The University of Queensland
AUSTRALIA
CREATE CHANGE

Advanced Database Systems (INFS3200)

Lecture 9: Data Integration and Linkage

Lecturer: A/Prof Sen Wang

School of Electrical Engineering and Computer Science (EECS)

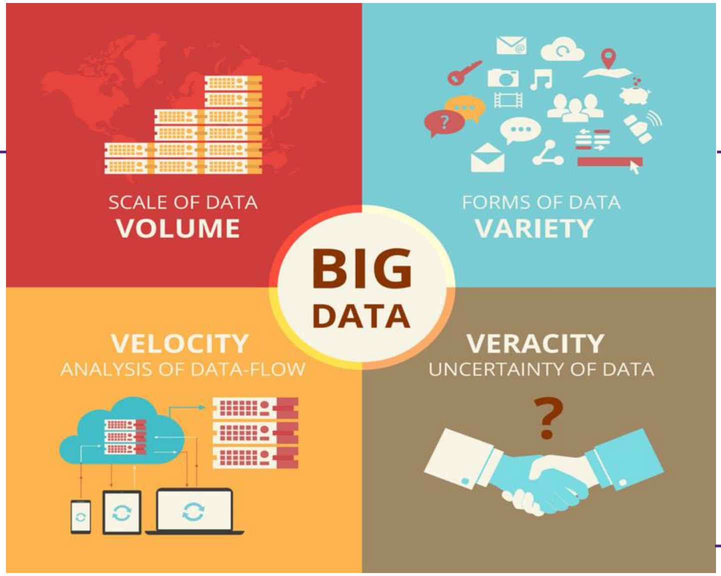
Faculty of Engineering, Architecture and Information Technology

The University of Queensland

Where Are We NOW?

OLTP

Distributed DB
- Design
- Query Processing
- Transaction Management



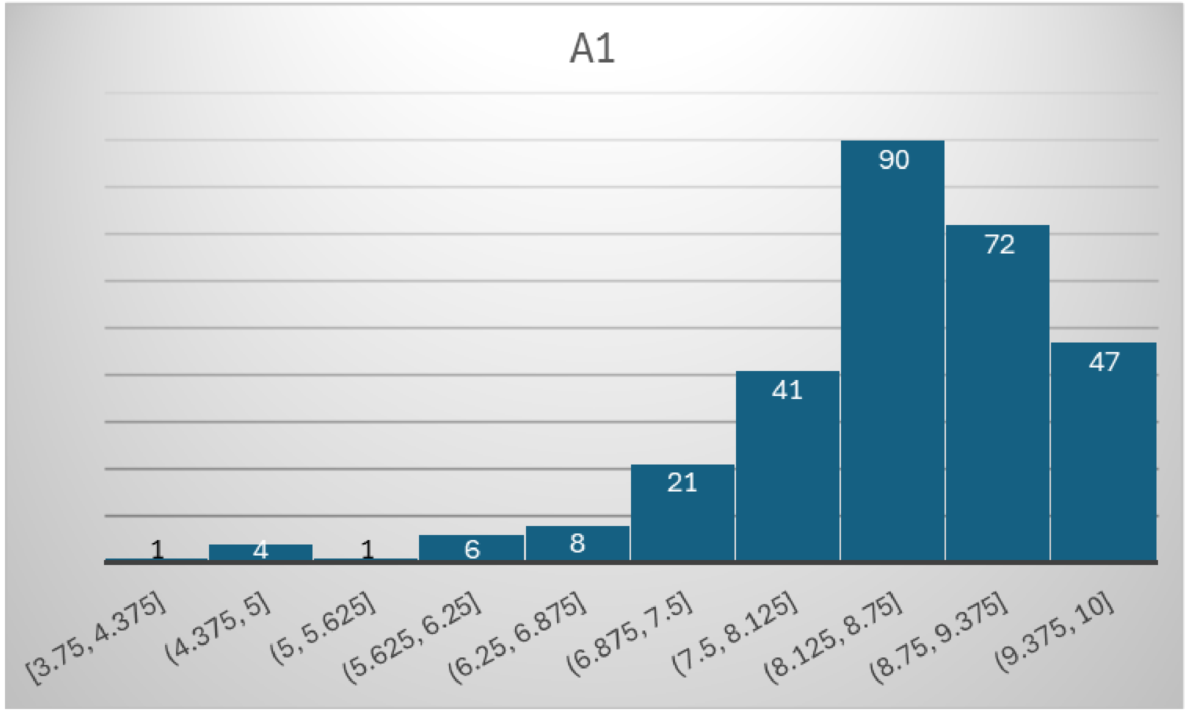
OLAP

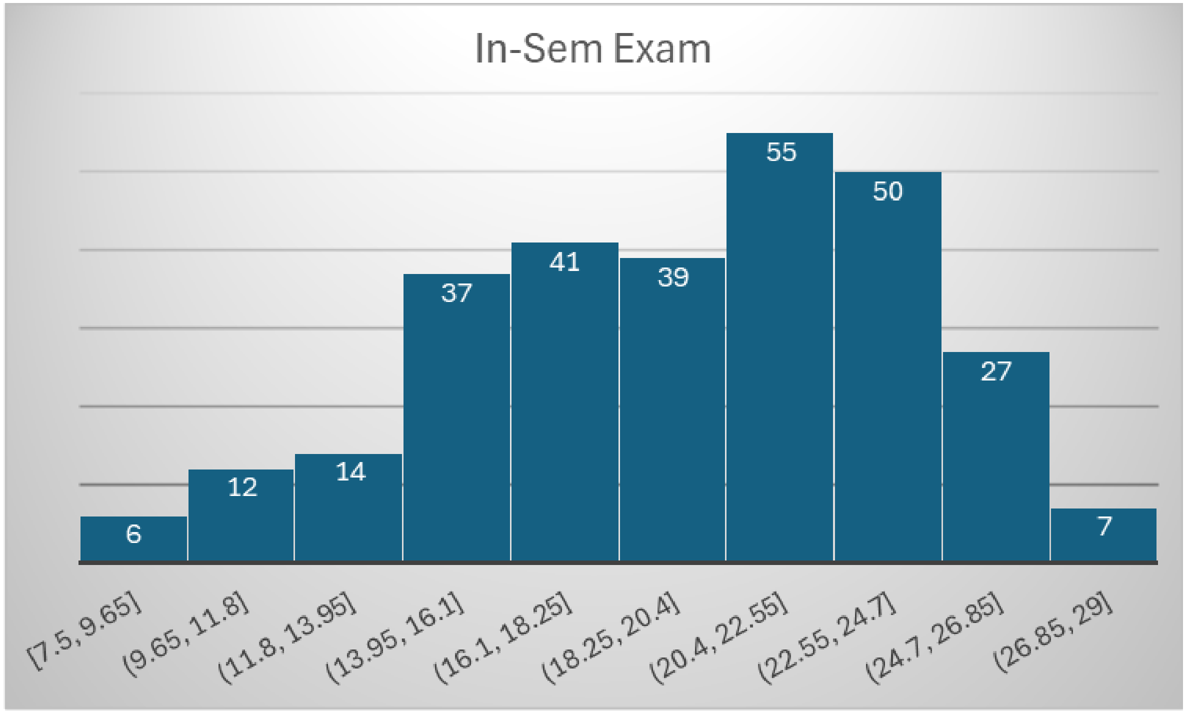
Data Warehouse
- Design
- Implementation

