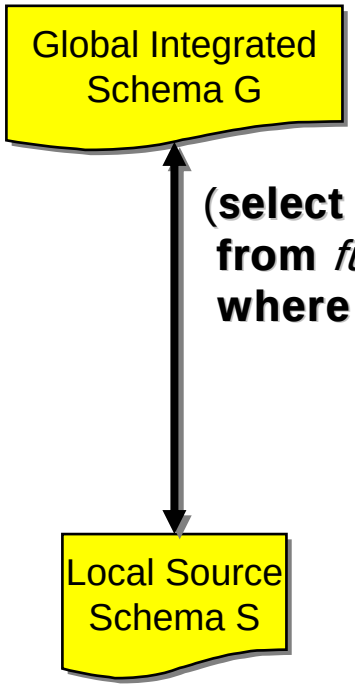


GAV vs. LAV (in “plain” English)

- GAV:
  - Gives direct information about which data satisfy the elements of the global schema
  - Not easily extendible on source schema changes or source additions
  - Query answering is “easy”...
- LAV:
  - Does **not** give direct information about which data satisfy the global schema
  - Easily extendible on source schema changes or source additions
  - Query answering is “hard”...

But There Is More...

Global-and-Local-as-View (GLAV)



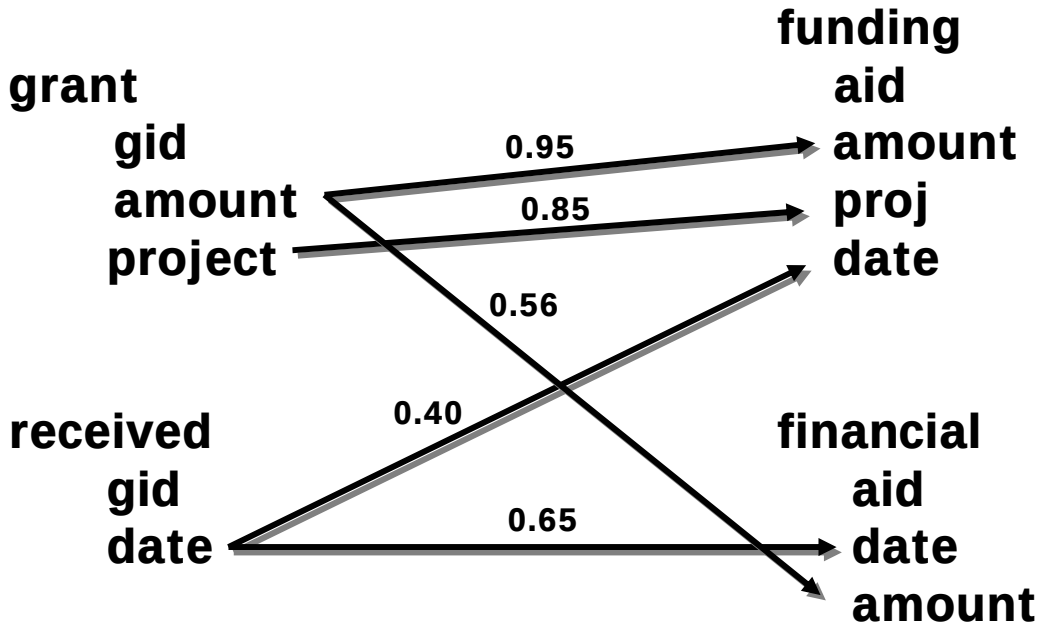
Relation: ***funding*** **(*aid, amount, project, date*)**
Relation: ***finances*** **(*aid, date, amount*)**

(**select** *funding.aid*, *funding.amount*, *funding.project*, *funding.date*
from *funding*, *finances*
where *funding.aid* = *finances.aid*) ⊇
(**select** *grant.gid*, *grant.amount*, *grant.project*, *received.date*
from *grant*, *received*
where *grant.gid* = *received.gid*)

Relation: ***grant*** **(*gid, amount, project*)**
Relation: ***received*** **(*gid, date*)**

