Logic as a Query Language: from Frege to XML

Victor Vianu

U.C. San Diego

Logic and Databases: a success story

- FO lies at the core of modern database systems
- Relational query languages are based on FO:

  SQL, QBE

- More powerful query languages (all the way to XML) are based on extensions of FO
- Foundations lie in classical logic

  FO : Frege relational algebra : Tarski

Why is FO so successful as a query language?

- easy to use syntactic variants

    SQL, QBE

- efficient implementation via relational algebra

    amenable to analysis and simplification

- potential for perfect scaling to large databases

    very fast response can be achieved

    using parallel processing

A relational database:

frequents

drinker	bar
Joe	King's
Joe	Molly's
Sue	Molly's
....	

serves

bar	beer
King's	Bass
King's	Bud
Molly's	Bass
....	

- logically a finite first-order structure

Find the drinkers who frequent some bar serving Bass

FO:

$$\{d\text{:drinker} \mid \exists\, b\text{:bar}\ (\text{frequents}(d,b) \wedge \text{serves}(b, \text{Bass}))\}$$

QBE:

frequents	drinker	bar
	d	b

serves	bar	beer
	b	Bass

answer	drinker
	d

Find the drinkers who frequent some bar not serving Bass

FO:

$$\{d:\text{drinker} \mid \exists\, b:\text{bar}\ (\text{frequents}(d,b) \wedge \neg\ \text{serves}(b, \text{Bass}))\}$$

QBE:

frequents	drinker	bar
	d	b

serves	bar	beer
¬	b	Bass

answer	drinker
	d

- Naïve implementation: nested loops

$\{d\text{:drinker} \mid \exists\ b\text{:bar}\ (\text{frequents}(d,b) \wedge \text{serves}(b, \text{Bass}))\}$

for each drinker

for each bar

check the pattern

Number of checks: |drinkers| × |bars|

Roughly n^2 : unacceptable for large databases!

- Better approach: relational algebra

Relational algebra operations

- union, difference

- selection σ $\qquad \sigma_{beer = Bass}$ (serves) =

bar	beer
King's	Bass
Molly's	Bass
....	

- projection π $\qquad \pi_{bar}$ (serves) =

bar
King's
Molly's
......

- join |×| frequents |×| serves

frequents

drinker	bar
Joe	King's
Joe	Molly's
Sue	Molly's
....	

serves

bar	beer
King's	Bass
King's	Bud
Molly's	Bass
....	

frequents |×| serves

drinker	bar	beer
Joe	King's	Bass
Joe	King's	Bud
Joe	Molly's	Bass
Sue	Molly's	Bass
....		

Relational algebra queries

Find the drinkers who frequent some bar serving Bass

$$\pi_{\text{drinker}} (\sigma_{\text{beer = Bass}} (\text{frequents} \bowtie \text{serves}))$$

drinker
Joe
Sue
…..

drinker	bar	beer
Joe	King’s	Bass
Joe	Molly’s	Bass
Sue	Molly’s	Bass
…………….		

drinker	bar	beer
Joe	King’s	Bass
Joe	King’s	Bud
Joe	Molly’s	Bass
Sue	Molly’s	Bass
…………….		

Theorem: Relational algebra and FO are equivalent

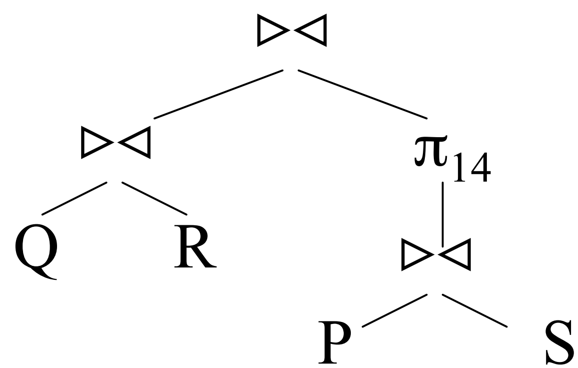
Journey of a Query

FO (SQL) $\exists z(P(xz) \wedge Q(zy)) \wedge \ldots$

Relational Algebra $\pi_{13}(P \bowtie Q) \bowtie \ldots$

Query Rewriting $\pi_{14}(P \bowtie S) \bowtie Q \bowtie R$

Query Execution Plan



Execution

Physical Level

- rewriting rules for algebra queries

$\pi_{drinker}(\sigma_{beer = Bass}(\text{ frequents } |\times| \text{ serves }))$

- efficient algorithms for individual operations

Indexes: special "directories" to data

cost: roughly n (log n)

much better than n^2 for large databases!

