PatManQL: A language to manipulate patterns and data in hierarchical catalogs

Panagiotis Bouros, Theodore Dalamagas, Timos Sellis, Manolis Terrovitis

Knowledge and Database Systems Lab
School of Electrical and Computer Engineering
National Technical University of Athens
{pbour,dalamag,timos,mter}@dblab.ece.ntua.gr

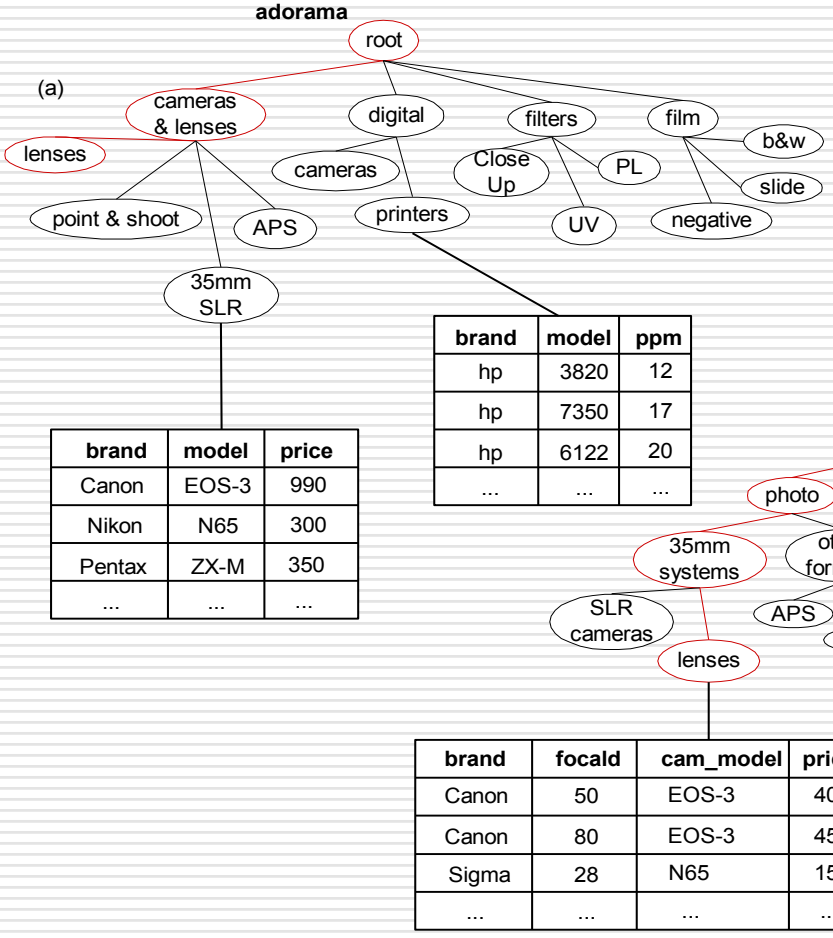
Outline

- Introduction
- Contribution
- Structures
- Operators
- Prototype
- Related work
- Conclusion

Introduction

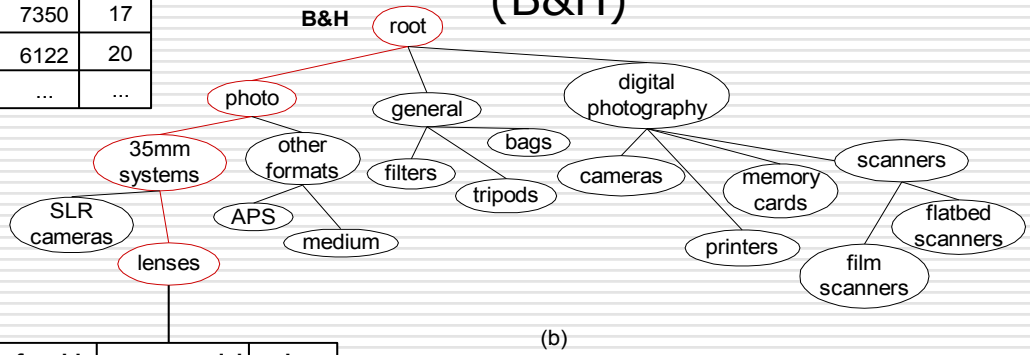
- Huge volumes of data on the Web
- Hierarchical structures and catalogs
- Paths → **knowledge artifacts**
  - Represent group of data
    - ⇨ Conceptual clustering of raw data based on common properties
  - Semantic guides
- Example: **Portal catalogs**

