Logistics

- Lab 2 grades and keys will be posted today or tomorrow
- Lab 3 posted
  - Monday: Class, Q&A, Quiz
  - Tuesday: Homework Due 11:59pm
- Midterm dates
  - Monday (10/10) or Wednesday (10/12)

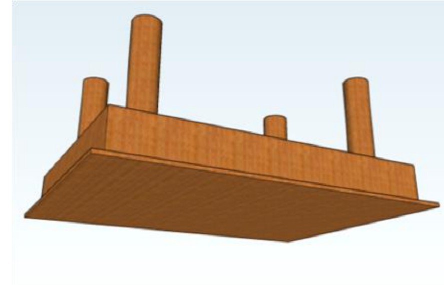
Review

- Tabular Data operations
  - Select, Project, Join
  - SPJ Query Semantics
  - Aggregation, Group by, Distinct, Sort…
- Data Cube operations
  - Slice, Dice
  - Pivot, Drill-Down/Up

OLAP tradeoffs

- Aggregates increase space and the cost of updates.
- On the other hand, since they are projections of data, or tree structures, the storage overhead can be small.
- Aggregates are limited, but cover a lot of common cases: avg, stdev, min, max.
- Operations (slice, dice, pivot, etc.) are conceptually simpler than SQL, but cover a lot of common cases.
- Good integration with clients, e.g. spreadsheets, for visual interaction, although there is an underlying query language (MDX).

Numpy/Matlab and OLAP



- Numpy and Matlab have an efficient implementation of nd-arrays for dense data.
- Indices must be integer, but you can implement general indices using dictionaries from indexval->int.
- Slicing and dicing are available using index ranges: a[5,1:3,:] etc.
- Roll-down/up involve aggregates along dimensions such as
  sum(a[3,4:6,:],2)
- Pivoting involves index permutations (.transpose()) and aggregation over the other indices.

Outline

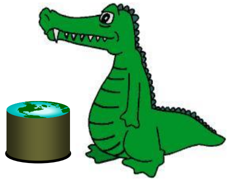
- Data Integration
  - Overview
  - Approximate Matching

Slides Adapted from “Principles of Data Integration”
By Anhai Doan, Alon Halevy and Zachary Ives

Data Integration

- Databases/Data Warehouses are great: they let us manage huge amounts of data
  - Assuming you've put it all into your schema.
- In reality, data sets are often created independently
  - Only to discover later that they need to combine their data! (When do we need to combine data?)
  - At that point, they're using different systems, different schemata and have limited interfaces to their data.
- The goal of data integration:

tie together different sources, controlled by many people, under a common schema.



DBMS: it's all about abstraction

- *Logical* vs. *Physical*; *What* vs. *How.*

Students:

SSN	Name	Category
123-45-6789	Charles	undergrad
234-56-7890	Dan	grad
	…	…

Takes:

SSN	CID
123-45-6789	CSE444
123-45-6789	CSE444
234-56-7890	CSE142
	…

Courses:

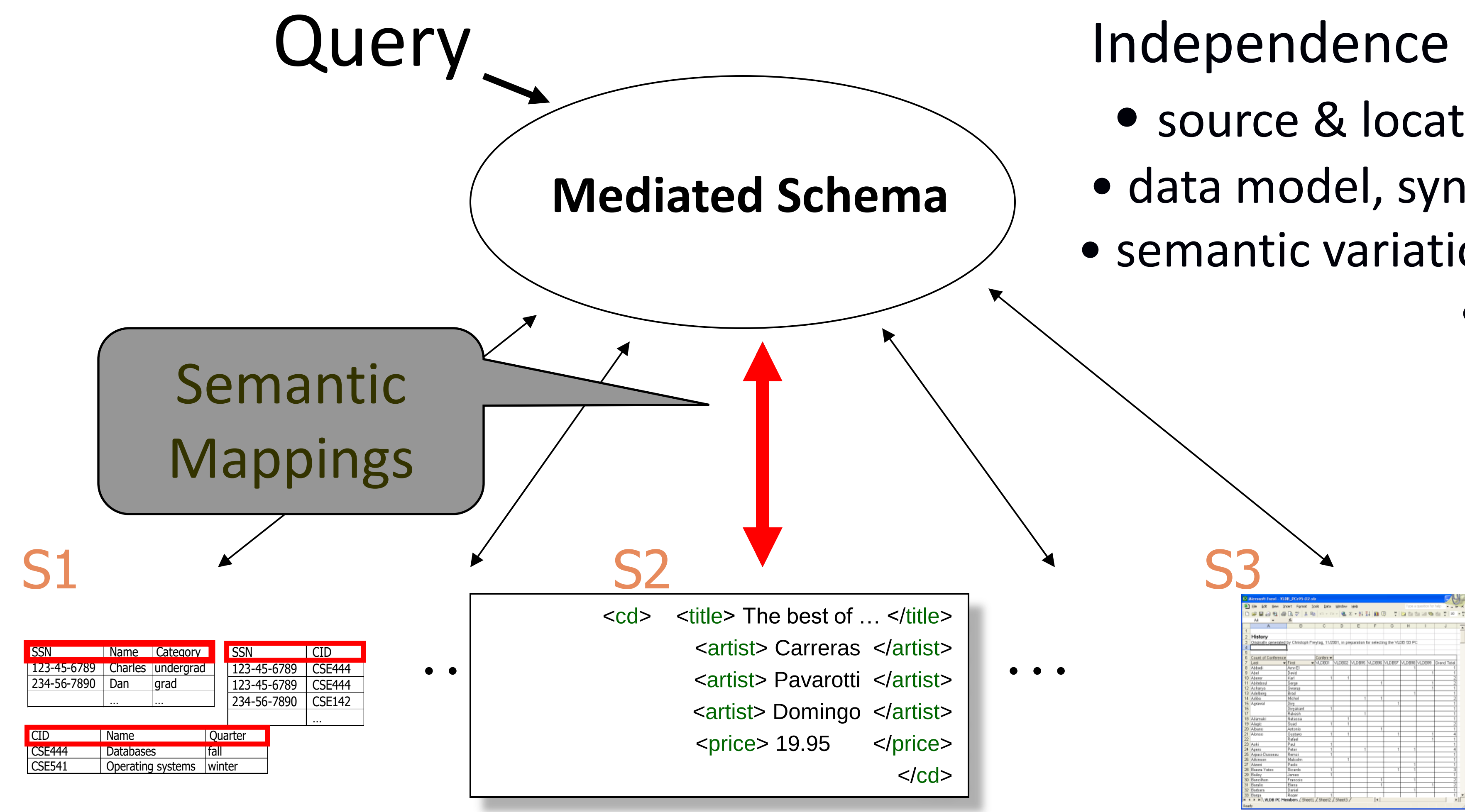
CID	Name	Quarter
CSE444	Databases	fall
CSE541	Operating systems	winter

```
SELECT  C.name
FROM Students S, Takes T, Courses C
WHERE S.name="Mary" and
        S.ssn = T.ssn and T.cid = C.cid
```

Data Integration: A Higher-level Abstraction

Independence of:

- source & location
- data model, syntax
- semantic variations
- …



Applications of Data Integration

- Business
- Science
- Government
- The Web
- Pretty much everywhere

Example: The Deep Web

- Millions of high quality HTML forms out there
- Each form has its own special interface
  - Hard to explore data across sites.
- Goal (for some domains):
  - A single interface into a multitude of deep-web sources.
- WebTables:
  - 2.6M Unique schemas (appear >1 time)
  - 5.4M Unique attribute (field) names (>1 time)
  - Found by web crawling/scraping

WebTables Extracted Tables

make	model	year
Toyota	Camry	1984

make	model	year
Mazda	Protégé	2003
Chevrolet	Impala	1979

make	model	year	color
Chrysler	Volare	1974	yellow
Nissan	Sentra	1994	red

name	addr	city	state	zip
Dan S	16 Park	Seattle	WA	98195
Alon H	129 Elm	Belmont	CA	94011

name	size	last-modified
Readme.txt	182	Apr 26, 2005
cac.xml	813	Jul 23, 2008

Schema	Freq
{make, model, year}	2
{make, model, year, color}	1
{name, addr, city, state, zip}	1
{name, size, last-modified}	1