- rewriting rules for algebra queries

$$\pi_{drinker}(\sigma_{beer = Bass}(\text{frequents} \bowtie \text{serves}))$$

$$\pi_{drinker}[\text{frequents} \bowtie \pi_{bar}(\sigma_{beer = Bass}(\text{serves}))]$$

- efficient algorithms for individual operations

Indexes: special "directories" to data

cost: roughly n (log n)

much better than n^2 for large databases!

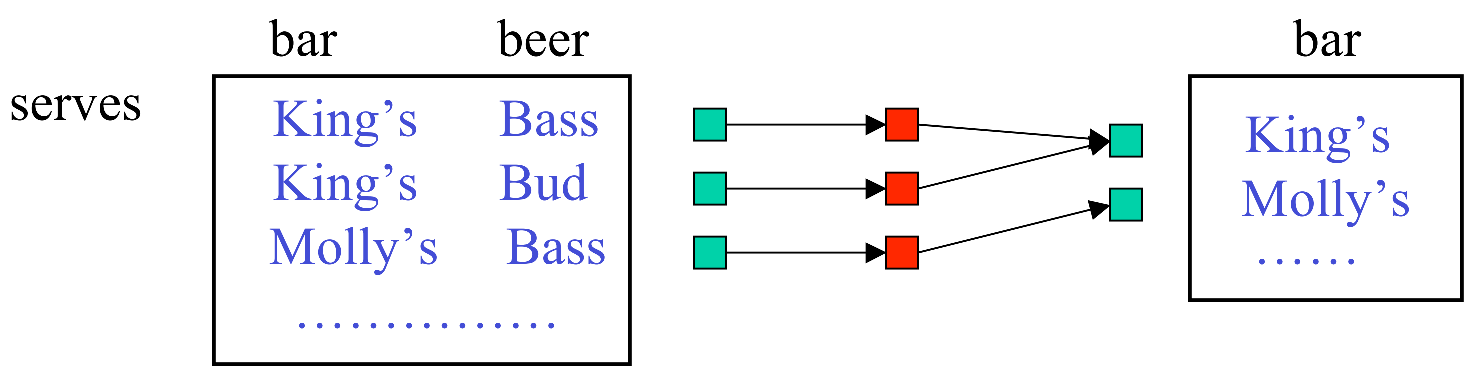
Most spectacular: theoretical potential for perfect scaling!

- perfect scaling: given sufficient resources, performance does not degrade as the database becomes larger
- key: parallel processing
- cost: number of processors polynomial in the size of the database
- role of algebra: operations highlight parallelism

Each algebra operation can in principle be implemented very efficiently in parallel