Introduction



brand	model	price
Canon	EOS-3	990
Nikon	N65	300
Pentax	ZX-M	350
...	...	...

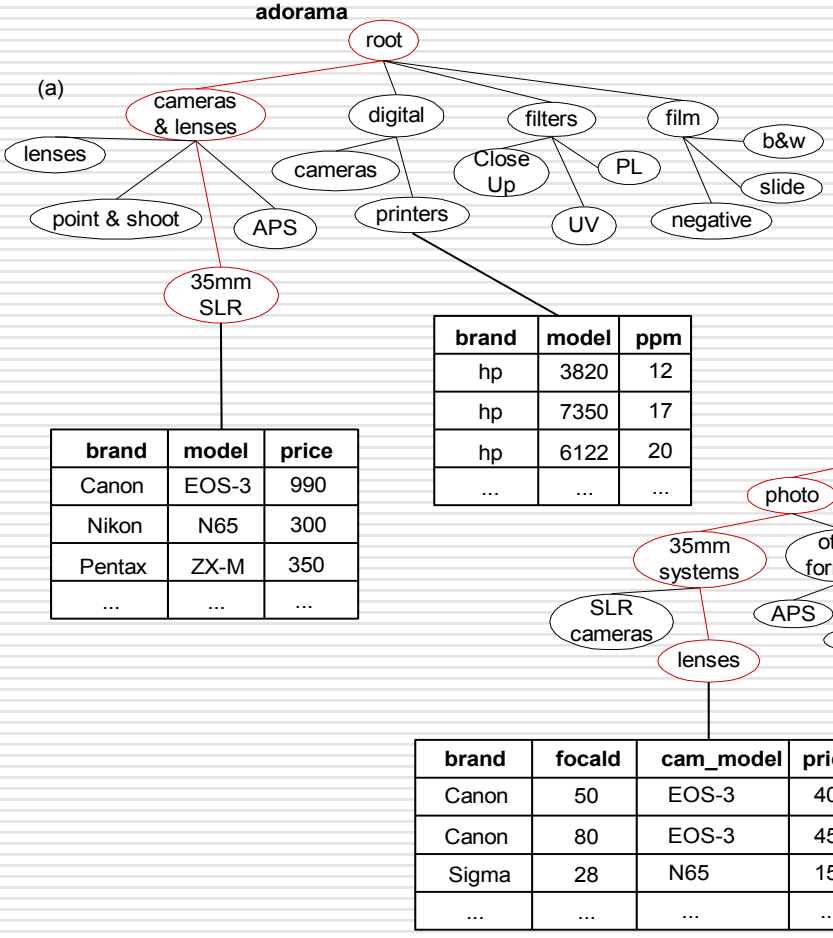
brand	model	ppm
hp	3820	12
hp	7350	17
hp	6122	20
...	...	...



brand	focald	cam_model	price
Canon	50	EOS-3	400
Canon	80	EOS-3	450
Sigma	28	N65	150
...	...	...	...