Attribute Correlation Statistics Database (ACSDb)

- Schema Auto Complete
- Attribute Synonym-Finding
- Join Graph Traversal
- ACSDb is useful for computing attribute conditional probabilities

Goal of Data Integration

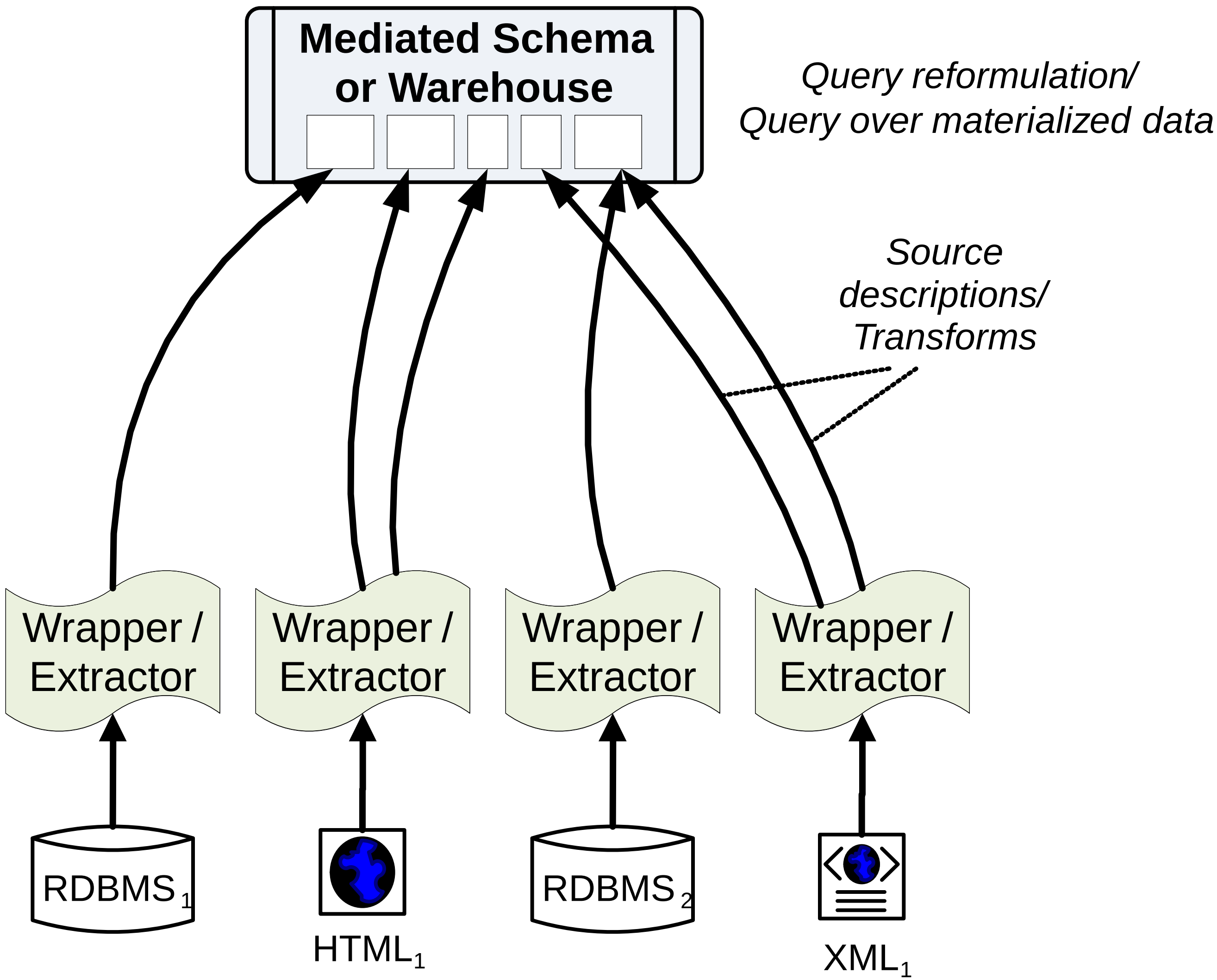
- Uniform query access to a set of data sources
- Handle challenges including
  - Schema/Data Heterogeneity
    - Approximate String/Data/Schema Matching
  - Type Heterogeneity: Semi-structure …
  - Scale of sources: from tens to millions
  - Support Autonomy