Example: projection

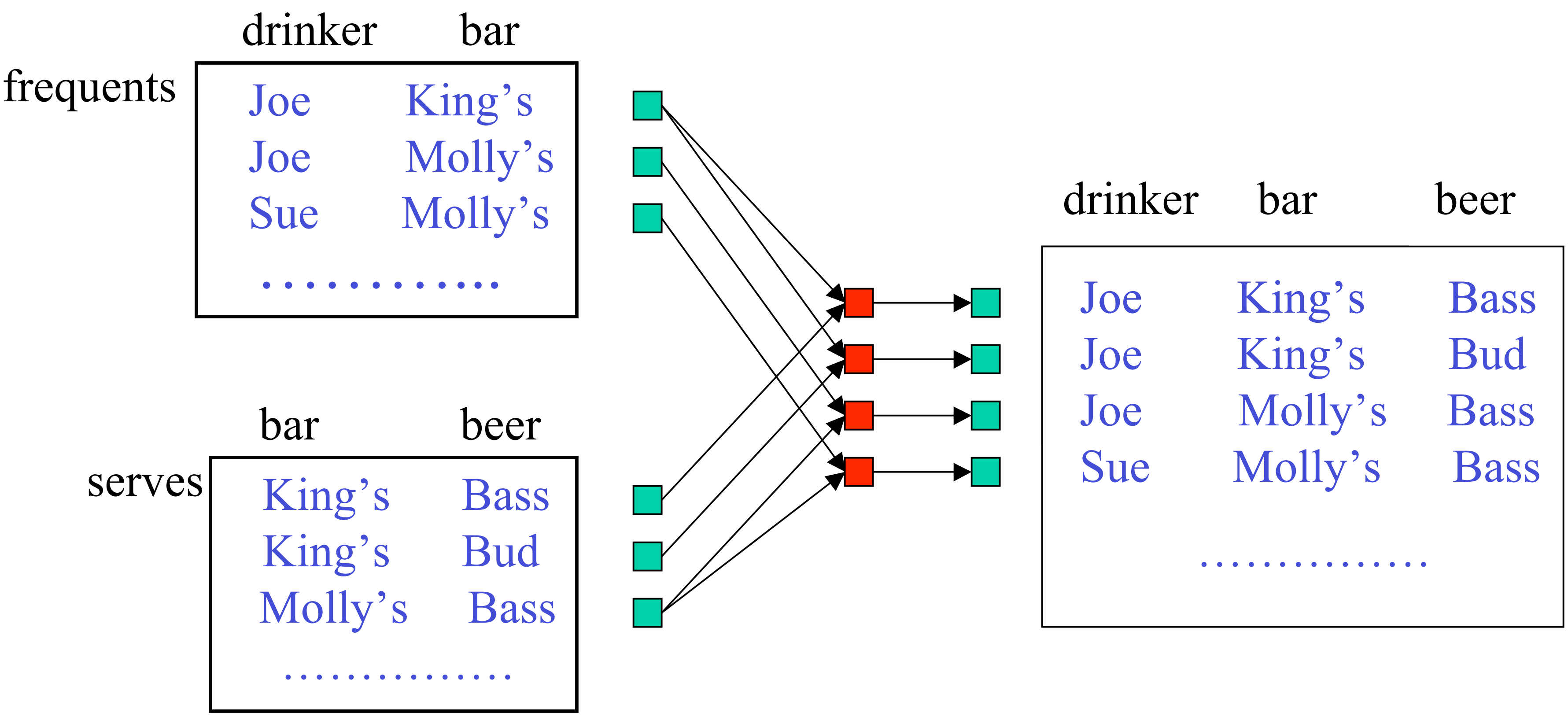
π_{bar} (serves)



Constant parallel time!