Data Preprocessing
- Integration & Linkage
- Data Privacy & Quality Management

Advanced Topics
- Column Store
- Data Lake
- Big Data Applications

Statistics of A1 and In-Sem Exam





	Median	Std
A1	8.5	1.01
In-Sem Exam	20	4.47

Topics

I. Data Integration

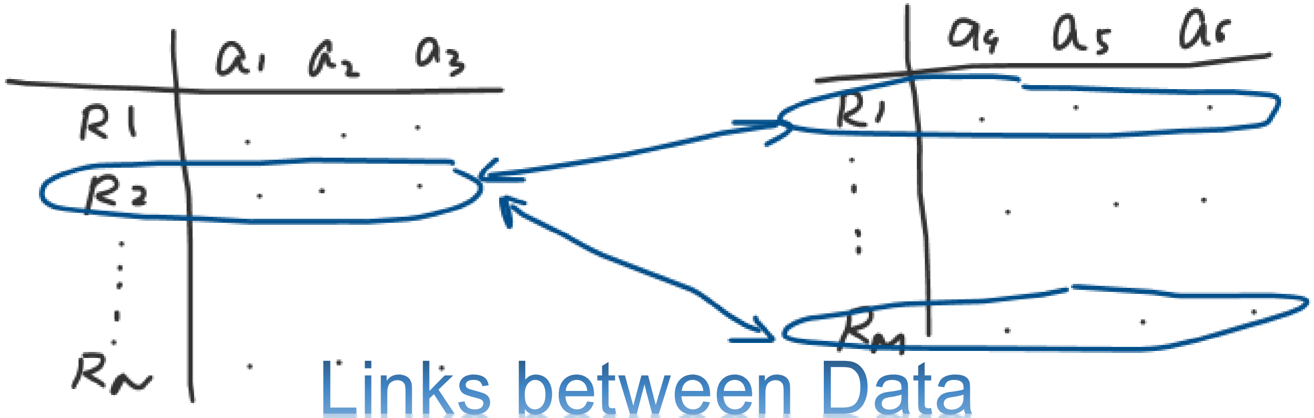
Unified Data Env

? schema matching

structure

semantics

II. Data Linkage



Links between Data

? Fuzzy Matching

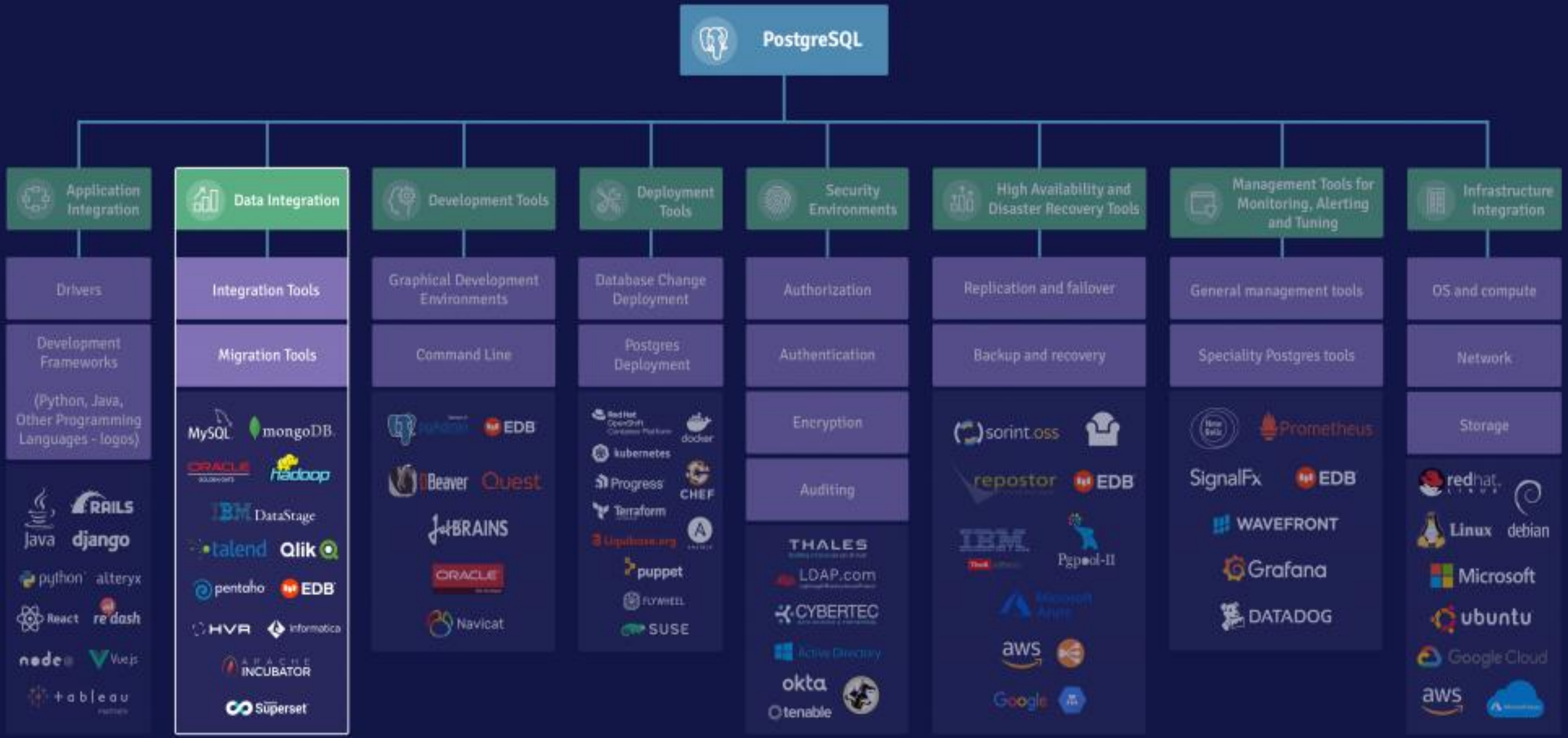
Outline

DB Integration:

- Why DB Integration and Related Issues
- Global Information Systems
  - Federated Databases
  - Multidatabases
- Mediator-Wrapper Architecture
- Challenges in DB Integration
- Three Steps for DB Integration
- View-based DB Integration
  - Global-as-view vs Local-as-view
- Limitations of Views

Data Linkage:

- Why Data Linkage & What is NOT Data Linkage
- Data Linkage Applications
- Linking with Different Granularity
  - Field Matching
    - Edit Distance
    - q-Gram and Jaccard Coefficient
    - TF/IDF and Cosine Similarity
    - Numeric Simililarity
  - Record Matching
    - Weighted Sum & Rule-based Approaches
  - Group Matching
  - Canopy Cluster


PostgreSQL
Application Integration
Drivers
Development Frameworks
(Python, Java, Other Programming Languages - logos)
Java
Rails
django
python
alteryx
React
redash
node
Vue.js
tableau
Data Integration
Integration Tools
Migration Tools
MySQL
mongoDB
ORACLE
hadoop
IBM DataStage
talend
Qlik
pentaho
EDB
HVR
informatica
APACHE INCUBATOR
Superset
Development Tools
Graphical Development Environments
Command Line
EDB
DBeaver
Quest
JetBRAINS
ORACLE
Navicat
Deployment Tools
Database Change Deployment
Postgres Deployment
docker
kubernetes
Progress
CHEF
Terraform
puppet
FLYWHEEL
SUSE
Security Environments
Authorization
Authentication
Encryption
Auditing
THALES
LDAP.com
CYBERTEC
okta
tenable
High Availability and Disaster Recovery Tools
Replication and failover
Backup and recovery
sorint.oss
repostor
EDB
IBM
Pgpool-II
aws
Google
Management Tools for Monitoring, Alerting and Tuning
General management tools
Speciality Postgres tools
Prometheus
SignalFx
EDB
WAVEFRONT
Grafana
DATADOG
Infrastructure Integration
OS and compute
Network
Storage
redhat
Linux
debian
Microsoft
ubuntu
Google Cloud
aws

Why Database Integration?

Scenarios:

- Want to combine databases when two companies merge
  - Single Source of Truth (SSOT)
- Want to enhance information using data from different sources
  - Deeper insights or more comprehensive understanding of data
- Want to access data in legacy systems
  - Valuable legacy – stop repeating mistakes

Why Database Integration?

Examples

- Telstra claims to have over 1,000 information systems
- Health Connect is an Australian Government initiative intended to integrate hospitals, medical practitioners, pathology laboratories, the Health Insurance Commission, health funds, and more (Australian MHR (My Health Record) Database)
- Supply chain management integrates retailers, wholesalers, manufacturers and suppliers
- E-commerce exchanges allow electronically mediated interaction among many thousands of businesses

Related Issues

Combine data from different structured or unstructured data sources…

Noisy Data?

- Data Cleaning, remove noise in original data
- Remove noise in integrated data
  - Inconsistency and redundancy

Duplicates? Wrong Matches?

- Data Linkage
  - Identifying records referring to the same real-world entity
  - Computing data similarity
  - Applicability of different similarities

Related Issues

Combine data from different structured or unstructured data sources…

Data Breaches?

- Data Privacy: Share data with the assurance that "private" information cannot be derived.

Data Quality issues:

- Dealing with constraints, augmentation, provenance

Global Information Systems

Three dimensions

- Distribution
- Heterogeneity
- Autonomy

Two approaches

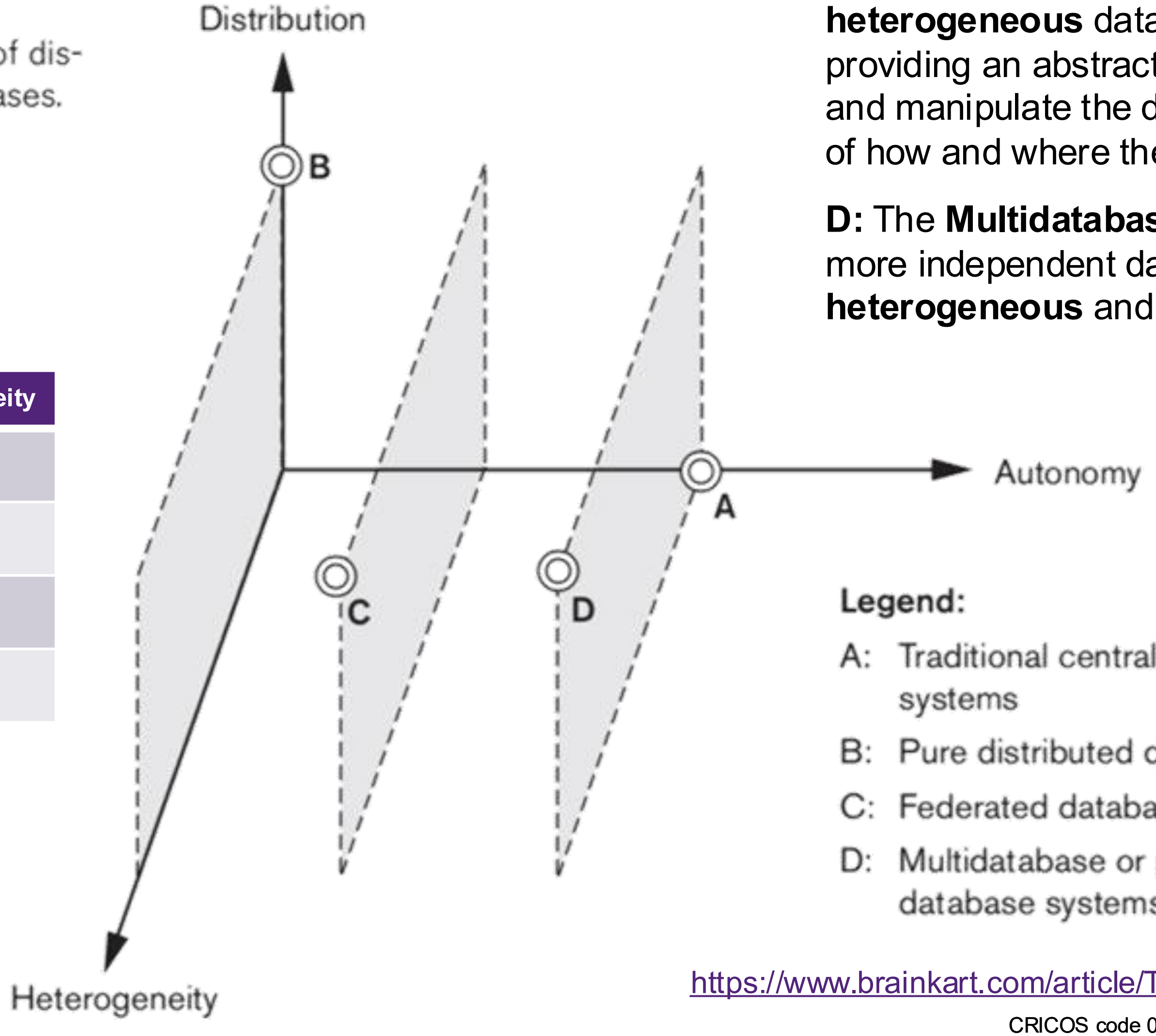
- Top-down (first-principles design)
- Bottom-up (integrating existing systems)

What we discuss in INFS3200

- Distributed database systems (Top-down)
- Data warehousing systems (Bottom-up)

Global Information Systems

Figure 25.2
Classification of distributed databases.