Virtual, Warehousing and in Between

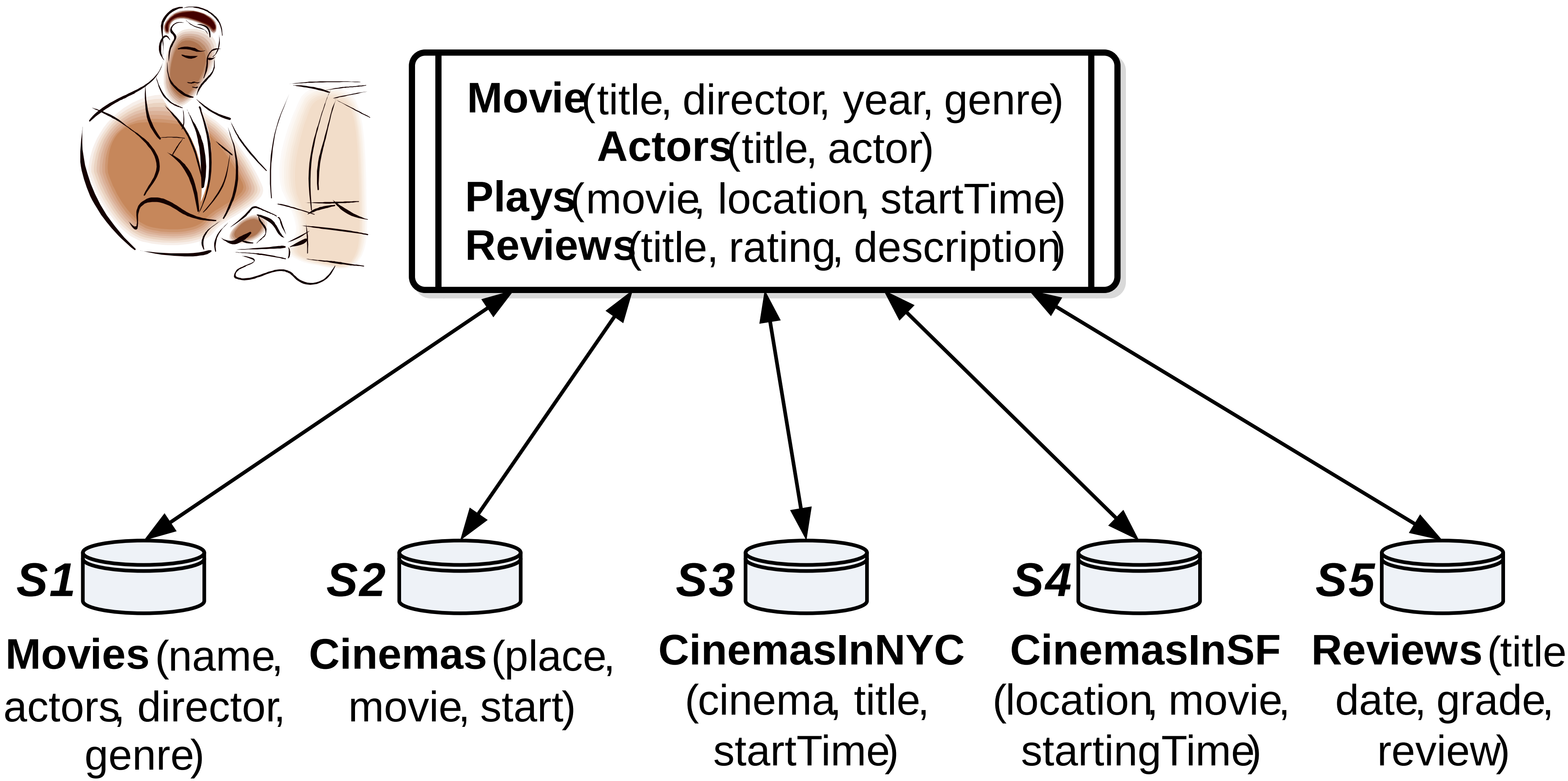
- Data warehousing: integrate by bringing the data into a single physical warehouse
- Virtual data integration: leave the data at the sources and access it at query time.