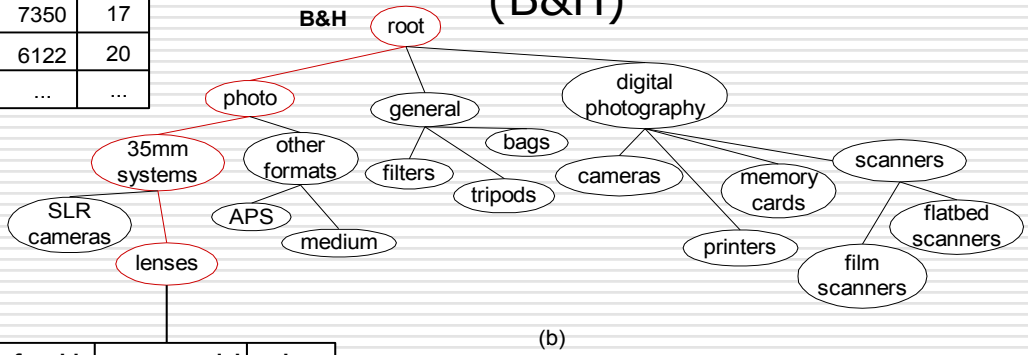
- Paths → **alternative pattern versions** for the same group of data
- Example: searching for lenses
  - /cameras & lenses/lenses (adorama)
  - /photo/35mm systems/lenses (B&H)

Introduction



brand	model	price
Canon	EOS-3	990
Nikon	N65	300
Pentax	ZX-M	350
...	...	...

brand	model	ppm
hp	3820	12
hp	7350	17
hp	6122	20
...	...	...



brand	focald	cam_model	price
Canon	50	EOS-3	400
Canon	80	EOS-3	450
Sigma	28	N65	150
...	...	...	...

- Paths → **complex pattern**
- Example: searching for integrated photo systems
  - /cameras & lenses/35mm SLR (adorama)
  - /photo/35mm systems/lenses (B&H)