	Distribution	Autonomy	Heterogeneity
A	None	High	None
B	High	None	None
C	High	Low	High
D	High	High	High

C: **Federated database** is a collection of **cooperating** but autonomous constituent databases that present themselves to users as a single database. The system integrates **heterogeneous** databases into a single federated schema, providing an abstraction layer that enables users to access and manipulate the data without needing to know the details of how and where the data is stored.

D: The **Multidatabase** is a system that manages two or more independent databases, which may be **heterogeneous** and are designed to be **autonomous.**

Legend:

A: Traditional centralized database systems

B: Pure distributed database systems

C: Federated database systems

D: Multidatabase or peer to peer database systems

https://www.brainkart.com/article/Types-of-Distributed-Database-Systems_11591/

Global Information Systems

Federated databases (FDB)

- Semi-autonomous database systems, a global view is provided, loosely or tightly coupled
- Example: **Healthcare Systems Integration**
  - Multiple hospitals with different internal databases (Oracle, PostgreSQL, etc.) federated into a single view for centralised patient care analytics.
  - A doctor queries a patient's full history across hospitals/GPs:
    - SELECT * FROM GlobalPatientRecords WHERE patient_id = 'A123';

Multi-databases (MDB)

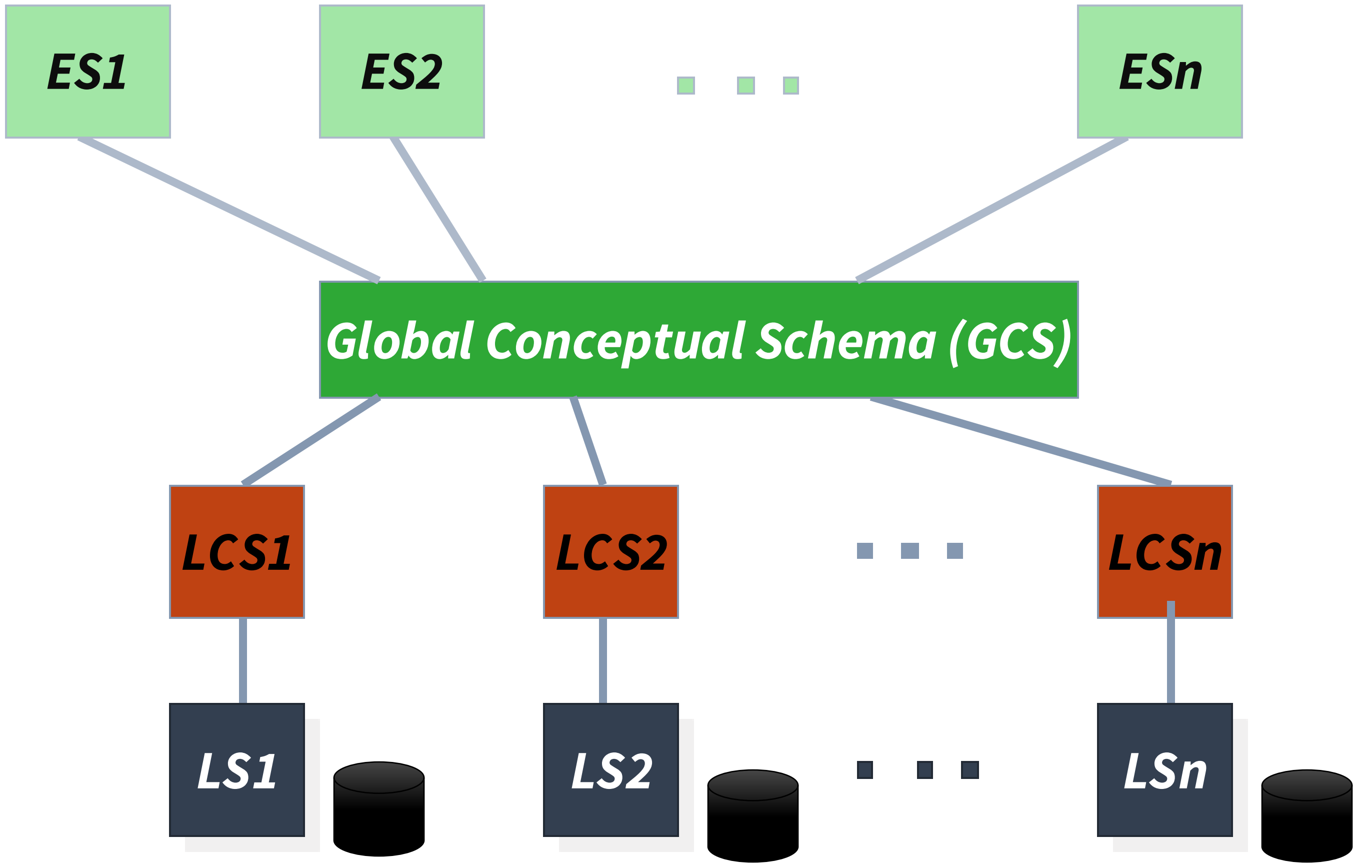
- Autonomous database systems, limited global view is provided
- Example: **Global Video Streaming System**
  - Multiple independent databases for each country or region (due to copyright, legal and regulatory differences). Data is accessed separately when needed.
  - An analyst writes separate queries for different databases:

```
-- Query from US branch DB
SELECT * FROM us_branch.accounts WHERE age > 30;
-- Query from AU branch DB
SELECT * FROM au_branch.accounts WHERE age > 30;
```

	Global Interface	Local node types	Full global DB function?	Integration method
DDB	*Internal DBMS Functions*	*DBs*	*Yes*	*Global Schema*
MDB/ FDB	*DBMS User Interface*	*DBs*	*Partial*	*Partial Global Schema*

Federated Databases

A federated database system connects all local databases



ES: External Schema

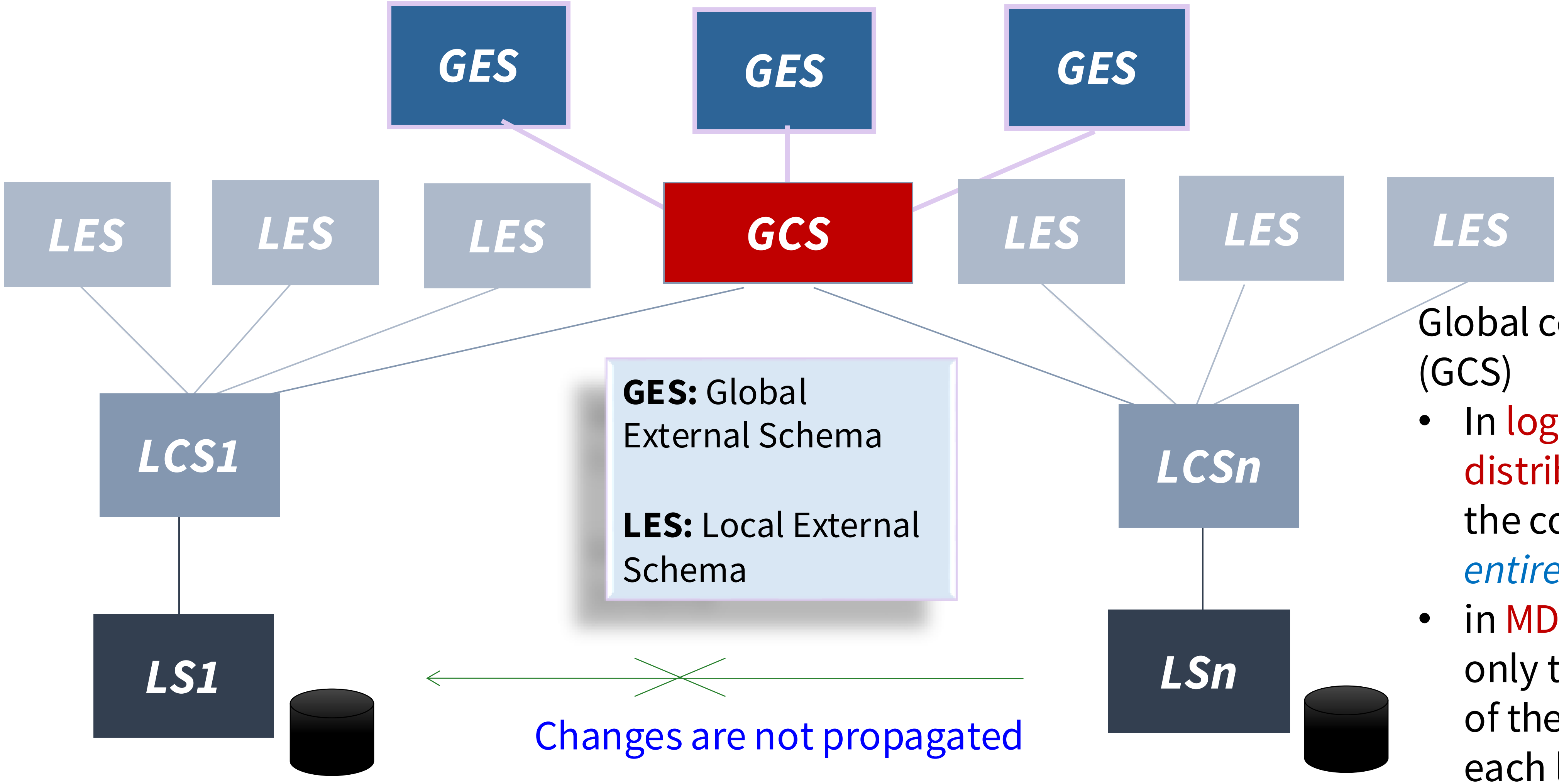
GCS: Global Conceptual Scheme

LCS: Local Conceptual Schema

LS: Local Schema

Multi-Databases (MDBSs)

databases are entirely isolated and not directly connected



GES: Global External Schema

LES: Local External Schema

Global conceptual schema (GCS)

- In logically integrated distributed DBMSs: defines the conceptual view of the *entire* database
- in MDBSs, it represents only the collection of *some* of the local databases that each local DBMS wants to share.