- Some differences, but semantic data/schema heterogeneity arises in both cases.
- Numerous intermediate architectures.

Virtual Data Integration Architecture

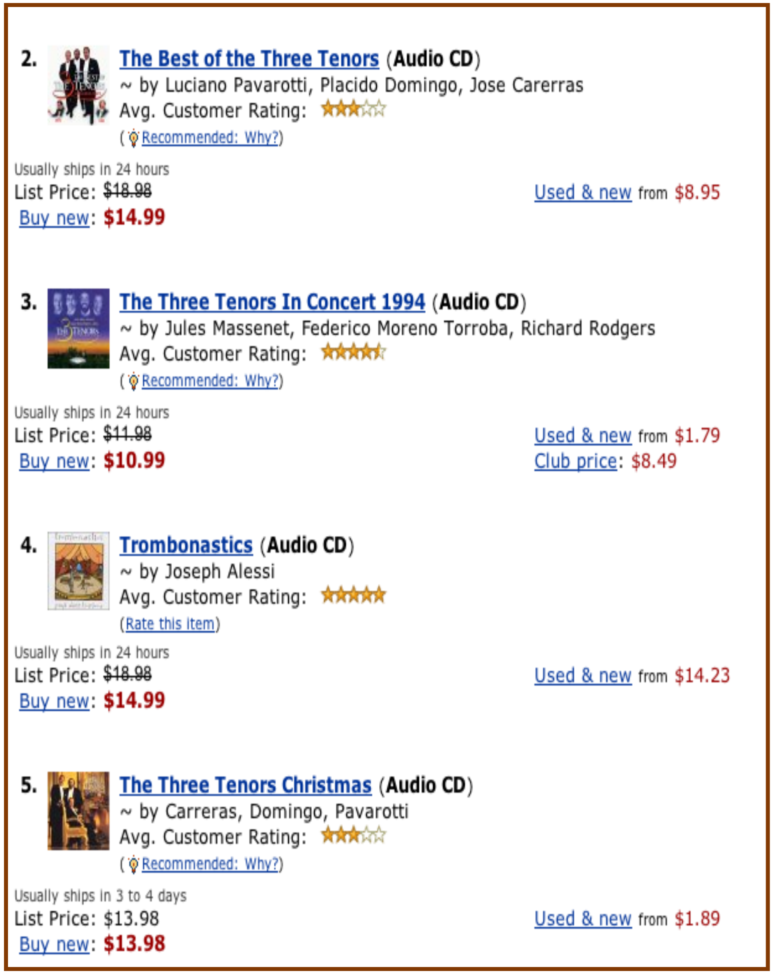


Example



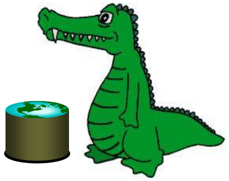
Movie(title, director, year, genre)
Actors(title, actor)
Plays(movie, location, startTime)
Reviews(title, rating, description)
S1
S2
S3
S4
S5
Movies (name, actors, director, genre)
Cinemas (place, movie, start)
CinemasInNYC (cinema, title, startTime)
CinemasInSF (location, movie, startingTime)
Reviews (title, date, grade, review)

Wrappers



<cd> <title> The best of … </title>
<artist> Abiteboul </artist>
<artist> Pavarotti </artist>
<artist> Domingo </artist>
<price> 19.95 </price>
</cd>
…

Send queries to data sources and transform answers into tuples (or other internal data model).