Creating Mappings

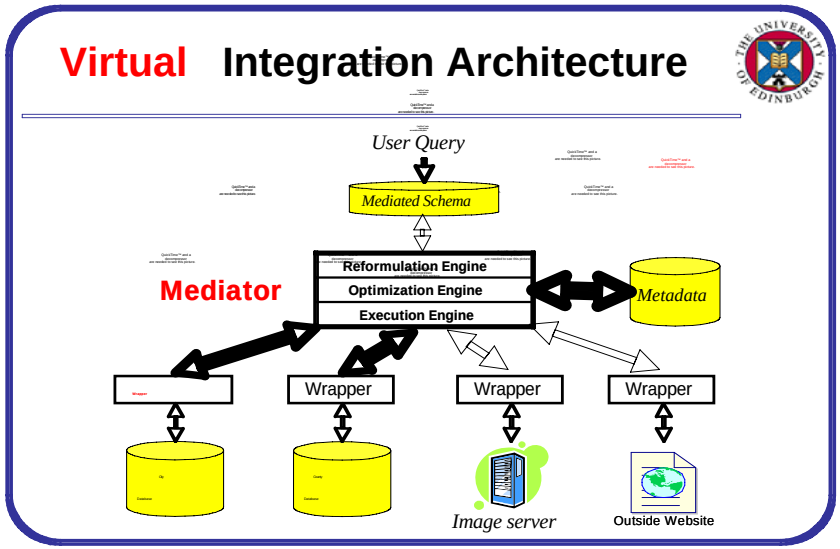
… with the help of **Schema Matching**



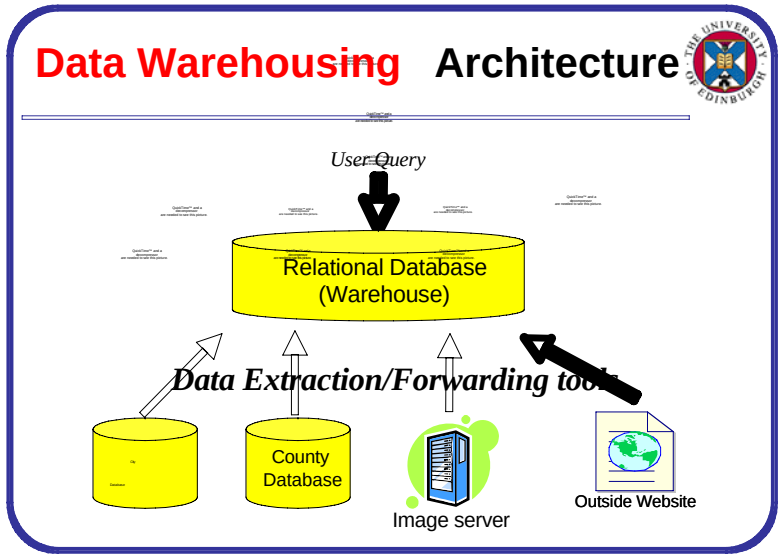
It uses schema-level and/or instance-level information

… And Using Them

In Data Integration



In Data Exchange



Data Integration vs. Exchange

- Data Integration
  - Global schema is a reconciled virtual view of heterogeneous sources
  - Uses GAV or LAV mappings
  - No **constraints** in the global schema are considered
  - Query is answered using source data; integration is **virtual**
  - Answer is set of tuples in query result on ALL possible target instances: **certain answers**

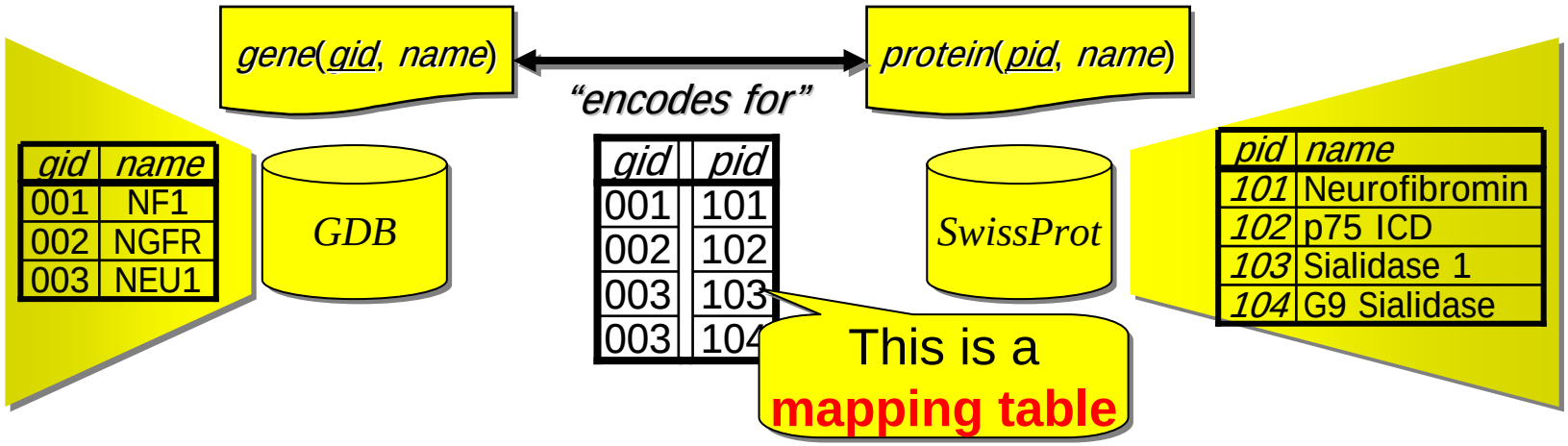
- Data Exchange
  - Global schema is an independently created local source schema
  - Uses GLAV mappings
  - Considers the presence of constraints
  - Query is answered using **ONE materialized** target
  - Can single target give same information as source(s)?