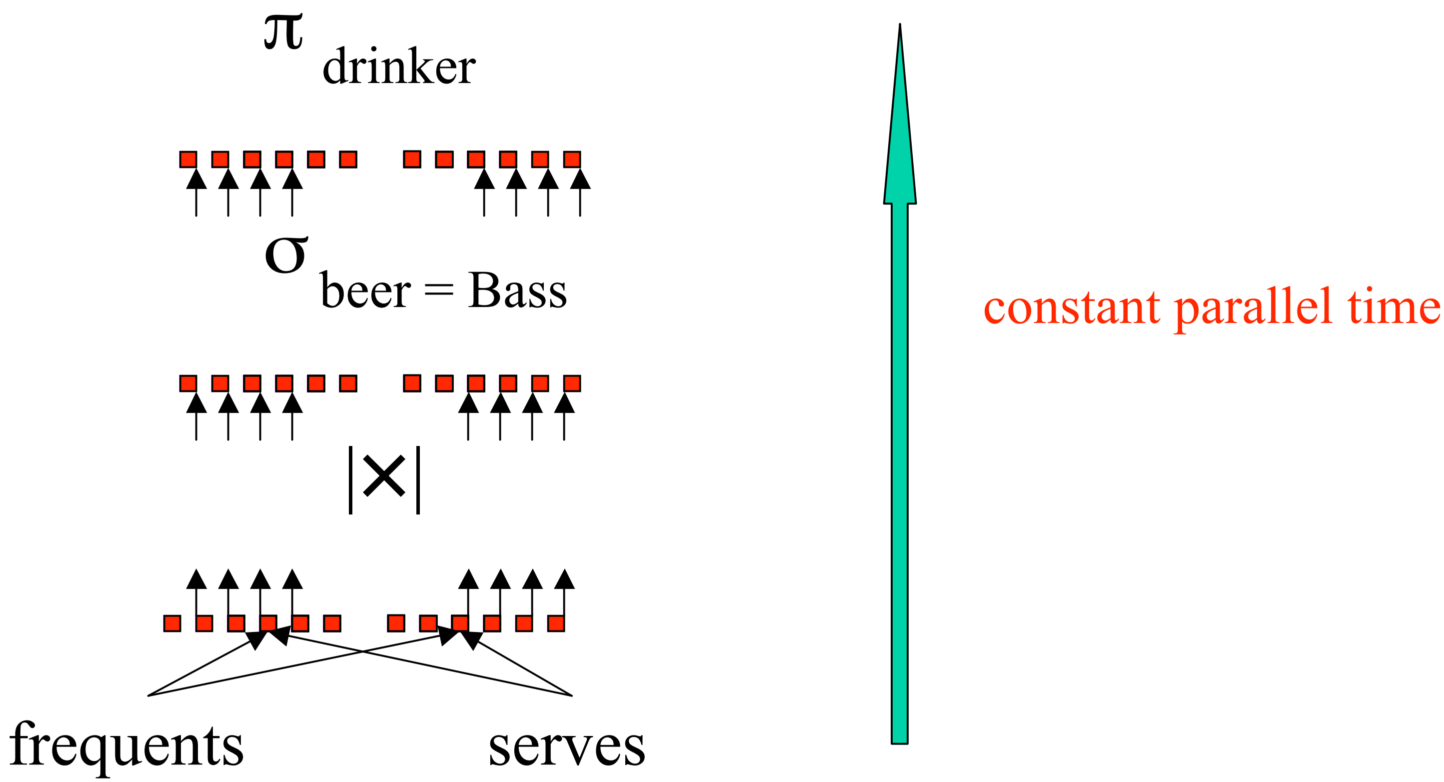
Another example: join

frequents |X| serves



Every relational algebra query takes constant parallel time!

$$\pi_{drinker}(\sigma_{beer = Bass}(\text{frequents} \bowtie \text{serves}))$$



Summary so far:

- Keys to the success of FO as a query language:

  -ease of use

  -efficient implementation via relational algebra

- Constant parallel complexity:

  the full potential of FO as a query language remains yet to be realized!

Beyond relational databases: the Web and XML

- relations replaced by trees (XML data)
- structure described by schemas (e.g., DTDs)

Again, logic provides the foundations:

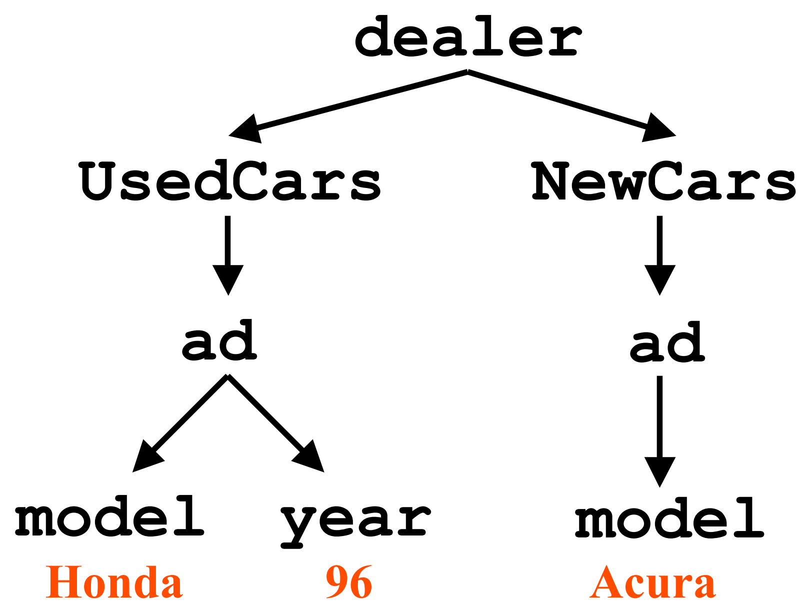
- DTDs are equivalent to tree automata (MSO on trees)
- XML queries are essentially tree transducers
- Can use automata and logic to understand semantics and expressiveness, perform static analysis

Most XML query languages are extensions of SQL

- implementation based on same paradigm
- uses extensions of relational algebra
- query optimization builds upon relational techniques

XML and DTDs

```
<dealer>
 <UsedCars>
  <ad>
   <model>Honda</model>
   <year>96</year>
  </ad>
 </UsedCars>
 <NewCars>
  <ad>
   <model>Acura</model>
  </ad>
 </NewCars>
</dealer>
```



Data Type Definition (DTD)