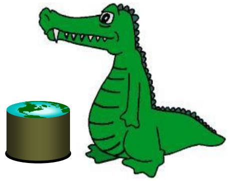


Example:
Woody Allen Comedies in NY

Mediated schema:

Movie: Title, director, year, genre
Actors: title, actor
Plays: movie, location, startTime
Reviews: title, rating, description

```
select title, startTime
from Movie, Plays
where Movie.title=Plays.movie AND
        location="New York"  AND
        director="Woody Allen"
```



Source Description And Matching

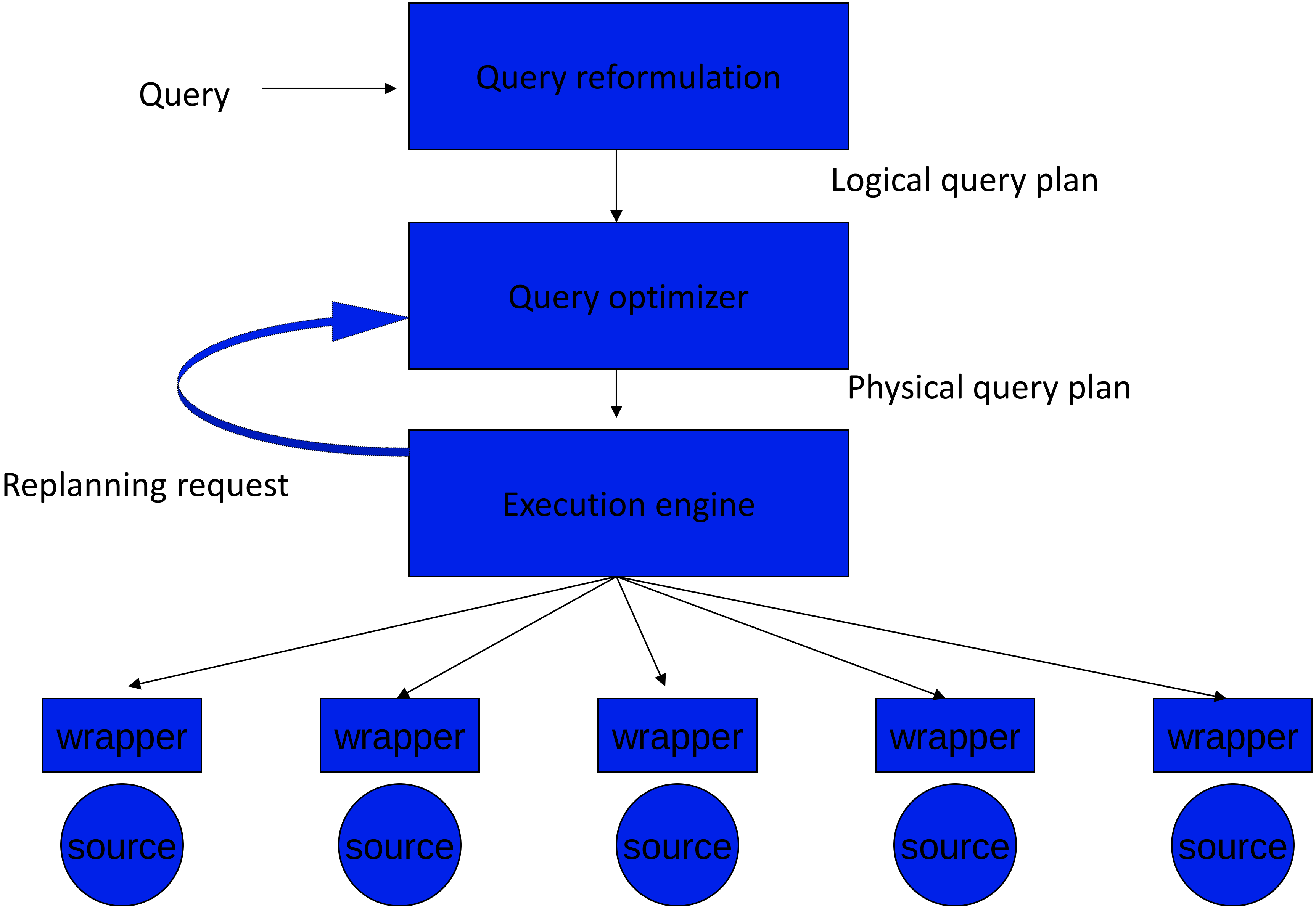
Movie: Title, director, year, genre
Actors: title, actor
Plays: movie, location, startTime
Reviews: title, rating, description