Recall in Distributed Databases…

In distributed systems including DDB, we assume that the entire project is under control of a single organization

In DDB, choices as to **fragmentation**, **replication** and tasks for **sub-transactions** are made on engineering considerations, assuming information availability

- for allocation of fragments and replicas to sites
- for system catalog supporting query optimization
- for resource locking and commit protocols…

So, building a DDB is a “white box” engineering problem.

Now, Consider a Different Scenario

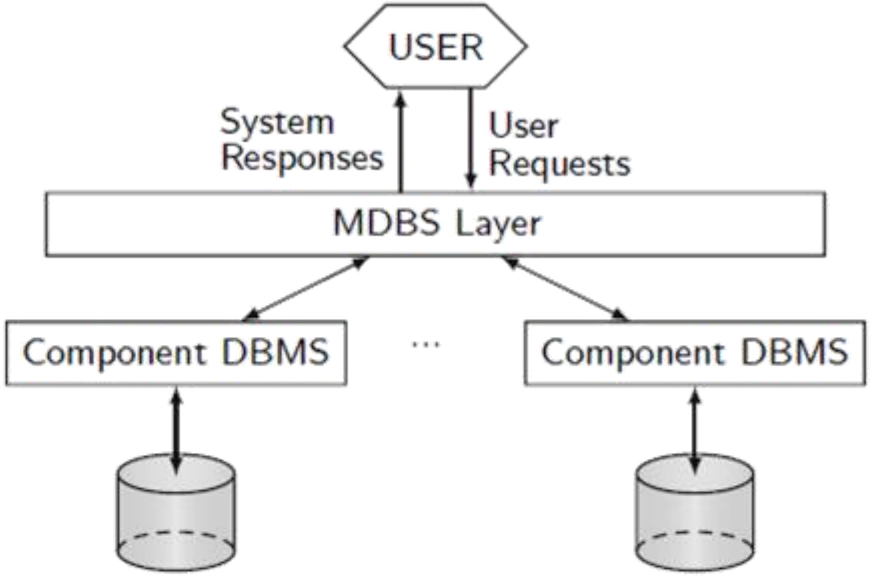
Now, we assume different parts of the system are controlled by different organizations or organizational units

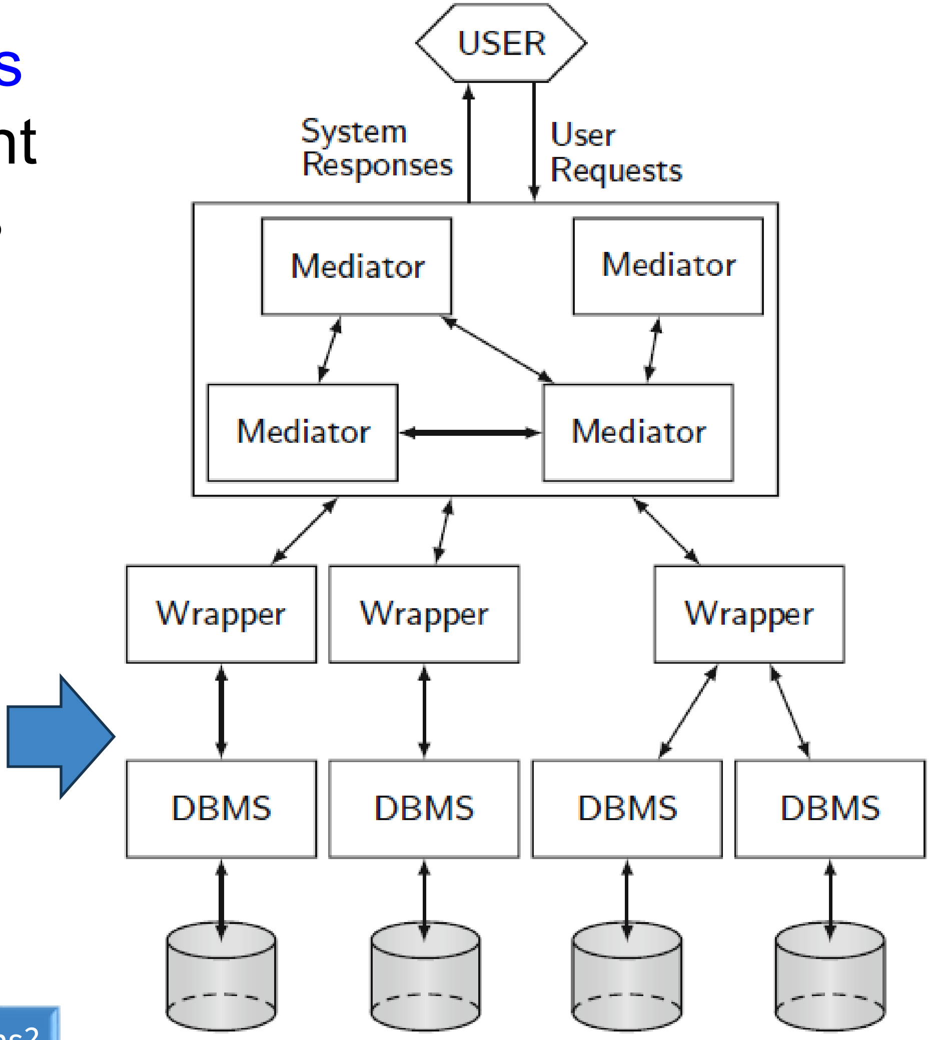
- Technological **boundaries**, organizational **boundaries** and political **boundaries**
- These organizations/units are taken to be autonomous
  - No one can tell another what to do
  - No organization/unit is required to expose the internals of their systems, including their system catalogs

We now have a "black box", or possibly "grey box" (e.g., some participants may reveal some information) problem.

Data Integration Example: Mediator- Wrapper Architecture

- Integrating information over different data sources (e.g. Websites), which may have radically different computing platforms, data formats and structures