Something Slightly Different…

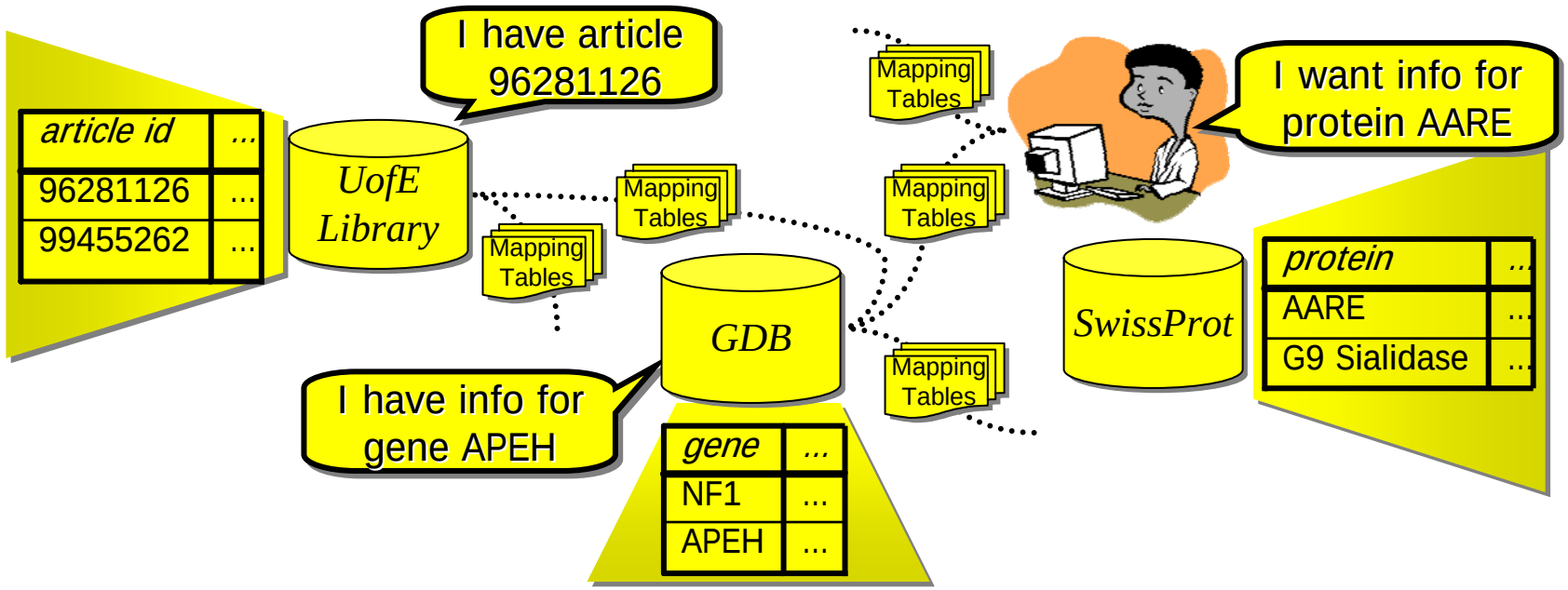
Data Mappings and **Data Sharing**

Useful in environments where:

- Sources are unwilling to share schemas
- The schema of one source cannot be expressed as a view of another
- There is a need to map (data) vocabularies