select title, startTime
from **Movie, Plays**
where Movie.title=Plays.movie AND
location="New York" AND
director="Woody Allen"

Sources S1 and S3 are relevant, sources S4 and S5 are irrelevant, and source S2 is relevant but possibly redundant.

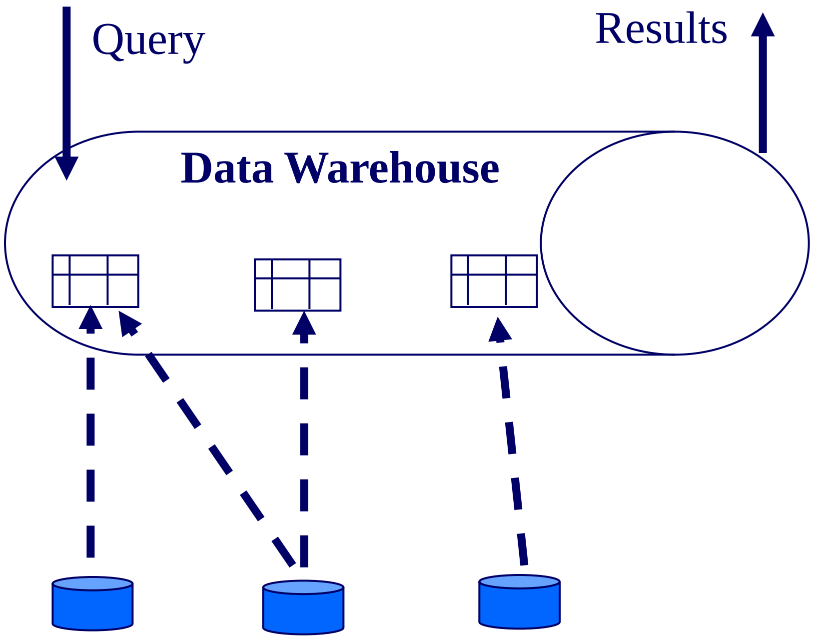
S1	S2	S3	S4	S5
Movies: name, actors, director, genre	Cinemas: place, movie, start	Cinemas in NYC: cinema, title, startTime	Cinemas in SF: location, movie, startingTime	Reviews: title, date grade, review

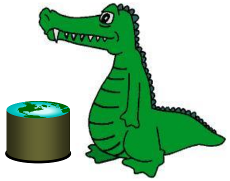
Query Reformulation and Processing



Data Warehouses – Offline Replication

- Determine physical schema
- Define a database with this schema
- Define procedural *mappings* in an "ETL tool" to import the data and clean it.
- Periodically copy all of the data from the data sources
  - Note that the sources and the warehouse are basically independent at this point





Pros and Cons of Data Warehouses

- ✘ Need to spend time to design the physical database layout, as well as logical
  - ✘ This actually takes a lot of effort!
- ✘ Data is generally not up-to-date (lazy or offline refresh)

- ✓ Queries over the warehouse don't disrupt the data sources
- ✓ Can run very heavy-duty computations, including data mining and cleaning

Approximate Matching

- Relate tuples whose fields are "close"
  - Approximate string matching
    - Generally, based on edit distance.
    - Fast SQL expression using a *q-gram* index
    - String as Set or Vector
  - Approximate tree/graph matching
    - For Nested Data Structures (or flattened ones)
    - Much more expensive than string matching
    - Recent research in fast approximations
  - Feature vector matching
    - Similarity search: collaborative filtering, K nearest neighbors
    - Many techniques discussed in the data mining literature.
  - Ad-hoc or Domain-focused matching
    - Use domain insights and/or clever tricks.