Σ: alphabet of element names, root ∈ Σ

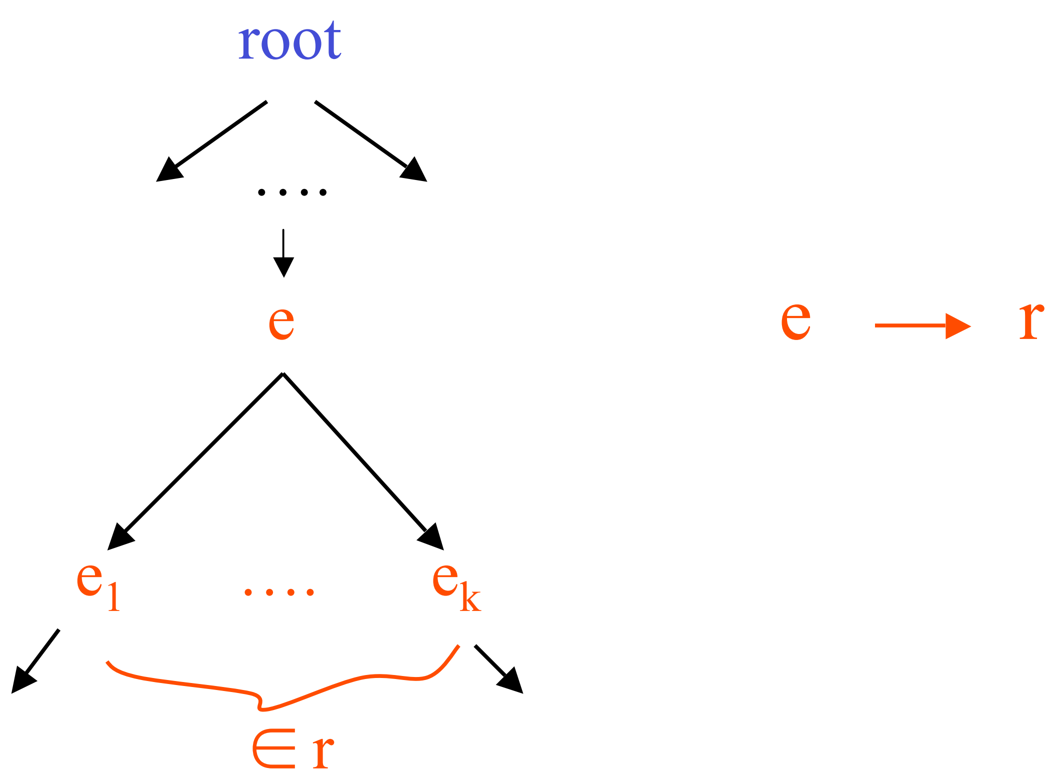
set of rules:

$$e \longrightarrow r$$

element name

regular expression over Σ

Documents satisfying a DTD

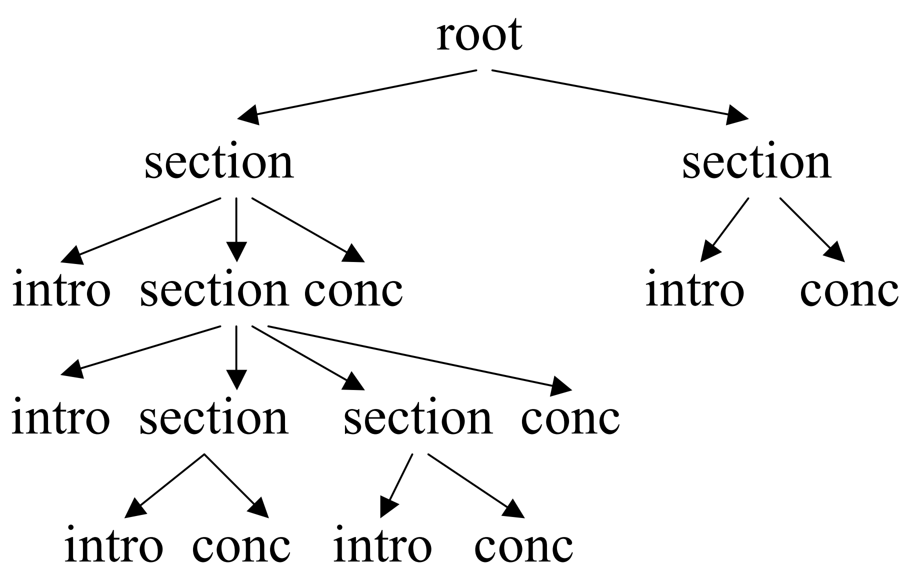


Set of trees satisfying DTD d: **T(d)**

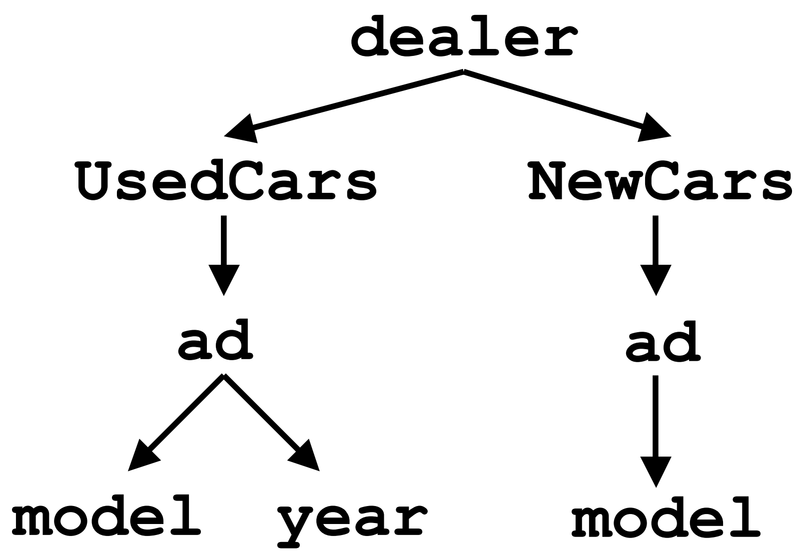
Example

A DTD and a tree satisfying it:

```
root → section*;
section → intro, section*,conclusions;
```

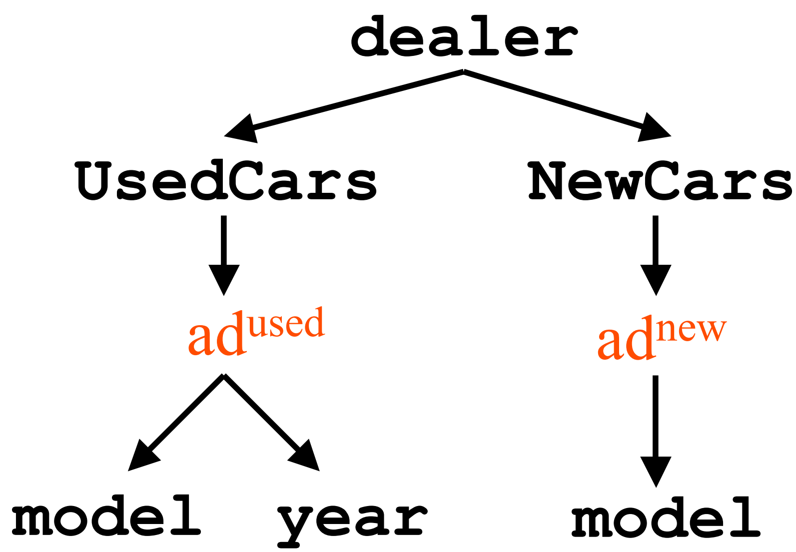


Specialization



ad has different structure in different contexts

Specialization



ad has different structure in different contexts

- What sets of trees can be defined?

Exactly the regular tree languages!