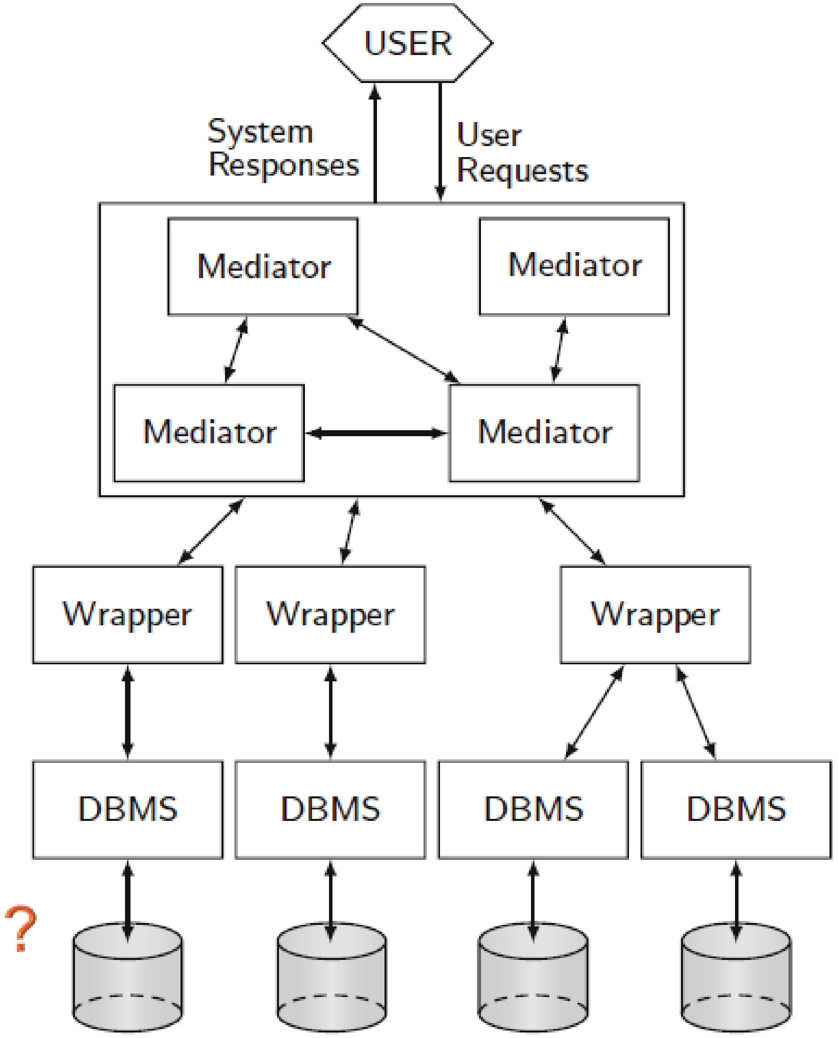




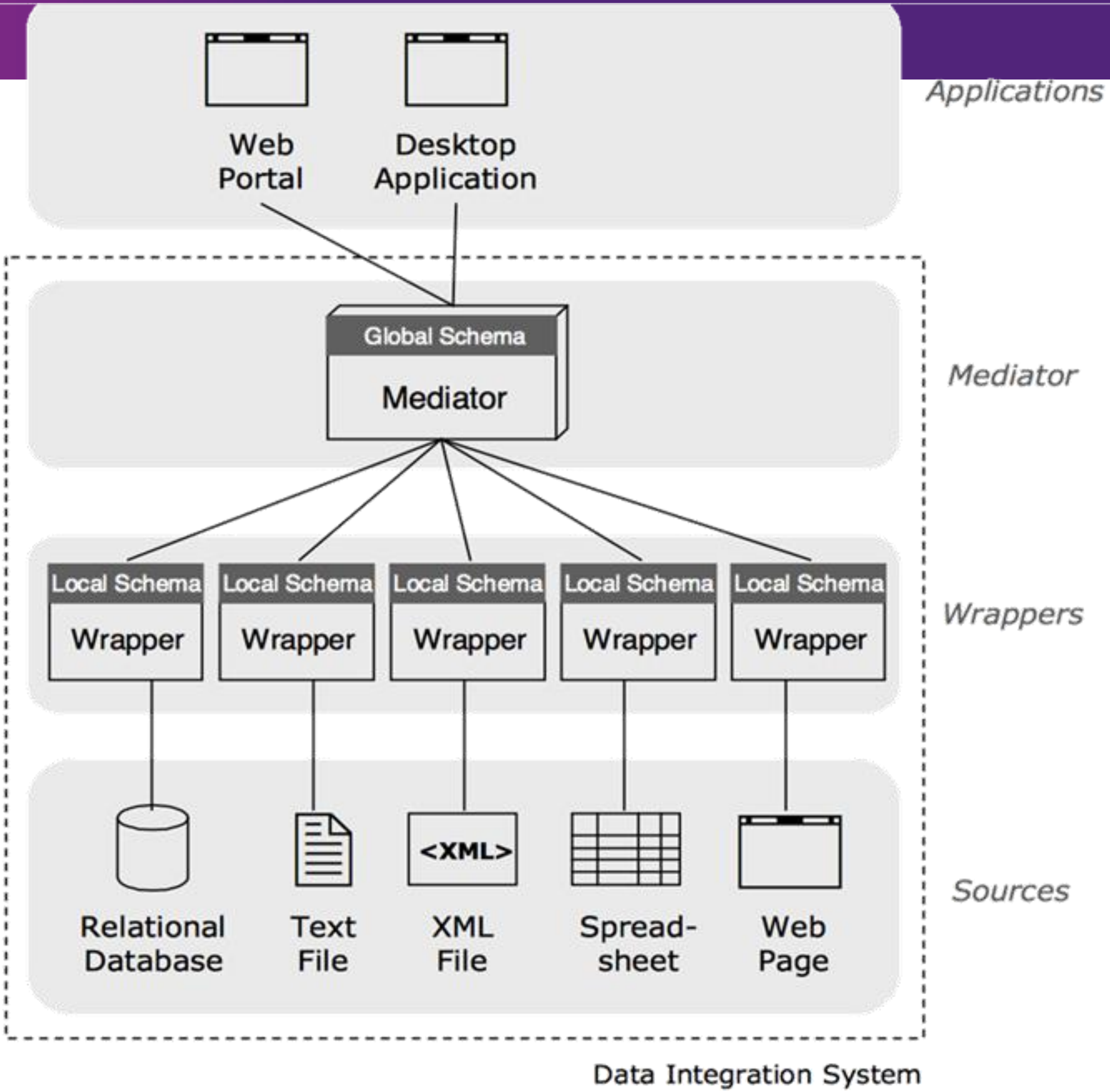
Is this architecture suitable for single or multiple organizations?

Data Integration Example: Mediator- Wrapper Architecture

- **Mediator**
  - ✓Centralizes the information provided by the wrappers in a unified view of all available data
  - ✓Decomposes user queries, and gathers partial results to compute query results
- **Wrapper**
  - ✓Provide a mapping between a source DBMSs view and the mediators' view



How many Mediators are required?



Benefits of Mediator/Wrapper:

- **Flexibility**: It allows for the integration of new data sources without changing the rest of the system. You only need to add a new wrapper for the new source.
- **Scalability**: The system can scale as new data sources are added or as the demand for data increases.
- **Data Autonomy**: Data sources can maintain control over their data and how it is accessed since the mediator respects the interface provided by each wrapper.
- **Heterogeneity**: The architecture can handle differences in data models, query languages, and data semantics across various data sources.

Challenges in DB Integration

- Each database could be in a different type of DBMS
  - Relational, semi-structured, NoSQL…
- Schema heterogeneity
  - S1: **Employee** (ID, name, address, position, salary, from, until)
  - S2: **Worker** (EID, name, address); **Position** (EID, PID, salary, from, until)
  - S3: **Name** (EID, address, salary, startingDate)
- Type heterogeneity
  - Employee ID could be a string or an integer

```sql
-- CAST one data type to another
SELECT column_name, CAST(column_value AS target_data_type) AS new_column
FROM source_table;

-- functions like TO_DATE, TO_TIMESTAMP, TO_NUMBER can help convert strings into date/time or numeric values.

SELECT TO_DATE(date_string, 'YYYY-MM-DD') AS formatted_date
FROM source_table;
```

- Value heterogeneity
  - The “*cashier*” position could be called “*associate*” in another system
- Semantic heterogeneity
  - Salary is *hourly salary*, or is *weekly salary* with allowances

Three Steps for DB Integration

Schema mapping: mapping of structures

- involves establishing **correspondences** between the structures of different databases.
- translates elements from one database schema to another.
- E.g. \`Customer.PhoneNumber\` ← → \`Clients.ContactNumber\`.

Data mapping: matching based on content

- is the process of **matching** and **aligning** the data based on its content.
- involves translating data values between systems according to the schema mapping and can involve transforming data formats, units, or other value transformations.
- E.g. String \`Customer.PhoneNumber\` ← → Integer \`Clients.ContactNumber\`

Data fusion: reconciliation of mismatching content

- is about **reconciling** and **merging mismatching content** from different databases.
- aims at resolving conflicts, ensuring that the integrated data is consistent and accurate.
- E.g. \`+61-0430123456\` in \`Customer.PhoneNumber\` vs \`61430123456\` in \`Clients.ContactNumber\`.

Example of Schema Difference

Consider **two companies'** data models :

Company-1: all records are stored in one table:

Emp (<u>Emp#</u>, Fname, Lname, Bdate, Dept#, Rank, Salary)

Company-2: uses one for each department:

DeptXX (<u>S-id</u>, Fname, Sname, Position, Phone#, e-mail, URL)

So we can build an integrated schema for both

Employee (<u>EmpID, DeptID</u>, Fname, Lname)

Schema Mapping

Mapping of structure

```sql
-- Create integrated_db
CREATE DATABASE integrated_db;

-- Connect to integrated_db
\c integrated_db;

-- Create tables in integrated_db based on schema mapping
CREATE TABLE integrated_table (
    id serial PRIMARY KEY,
    common_column text,
    db1_specific_column text,
    db2_specific_column text
);
```

Data Mapping

Mapping of data

```
\c db1;

-- Assuming db1 has a table named db1_source_table
SELECT id, common_column, db1_specific_column, NULL AS db2_specific_column
INTO integrated_table
FROM db1_source_table;

-- Connect to db2
\c db2;

-- Assuming db2 has a table named db2_source_table
INSERT INTO integrated_table (common_column, db1_specific_column, db2_specific_column)
SELECT common_column, NULL AS db1_specific_column, db2_specific_column
FROM db2_source_table;
```

Data Fusion

Removing duplicates

```sql
-- Removing duplicates based on common_column
DELETE FROM integrated_table
WHERE id NOT IN (
    SELECT MIN(id)
    FROM integrated_table
    GROUP BY common_column
);
```

id	common_column	others
1	A	
2	A	
3	B	
4	B	
5	B	

SELECT MIN(id) FROM integrated_table
GROUP BY common_column

1
3

View-Based Database Integration

Problem definition: **_<G, S, M>_**

- ***G***: the global schema
- ***S***: a set of local schemas
- ***M***: the mapping to translate queries between *G* and *S*

A global query is issued over *G* and processed over *S*

Two popular ways of view mapping

- **Global as View** (GAV): *G* is a set of views over *S*
- **Local as View** (LAV): *S* is a set of views over *G*

P.133, Ch4.1, Ozsu & Valduriez: Principles of Distributed Database Systems, 3rd Ed.
A Survey Paper: https://www.inf.unibz.it/~calvanese/papers/calv-lemb-lenz-D2I-D1R5-2001.pdf

View-Based Database Integration

- **Global as View** (GAV): *G* is a set of views over *S*
  - *M* associates each element in *G* as a query over *S*
  - **Employee.EmpID** 🡰 Emp.Emp# || DeptXX.S-id (**G** ← **S**)
  - *Take local schemas as input and map them into a global view to provide uniform access*
  - A GAV mapping is a set of queries on **local** sources $S_1, S_2, \ldots, S_n$, one for each element g in G, indicating how **global G** is constructed from **local S**.
- **Local as View** (LAV): *S* is a set of views over *G*
  - *M* associates each element in *S* as a query over *G*
  - Emp.Emp#🡰 **Employee.EmpID** DeptXX.S-id 🡰 **Employee.EmpID (S ← G)**
  - *Take an existing global database and maps "backwards " into local views for specific application*
  - An LAV mapping is a set of queries on the **global** schema, one for each local source, indicating how **local sources** contribute to the **global schema.**

P.133, Ch4.1, Ozsu & Valduriez: Principles of Distributed Database Systems, 3rd Ed.

A Survey Paper: https://www.inf.unibz.it/~calvanese/papers/calv-lemb-lenz-D2I-D1R5-2001.pdf

View-Based Database Integration – GAV Example

S1

ID	Name	Program
1	Alice	BIT
2	Bob	MCS

S2

ID	Name	Program
1	Alice	BIT
3	Tom	MCS

G_students

ID	Name	Program
1	Alice	BIT
2	Bob	MCS
3	Tom	MCS

Global tables/views are just **queries over local tables**.