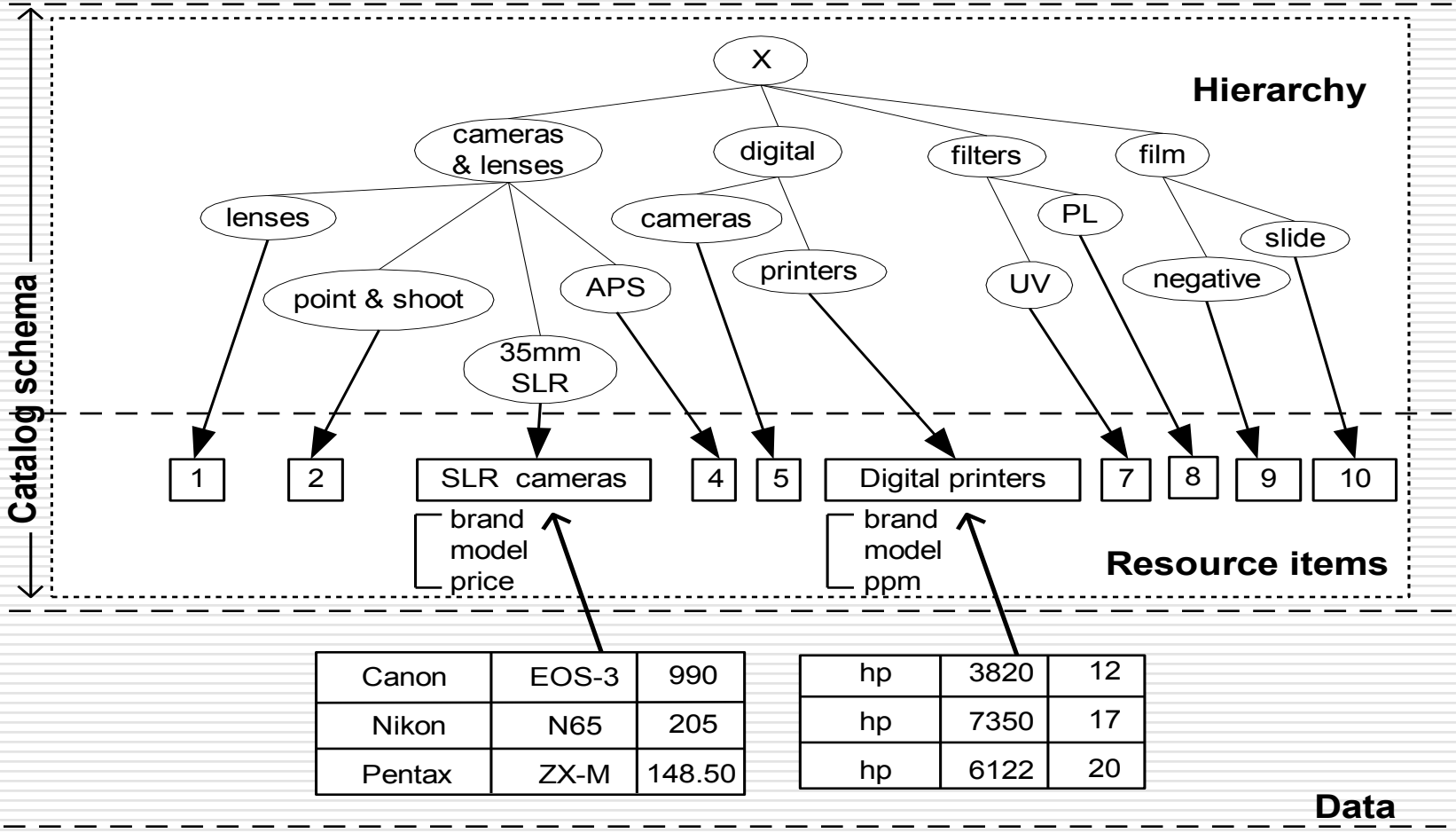
Contribution

- A model to represent paths as knowledge artifacts
- The **PatManQL** language:
  - Operators to manipulate path-like patterns
  - Relational operators for data
- A prototype

Catalog Schema

- A tree with:
  - a root (⊗)
  - a set of non-leaf nodes (○)
  - a set of **resource items** as leaves (□)
- Data: instances (records) of resource item
  - Resource item: **Relation** R(a1, a2, …, an), where a1, a2, … **attributes**