Some Similarity Measures

Handle
Typographical errors

- Equality on a boolean predicate
- Edit distance
  - Levenstein, Smith-Waterman, Affine
- Set similarity
  - Jaccard, Dice
- Vector Based
  - Cosine similarity, TFIDF

Good for Text like
reviews/ tweets

Good for Names

- Alignment-based or Two-tiered
  - Jaro-Winkler, Soft-TFIDF, Monge-Elkan
- Phonetic Similarity
  - Soundex
- Translation-based
- Numeric distance between values
- Domain-specific

Useful for
abbreviations,
alternate names.

From: Getoor & Machanavajjhala: “Entity Resolution Tutorial”, VLDB 2012

String Matching: Problem Description

- Given two sets of strings X and Y
  - Find all pairs x in X and y in Y that refer to the same real-world entity
  - We refer to (x,y) as a match
  - Example

Set X	Set Y	Matches
x_1 = Dave Smith	y_1 = David D. Smith	(x_1, y_1)
x_2 = Joe Wilson	y_2 = Daniel W. Smith	(x_3, y_2)
x_3 = Dan Smith		
(a)	(b)	(c)

- Two major challenges: accuracy(precision)/recall & scalability

Accuracy and Recall

		Diagnosis	
		No cancer	Cancer
True state	No cancer	*TN*	*FP*
	Cancer	*FN*	*TP*

precision: TP/cancer diagnoses

		Diagnosis	
		No cancer	Cancer
True state	No cancer	*TN*	*FP*
	Cancer	*FN*	*TP*

recall: TP/cancer true states

Accuracy Challenges

- Matching strings often appear quite differently
  - Typing and OCR errors: David Smith vs. Davod Smith
  - Different formatting conventions: 10/8 vs. Oct 8
  - Custom abbreviation, shortening, or omission: Daniel Walker Herbert Smith vs. Dan W. Smith
  - Different names, nick names: William Smith vs. Bill Smith
  - Shuffling parts of strings: Dept. of Computer Science, UW-Madison vs. Computer Science Dept., UW-Madison

Edit Distance

- Also known as Levenshtein distance
- d(x,y) computes minimal cost of transformi x into y, using a sequence of operators, each with cost 1
  - Delete a character
  - Insert a character
  - Substitute a character with another
- Example: x = David Smiths, y = Davidd Simth,
  - d(x,y) = 4, using following sequence
    - Inserting a character d (after David)
    - Substituting m by i
    - Substituting i by m
    - Deleting the last character of x, which is s

Edit Distance

- Models common editing mistakes
  - Inserting an extra character, swapping two characters, etc.
  - So smaller edit distance ➔ higher similarity
- Can be converted into a similarity measure
  - s(x,y) = 1 - d(x,y) / [max(length(x), length(y))]
  - Example
    - s(David Smiths, Davidd Simth) = 1 – 4 / max(12, 12) = 0.67

Edit Distance

- Character Operations: I (insert), D (delete), R (Replace).
- Unit costs.
- Given two strings, s,t, edit(s,t):
  - Minimum cost sequence of operations to transform s to t.
  - Example: edit(Error,Eror) = 1, edit(great,grate) = 2
- Folklore dynamic programming algorithm to compute edit();
- Computation and decision problem: quadratic (on string length) in the worst case.
  - May be costly operation for large strings
  - Suitable for common typing mistakes
    - Comprehensive vs Comprenhensive
  - Problematic for specific domains
    - AT&T Corporation vs AT&T Corp
    - IBM Corporation vs AT&T Corporation

From: Koudas, Sarawagi, Strivastava, “Record Linkage: Similarity Measures and Algorithms”, VLDB 2006