- Easy to **query** because mappings are explicit.
- Harder to **maintain** if local sources change.

```
CREATE VIEW G_Students AS
SELECT S1.ID AS ID, S1.Name AS Name, S1.Program AS Program
FROM S1
UNION
SELECT S2.ID AS ID, S2.Name AS Name, S2.Program AS Program
FROM S2
```

```
SELECT * FROM G_Students WHERE Program='MCS'
```

ID	Name	Program
2	Bob	MCS
3	Tom	MCS

P.133, Ch4.1, Ozsu & Valduriez: Principles of Distributed Database Systems, 3rd Ed.
A Survey Paper: https://www.inf.unibz.it/~calvanese/papers/calv-lemb-lenz-D2I-D1R5-2001.pdf

View-Based Database Integration – LAV Example

S1

ID	Name	Program
1	Alice	BIT
2	Bob	MCS

G_students

ID	Name	Program
1	Alice	BIT
2	Bob	MCS
3	Tom	MCS

Each source describes how it fits into the overall global view.

- Easier to add new sources (flexible).
- Query answering is harder (requires reasoning/inference).

S2

ID	Name	Program
1	Alice	BIT
3	Tom	MCS

```
-- Mapping for S1
SELECT ID, Name, Program
FROM S1
=> corresponds to G_students(ID,Name, Program)
-- Mapping for S2
SELECT ID, Name, Program
FROM S2
=> corresponds to G_students(ID,Name, Program)
```

Conceptually Written

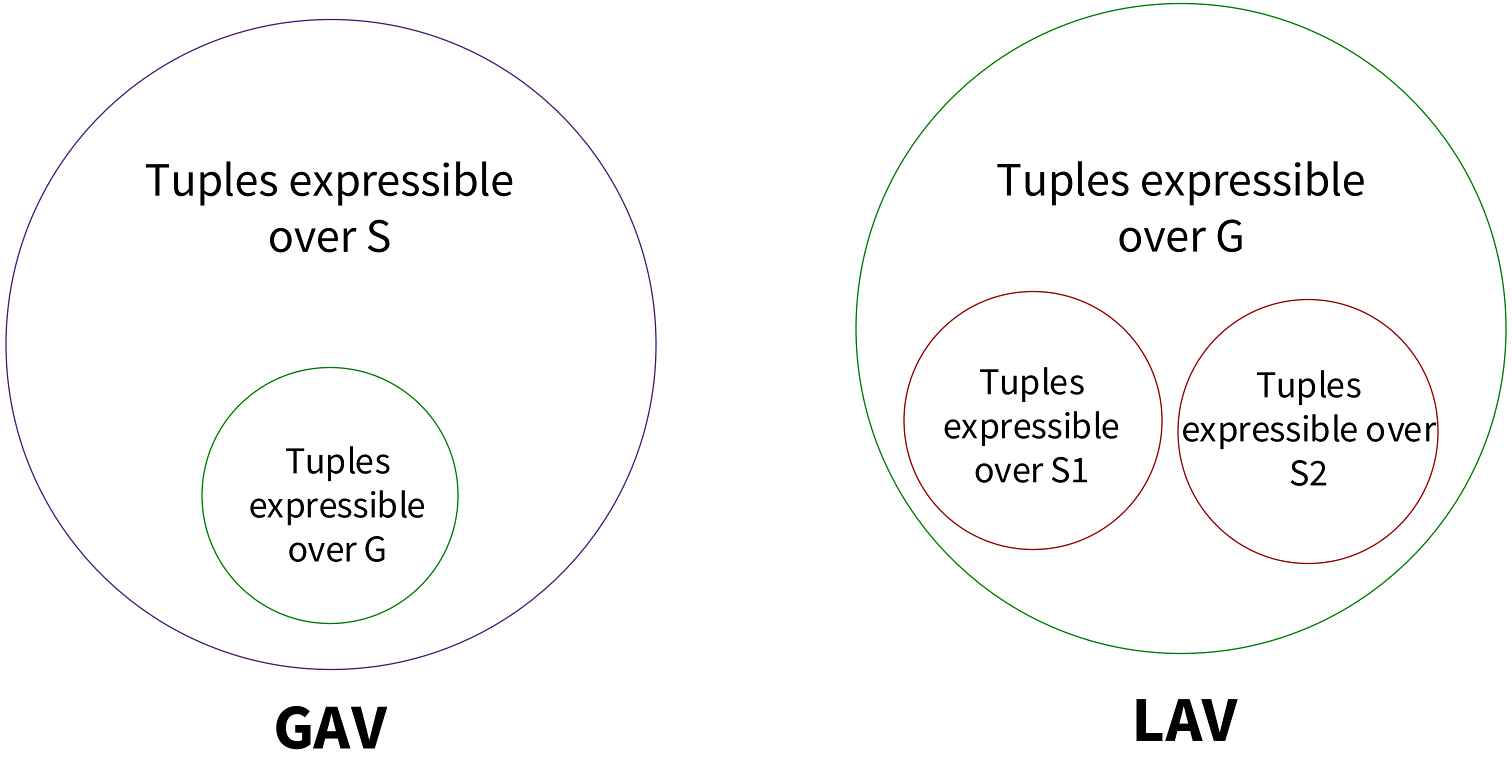
```
SELECT * FROM G_Students WHERE Program='MCS'
```

ID	Name	Program
2	Bob	MCS
3	Tom	MCS

P.133, Ch4.1, Ozsu & Valduriez: Principles of Distributed Database Systems, 3rd Ed.

A Survey Paper: https://www.inf.unibz.it/~calvanese/papers/calv-lemb-lenz-D2I-D1R5-2001.pdf

A Graphical View on view-based database integration



- In **GAV**, the set of possible tuples is defined on source *S*. While the set of tuples expressible over the sources *S* may be much larger and richer.
- In **LAV**, the set of possible tuples for each source S_i is defined on G. While the set of tuples expressible over *G* can be much larger (thus, LAV must deal with incomplete answers).

A Comparison Table for GAV vs LAV

Feature	GAV	LAV
Mapping direction	Global schema = Query over local	Local schema = View over global
Query rewriting	Easy	Complex (need inference)
Adding new sources	Difficult (needs redefining views)	Easier (just add new mappings)
Maintenance	Harder when sources change	Easier to maintain
Good for	Static environments (e.g. DW)	Dynamic, growing environments (e.g. FDB)

Limitations of Views

Views are for structures, not semantics

- Views in general are not possible to address data integration problems due to semantic heterogeneity where similar terms could mean different things in different systems

Semantic heterogeneity is less of a problem where organizations do business together

- To do business, the organizations must agree on the terms involved
- **Integrated systems** require a global schema (ontology) developed before the application, and the participating systems must give up some of their autonomy to commit to the ontology

An **ontology** is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse. It is thus a practical application of philosophical **ontology**, with a taxonomy.

Semantic Issues

Consider this application to have an integrated Student table…

A consortium of universities has a global schema with two tables

- Student (<u>ID</u>, PublicID, StudentStatus, VisaStatus)
- Services (<u>StudentStatus, VisaStatus</u>, ServicesApplying)

A distributed system might have Student table at each university

- StudentU (<u>ID</u>, PublicID, StudentStatus)

And a separate server for

- StudentV (<u>ID</u>, PublicID, VisaStatus)

A query linking students with services needs to navigate both StudentU and StudentV to get the key for Services

The Bottom-Up Approach

StudentU is operated by a *consortium of universities*

StudentV is operated by the *Department of Immigration*

Services operated by someone else

Need to consider

- Agreement on identification of instances
- Coverage of instances
- How these affect queries

Agreement on Instance IDs

StudentU(<u>IDu</u>, PublicIDu, StudentStatus) @Universities

StudentV(<u>IDv</u>, PublicIDv, VisaStatus) @Immigration Dept

How do we get association between student and status fields?

- No reason to suppose **IDu** and **IDv** are related
- PublicIDu could be name and date of birth
- PublicIDv could be thumbprint

Join is impossible, *even if* the two systems have information on exactly the same people

Organizations must agree on IDs and at least one must gather more data and maintain a correspondence between two sets of IDs

Coverage of Instances

Even with an agreement on IDs, in practice, the two systems could cover different populations

- StudentU may include domestic as well as overseas students
- StudentV may include all sorts of student visa holders, not just university students

What does it tell us?

- If a person is linked to both StudentStatus and VisaStatus by this system
- If a person is linked to one but not the other

Depends on how reliably the two systems are updated, and on how frequently

- Need agreements on quality of service (QoS)

Ontological Commitment

Building a multi-database bottom-up by schema and population integration can be problematic

A global schema can only be created by agreement (ontology)

Each participant must commit to the ontology

- Create views, often modify schemas
- Often introduce global identifiers like ISBN and establish correspondences between local and global identifiers

This is actually a *top-down* approach, no longer a *bottom-up* approach

Semantics/Ontology and Standards

A very large community of researchers is working on improving understanding of data (schema and value) semantics

- Knowledge graph

Many influential efforts towards providing standards for data exchange

Motivation resides in the ability to allow disparate systems to interoperate seamlessly

The goal of the Semantic Web is to make Internet data machine-readable.

For Reference on Semantic Web, see: www.w3.org/DesignIssues/Semantic.html

Outline of Data Linkage

- Identifying records referring to **the same real-world entity**
- Computing data **similarity**
- Applicability of different similarities with different **granularity**

What is Data Linkage?