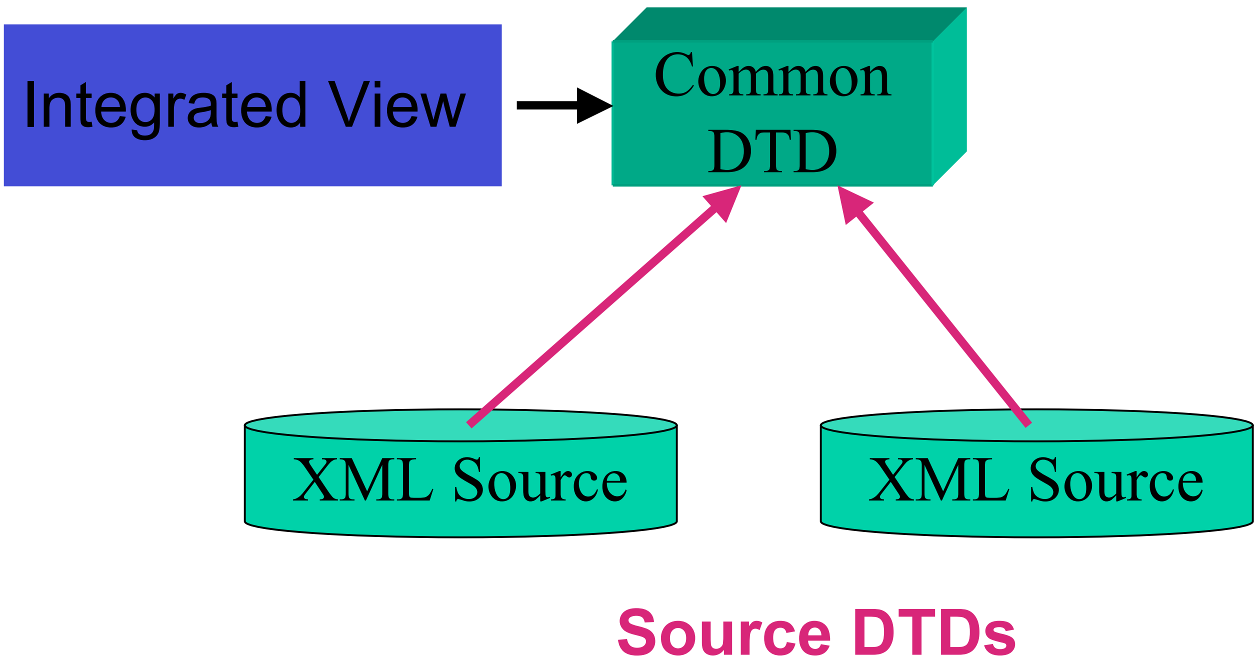
--trees accepted by tree automata
--trees defined by Monadic Second-Order Logic (MSO)

- XML query languages are essentially tree transducers

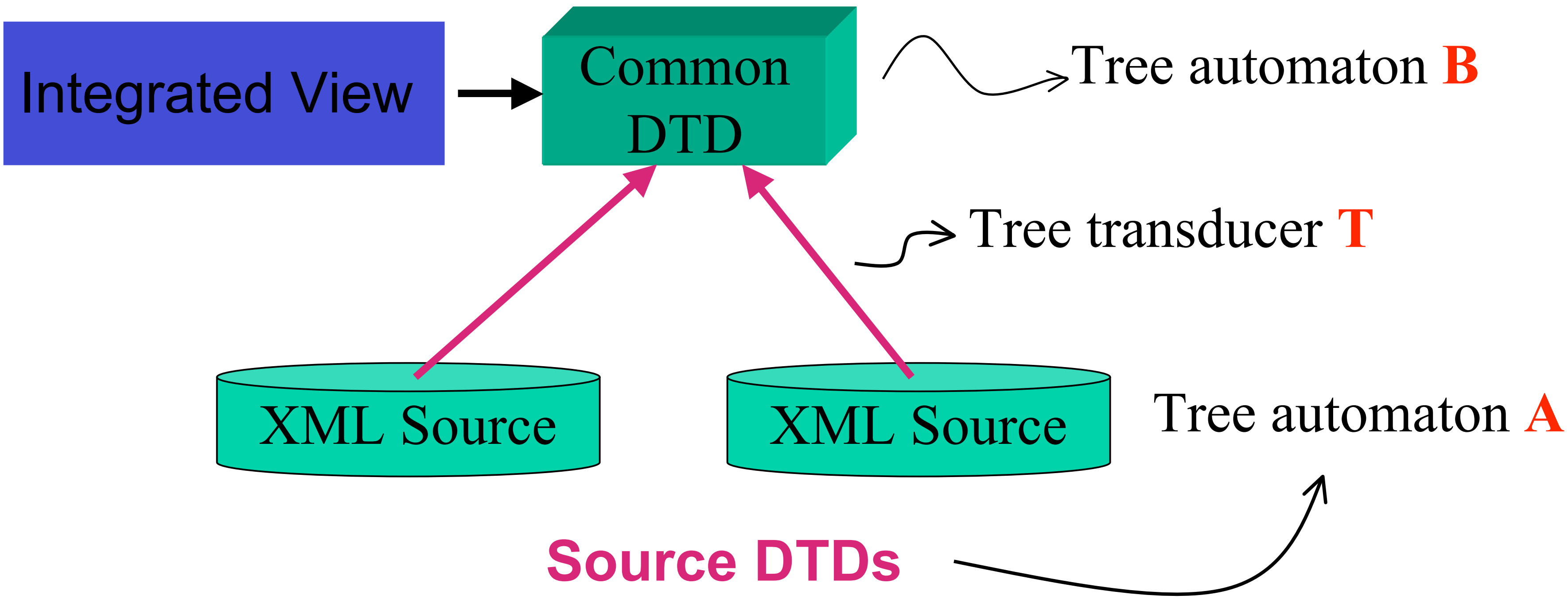
- Consequences:

can use automata/logic techniques to analyze
and manipulate DTDs and XML queries

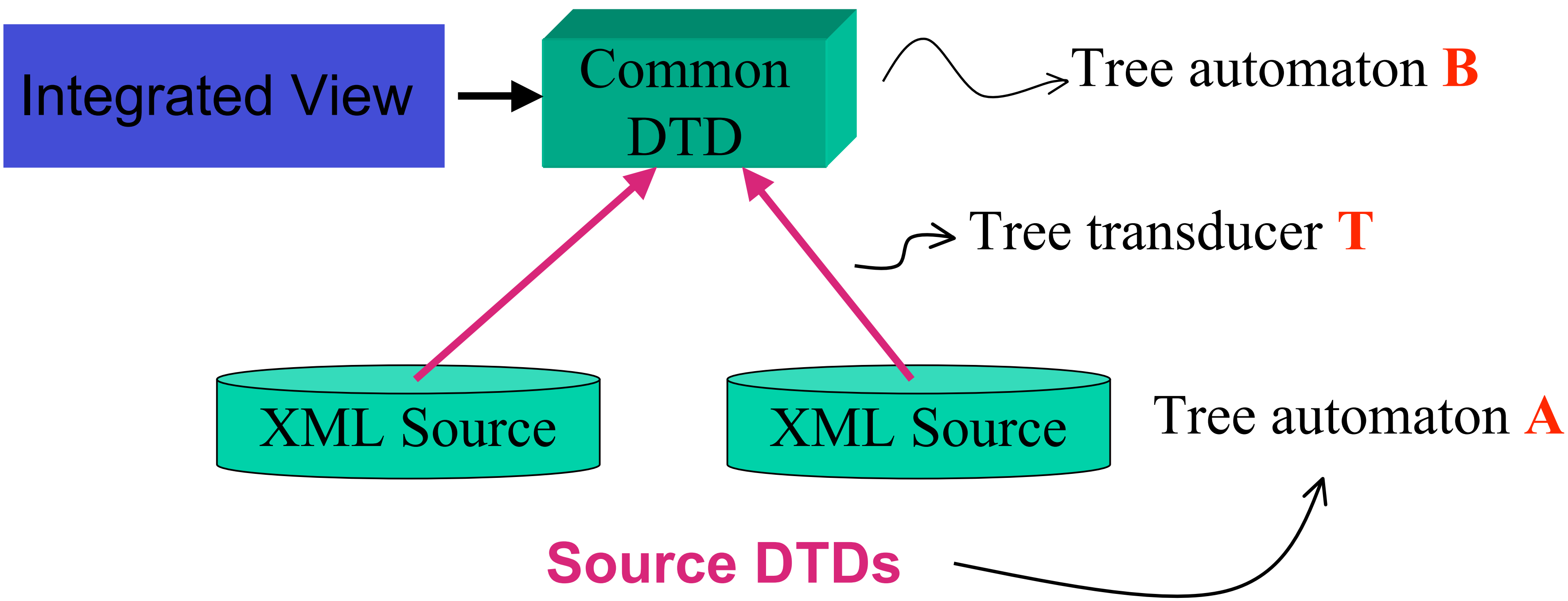
Example: static analysis for robust data integration



Example: static analysis for robust data integration



Example: static analysis for robust data integration



Need to check: **T(A) ⊆ B**

Key: **T^{-1}(B)** definable in MSO

Conclusion

- Logic has provided the foundations of databases, from relational databases all the way to XML
- FO lies at the core of relational database systems
- XML and its query languages are founded upon tree automata, tree transducers, and logics on trees
- Implementation uses extensions of relational algebra and builds upon relational database techniques