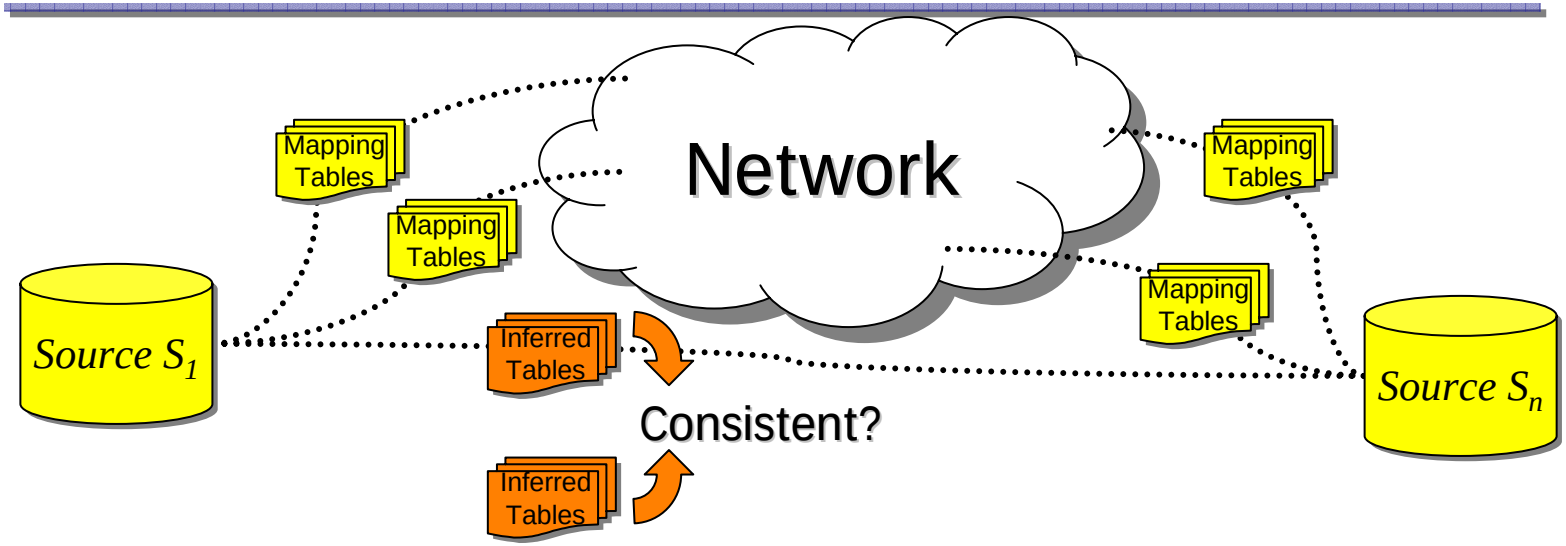


Data Sharing Architecture



- Establish mapping tables between the vocabularies of different sources.
- Use these tables to translate query requests between the sources

Management of Tables



The Consistency and Inference problems, the main vehicles for managing mapping tables. Solving these problems allows us to:

- Infer new mapping tables from existing ones
- Augment existing mapping tables with new associations
- Validate mapping tables

Closing Remarks...

... and things to remember (other than the **keywords**):

- Integration, one of the oldest problems in database research. Research is still going strong in this area
- Exchange, an interesting and practical problem (e.g. B2B apps)
- Sharing, the latest twist in the integration problem, also of practical importance (e.g. in P2P apps)

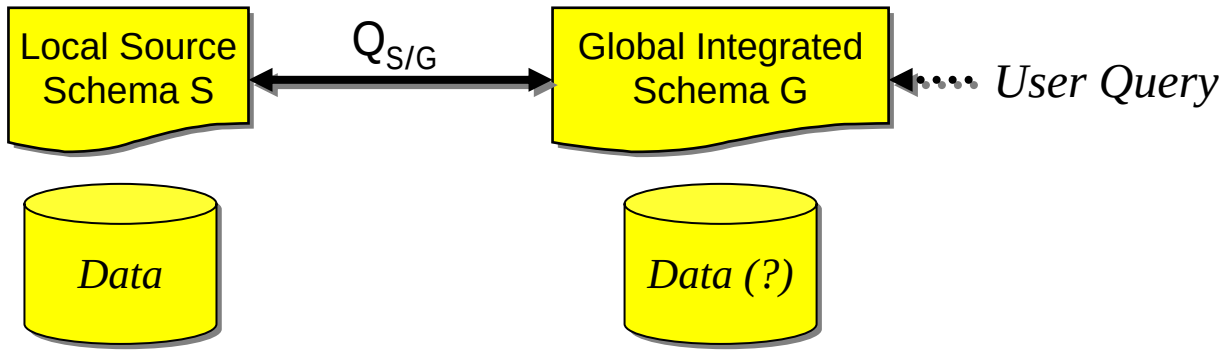
Disclaimer:

This talk provides only a glimpse of the research issues in these areas.

Note: If you find any of these interesting, **TALK** to us...
We are in Appleton Tower, 2nd Floor

Questions!?

GAV vs. LAV

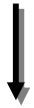


- GAV: $Q_S(S) \subseteq R_G$,
  where R_G is a relation in G, Q_S is a query on S
- LAV: $R_S \subseteq Q_G(G)$,
  where R_s is a relation in S, Q_G is a query on G

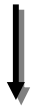
Query Answering

GAV uses **Query/View Unfolding**

select *project*
from *funding*
where *amount* > *$45.000*



create view *funding* (*aid*, *amount*, project, *date*) **as**
(**select** *grant.gid*, *grant.amount*, *grant.project*, *received.date*
from *grant*, *received*
where *grant.gid* = *received.gid*)



select *project*
from *grant, received*
where *grant.gid = received.gid* **AND**
grant.amount > $45.000

What about LAV?
It uses a method called
Query Rewriting
(not presented here)