An operation to identify records referring to the same real-world entity

- a.k.a. *data cleaning, data scrubbing, record linking, de-duplication, fuzzy matching, entity resolution, merge/purge, reference reconciliation, data hardening, entity disambiguation…*

An essential pre-step for data integration and data mining

- For data integration
- For duplication detection

Increased interest recently for automatic entity resolution, because of increasing complexity in:

- Data volumes
- Data diversity
- Data usage

What Data Linkage is Not

Data linkage is related but different from:

- *System heterogeneity* (to link data stored in different systems)
- *Structural heterogeneity* (e.g., address vs (street#, street_name, city…)
- *Mirror detection* (e.g., near-duplicate docs or web pages)
- *Co-reference resolution* (e.g., *I graduated from the University of Queensland. It is one of the best universities in the world*)

Typically, **semantic equivalence** issues are not considered (too hard)

- E.g., **"oz" = "Australian"**

Records with a well-defined ID are not considered (too easy)

Application: Data Integration

Company **acquisition and merge**

Data Preprocessing in an ETL (Extraction, Transformation, Loading) process

The **whole-of-X** approach for a grand information system (with an ontology database)

- X: government, health care, water systems…
- Different data sources are integrated (*horizontally* or *vertically)* according to business workflows

Data mining and data warehousing

ID	Given_name	Surname	DOB	Gender	Address	Loan_type	Balance
6723	peter	robert	20.06.72	M	16 Main Street 2617	Mortgage	230,000
8345	smith	roberts	11.10.79	M	645 Reader Ave 2602	Personal	8,100
9241	amelia	millar	06.01.74	F	49E Applecross Rd 2415	Mortgage	320,750

Bank database

Health database

PID	Last_name	First_name	Age	Address	Sex	Pressure	Stress	Reason
P1209	roberts	peter	41	16 Main St 2617	m	140/90	high	chest pain
P4204	miller	amelia	39	49 Aplecross Road 2415	f	120/80	high	headache
P4894	sieman	jeff	30	123 Norcross Blvd 2602	m	110/80	normal	checkup

Application: Duplicate Detection

90% data cleaning work is associated with **de-duplication** when archiving data by inputting files to data warehouse

- [National security] [www.dailykos.com/story/2006/1/5/85158/32663]: Some innocent people included in "*No Fly Watch List*"
- [Statistics]: One patient has several diagnosis records
- [Marketing]: One customer has several records - sending extra catalogues

De-Duplication: Merge & Purge Process

[Win06] Overview of record linkage and current research directions, W. E. Winkler, Research report series (Statistics #2006-2), Statistical Research Division, U.S. Census Bureau.

The Causes of Differences

Typographical errors (e.g. date -> data)

Missing or uncertain values

Phonetic issues (e.g. Jon vs John, Sara vs Sarah)

Numeric issues (e.g. $ - USD or AUD?)

Inconsistent abbreviations (e.g. St. -> saint or street?)

Some ‘natural’ causes:

- Context-related variations (e.g. football in US and Europe)
- Dynamic nature of data (variations over time, regions, disciplines)
- Historical reasons, …

…

Why Difficult?

Beyond string similarity

- The same real-world object can be represented as different strings (NYC, New York City)
- The same string can represent different real-world objects (crane vs crane, Jaguar vs Jaguar)

Quadratic complexity to data sizes

- With very high complexity for ‘unit’ operations

Often no black-and-white answers – probabilistic answers

Need to consider privacy issues, too

Linking with Different Granularity

Data matching methods

- Field-level: for two **given attributes**, to decide if they are identical (e.g., two names, or two addresses)
  - This is the most basic form the entity linking
- Record-level: for **two database records**, to check if they are about the same entity (a.k.a data augmentation)
  - With more contextual information the linking accuracy can be improved
- Group-level: to check if **two groups of records** are about the same composite entity (e.g., two families with each record for one family member)
  - One-to-one mapping within the composite entity can help to improve linking accuracy

Group Match Example:

Database A contains records for a family with the following members:
- John Doe, father, admitted for surgery.
- Jane Doe, mother, admitted for consultation.
- Jack Doe, son, admitted for a minor injury.

Database B contains records with slight variations in the information:
- J. Doe, admitted for an operation.
- J. Doe, admitted for medical advice.
- J. Doe, Jr., admitted after an accident.

Matching records individually might lead to uncertainties due to common surnames and initials

Field Matching

Find similarity for two given text strings

- A basic operation for more complex similarity measures

Main problem to address: typographical issues

- Spelling Errors, e.g. “Jhn” instead of “John”
- Incorrect Input, e.g. “AIMS Bank” instead of “AIMS Finance”
- Case sensitivity: e.g. Search_string vs search_string

```
SELECT * FROM table_name WHERE column_name ILIKE 'search_string';
```

```
postgres=# SELECT 'hello' ILIKE 'HelLO';
 ?column?
----------
 t
(1 row)
```

Similarity by Edit Distance

The edit distance between two strings is the minimum number of operations to transform one string to another

- Operations: delete, insert or substitute one character

What's the edit distance?

- 'John', 'Jon'
- 'John' , 'Jhn'
- 'John', 'Josh'

There are many types of edit distance, the one with *insertion, deletion, substitution* is called Levenshtein distance. Other well-known edit distance types include Hamming distance, Longest Common Subsequence (LCS) distance.

```
SELECT * FROM table_name WHERE levenshtein(column_name, 'search_string') < 3;
```

Similarity by Edit Distance

Two strings are considered identical if their ED is less than a pre-defined threshold

- Normalization is recommended

$$sim(a,b) = 1 - \frac{ED(a,b)}{\max(|a|,|b|)}$$

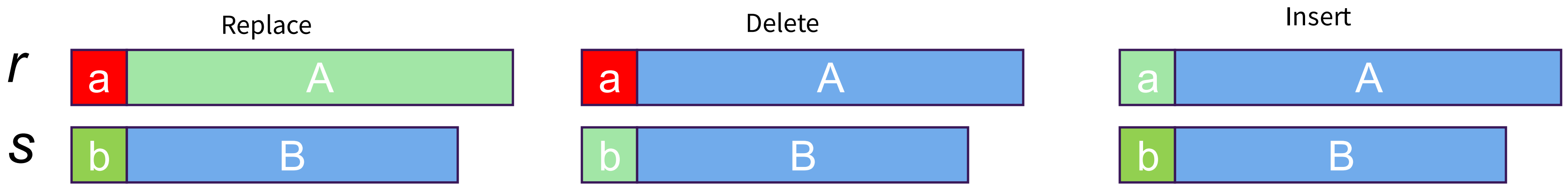
Usually asked questions about a distance:

- When is *ED(a, b)=0*?
- *ED(a, b)* vs. *ED(b, a)*?
- *ED(a, c)* vs. (*ED(a, b)* + *ED(b, c)*)?
- $|a| = m$ and $|b| = n$, what are the ***min*** and ***max*** *ED(a, b)*?
- How to compute ED for two given strings?
- How this can be done for two large data sets?

Edit Distance Computing

Dynamic programming algorithm

- Solve the current problem with the known results of **sub-problems**
- Keep reducing the problem size **recursively**



d(A, B)+1

d(A,“b”+B)+1

d(“a”+A, B)+1

r = “*a*” + *A*
s = “*b*” + *B*

Equation

$$ED(r,s) = \begin{cases} n, & m = 0 \\ m, & n = 0 \\ \min\begin{Bmatrix} ED(sub(r), sub(s)) + subcost, \\ ED(sub(r), s) + 1, \\ ED(r, sub(s)) + 1 \end{Bmatrix}, & otherwise \end{cases}$$

$$subcost = \begin{cases} 0, & head(r) = head(s) \\ 1, & otherwise \end{cases}$$

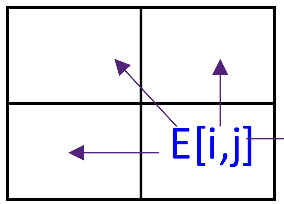
An Example

Use (|r|+1)×(|s|+1) matrix E

Start from E[0,0], and **E[| r |,| s |] is the edit distance**

$$E[i,j] = \begin{cases} [i-1, j-1] & if\ r_i = s_j, \\ \min([i, j-1], [i-1, j-1], [i-1, j]) + \mathbf{1} & if\ r_i \neq s_j \end{cases}$$

Complexity is O(|r|×|s|)



j=1, 2, 3, 4

i=1, 2, 3

E [i,j]		j	o	h	n
j					
h					
n					

Result

G. Navarro, "A Guided Tour to Approximate String Matching," ACM Computing Survey 2001

Another Example

ED(“Dubios”, “Dubose”) ?

- E[i,j] = [i-1, j-1] **if $r_i = s_j$**
- E[i,j] = **min**([i, j-1]+1, [i-1, j-1]+1, [i-1, j]+1) **if $r_i \neq s_j$**

		D	U	B	O	S	E
	0	1	2	3	4	5	6
D	1	0	1	2	3	4	5
U	2	1	0	1	2	3	4
B	3	2	1	0	1	2	3
I	4	3	2	1	1	2	3
O	5	4	3	2	1	2	3
S	6	5	4	3	2	1	2

Comments on Edit Distance

The ED discussed so far is known as Levenshtein Metric (1965)

Observations

- A costly operation for large strings
- **Suitable for common typing mistakes**
  - "Comprehensive" vs "Comprenhensive"
- **Problematic for specific domains or abbreviations**
  - "*AT&T Corporation*" vs "*AT&T Corp*"
  - "IBM Corporation" vs "AT&T Corporation"
  - "*ITEE*" vs "*IEEE*" vs "*School of Information Technology and Electrical Engineering*"

Similarity by Tokenization (q-grams)

Varying semantics of 'term'

- Words in a field
  - S= 'AT&T Corporation' -> S1 = 'AT&T' , S2 = 'Corporation'
- q-grams (sequence of q-characters in a field, a.k.a. *n-grams*)
  - *{'AT&','T&T','&T_', 'T_C','_Co', 'Cor','orp','rpo','por','ora','rat','ati','tio','ion'} (a total of* ***14*** **3-grams)**

    |q-gram(S)| = n – q +1; where |S| = n; q=3;
  - Can add *'##A', '#AT'* and '*on#', 'n##'* to the set for the ends of sequence (a total of **18** 3-grams)

    |*q-grams(S)*| = n + q -1; where |S| = n; q=3;

Calculate similarity by manipulating **sets** of terms

Question

- For a string of *n* characters, how many *q*-grams does it contain?
- |q-gram(S)| = n – q +1; where |S| = n; q=3; or
- |*q-grams(S)*| = n + q -1; where |S| = n; q=3; (with "#")

q-Gram and Jaccard Coefficient

Idea: if two strings share many q-grams, they can be considered as similar

Given two sets of terms S, T

- Jaccard coefficient: **Jaccard(S,T) = |S∩T|/|S∪T|**

A common technique used in language processing

- Text recognition, spelling checking
- Insensitive to word orders: "**University of Queensland**" vs "**Queensland University**"
- Can be computationally efficient

Problem

- "School of EECS" vs. "EECS" vs. "School of Art"

Similarity Measure with TF/IDF

Term frequency (**tf**) inverse document frequency (**idf**)

- tf : # of times 'term' appears in a document
- idf : *number of documents (N) / number of documents containing 'term' (n)*
- Term score: tf*idf

Widely used in traditional IR (Information Retrieval) approaches

- Intuitively: a term which appears **frequent 'locally', but rare 'globally'** is more important
- "School of EECS": w_1 * "school", w_2 * "of", w_3 * "EECS"
  - $w_3 > w_1 > w_2$

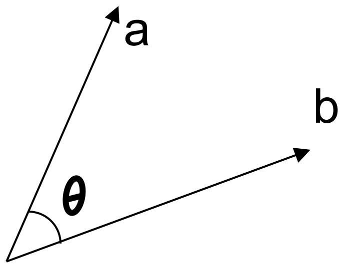
$$weight_{string}(token) = \log(tf_{token} + 1) * \log(idf_{token})$$

[Ww98] Integration of Heterogeneous databases without common domains using queries based on textual similarity, sigmod 1998

Cosine Similarity for TF/IDF

Each field value is transformed via **tf/idf** weighting to a vector in ***d* high dimensional document space**.

Let ***a***,***b*** be two field values and $\boldsymbol{V_a}$, $\boldsymbol{V_b}$ be the set of **tf/idf** scores for terms in ***a*** and ***b***.



$$sim(a,b) = \frac{\sum_{i=1..d} V_a(i) \bullet V_b(i)}{\| V_a \|_2 \bullet \| V_b \|_2}$$

Numeric Similarity

Numbers with similar values but presentations are dissimilar

- For example: **500** is similar to **499**

Two straightforward ways

- Treat a number as a word
  - ➢Similarity of two numbers is measured by edit distance
  - ➢Problem: e.g. **500** versus **{499, 800}**
- Numbers a, b are similar if a, b in a range T, that is, $|a-b| < T$
  - ➢Problem: what is a proper range?

Record Matching (3-1)

Problem: measuring similarity with multiple attributes

- Q: combine all attributes together into a long string?
- Problems with string concatenation

Name	**Address**
RAM Finance	16, Finance St, South Bank, QLD

"RAM Finance 16, Finance St, South Bank, QLD"

- Finance is universal in *name* field, **less** discrimination power
- Finance is rare in *address* field, **more** discrimination power

Record Matching (3-2)

Weighted-Sum

Measure the distance between individual fields, and then compute the weighted distance between records.

- Static Weight
  - e.g. ***sim = sim(name)* w + sim(address)* (1-w)***
  - Easy to implement but a good value of ***w*** is not obvious
- Dynamic Weight [NAD04]
  - Give more weight to the longer field so as to unify the influence of field length to the similarity

[NAD04] Flexible string matching against large databases in practise (2004). N. Koudas, A. Marathe and D. Srivastava

Record Matching (3-3)

Rule-Based Approaches

Equation between records can be inferred by specified **equation theory**

- Equation theory dictates the logic of domain equivalence, not simple value or string equivalence [HS95]

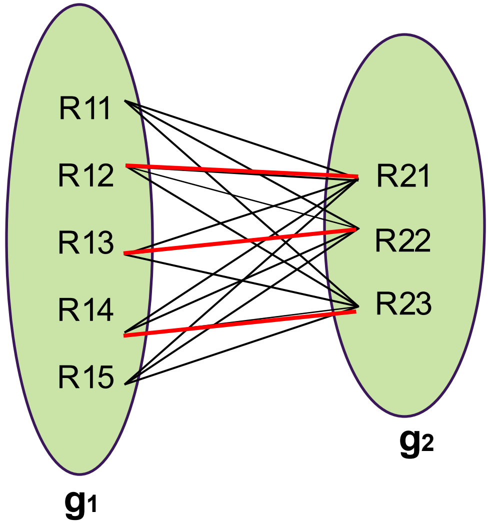
> Given two records, r_1 and r_2
> IF the ***last name*** of r_1 equals the ***last name*** of r_2,
> AND the ***first names*** differ slightly,
> THEN
> $r_1 = r_2$

[HS95] M.A. Hernandez and S.J. Stolfo, The merge/purge problem for large databases, Sigmod 1995

Group Matching

Discover two groups of records referring to same entity, e.g. a family

- For two groups (g_1, g_2), ($|g_1| = m_1, |g_2| = m_2$), similarity between all pairs of records is computed as a *Bipartite Matrix (BM)*
- Construct a bipartite graph (**|*M*| = *number of matched pairs***)
- Find the maximum weight matching from the graph
  - ➢ 1:1 matching between records in two groups (***sim(r_{1i},r_{2j})***)
  - ➢ Summation of similarity is maximized



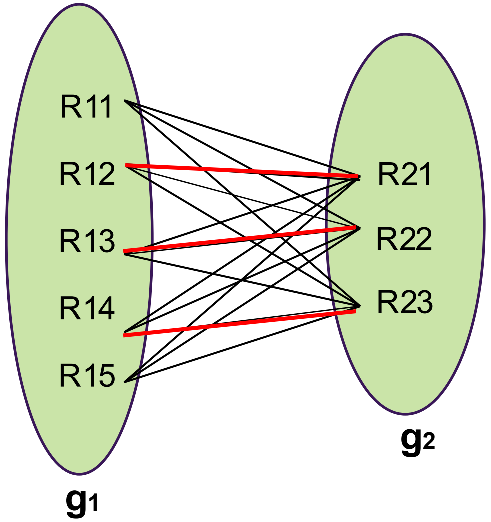
$$BM = \frac{\sum_{(r_{1i}, r_{2j} \in M)} (sim(r_{1i}, r_{2j}))}{m_1 + m_2 - |M|}$$

[BND07] B.W. On, N. Koudas, D. Lee, D. Srivastava, Group Linkage, ICDE 2007

Group Matching – cont'd

Pair	Similarity
R11, R21	0.1
R11, R22	0.2
R11, R23	0.3
R12, R21	0.9
R12, R22	0.3
R12, R23	0.4
R13, R21	0.5
R13, R22	0.9
R13, R23	0.3

Pair	Similarity
R14, R21	0.1
R14, R22	0.2
R14, R23	0.3
R15, R21	0.3
R15, R22	0.4
R15, R23	0.9



Option 1: (R11,R21); (R12,R22); (R13,R23) -> 0.1+0.3+0.3

…

Maximized

…

Option x: (R12,R21); (R13,R22); (R14,R23) ->**0.9+0.9+0.9**

In this process:

1. **Compute Similarity**: Calculate how similar each pair of records from the two groups are.
2. **Construct Bipartite Graph**: Create a graph where each group forms one set of nodes, and edges represent possible matches weighted by similarity.
3. **Maximum Weight Matching**: Find the optimal matching where the total similarity is maximized, resulting in the best matches between records from G1 and G2.

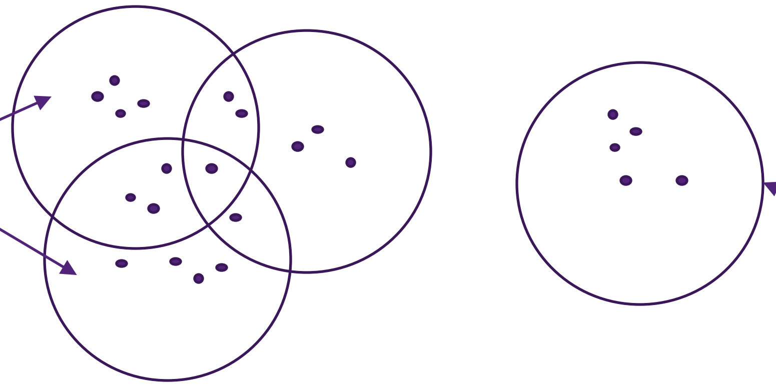
Canopy Clustering

Two steps:

The canopies approach is applicable for data linkage problem to deal with (1) a large number of ***records*** *with (2) high* ***dimensions****, that are required to derive (3) a big number of* ***clusters****.*

- **Step 1:** Data are divided (by **inexpensive distance** measurement) into overlapping subsets, called canopies
  - *E.g., Patients with a common diagnosis fall in same canopy.*
  - *E.g., Publications with one same author fall in same canopy*
  - *E.g., Using inverted index in a search engine: two documents having a certain number of common words fall in same canopy.*
- **Step 2: Expensive distance** measurement made among points within a same canopy.

Points not appearing in any common canopy are not possibly be in the same cluster.



An **inverted index** is a sparse matrix representation in which, for each word, we can directly access the list of documents containing that word.

[MKL00] MA. McCallum, K. Nigam, L. H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, KDD (2000)

Summary

DB Integration:

- Why DB Integration and Related Issues
- Global Information Systems
  - Federated Databases
  - Multidatabases
- Mediator-Wrapper Architecture
- Challenges in DB Integration
- Three Steps for DB Integration
- View-based DB Integration
  - Global-as-view vs Local-as-view
- Limitations of Views

Data Linkage:

- Why Data Linkage & What is NOT Data Linkage
- Data Linkage Applications
- Linking with Different Granularity
  - Field Matching
    - Edit Distance
    - q-Gram and Jaccard Coefficient
    - TF/IDF and Cosine Similarity
    - Numeric Simililarity
  - Record Matching
    - Weighted Sum & Rule-based Approaches
  - Group Matching
  - Canopy Cluster

Next week: Data Privacy