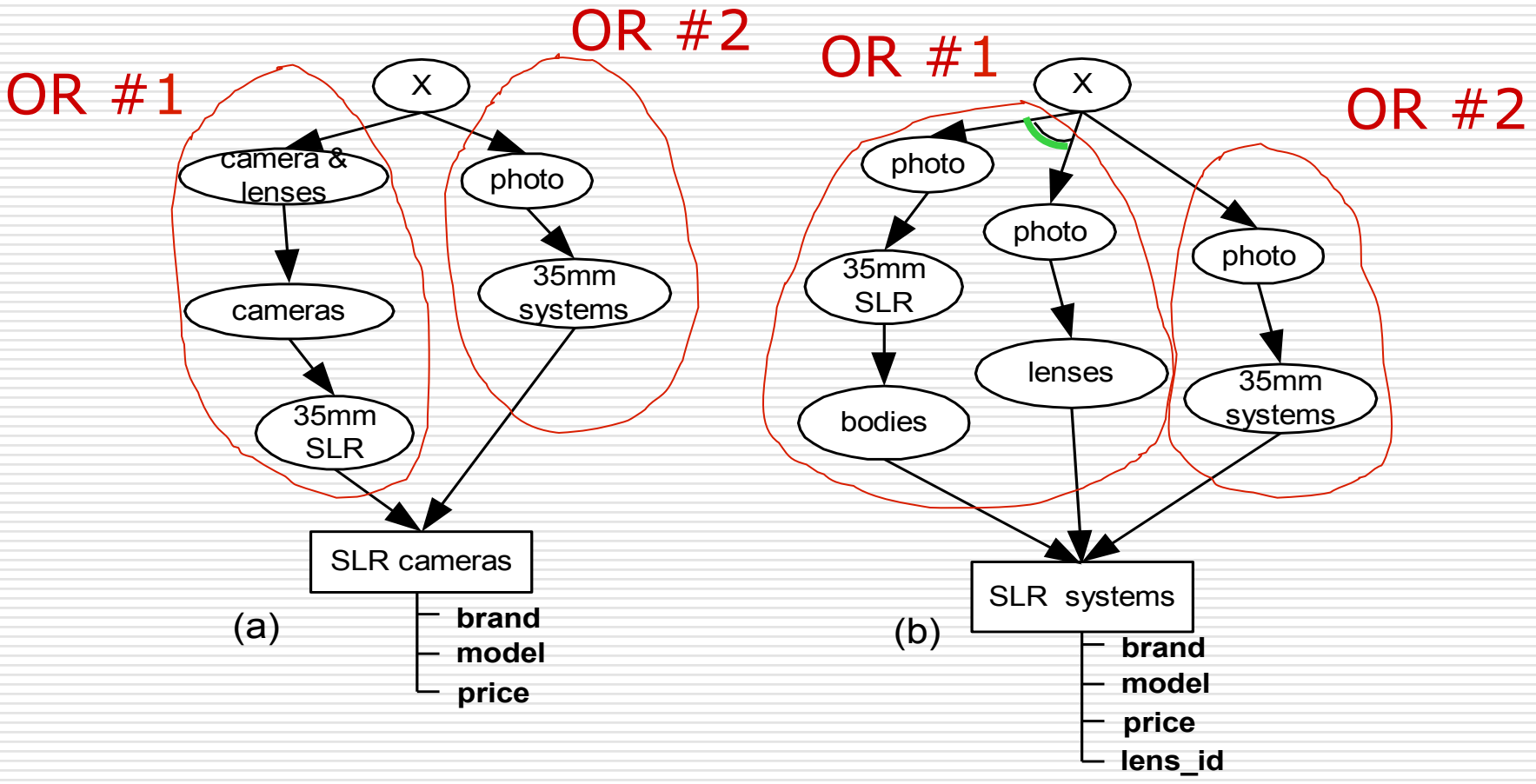
Catalog Schema



Tree-Structure Relations (TSRs)

- Combining catalog schemas with common resource item
- Tree-Structure Relation (AND/OR-like graph):
  - **One** resource item
  - Paths organized in **OR components**
    - OR component: group of one or more paths (AND group)
    - OR components are alternative ways to access the common resource item
  - **Paths = patterns**

Tree-Structure Relations (TSRs)



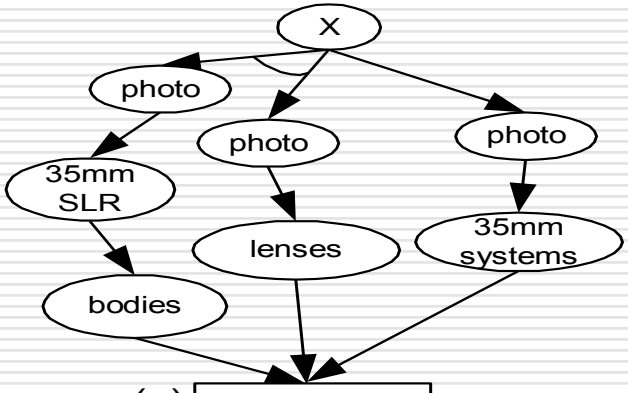
Operators

- **Select** (σ)
  - $\sigma_{\text{<attribute condition><path condition>}}$ (TSR)
    - ⇨ attribute condition: {=, ≠, <}
    - ⇨ path condition: {=, ≠, ⊂, ∠}
  - Filters instances of resource items and OR components

Select example

'Select all non Pentax cameras with price greater than 200Euros, having "/photo/35mm systems" in their paths':

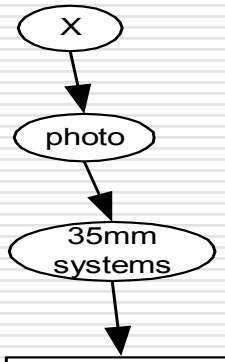
$\sigma_{<brand !="Pentax", price > 200><"/photo/35mm systems" \subset \$\_>}$(SLR systems)



(a)

SLR systems

brand	model	price	lens_id
Canon	EOS-3	990	1
Nikon	N65	205	2
Pentax	ZX-M	148.5	3
...	...	...	...



(b)

SLR systems

brand	model	price	lens_id
Canon	EOS-3	990	1
Nikon	N65	205	2
...	...	...	...

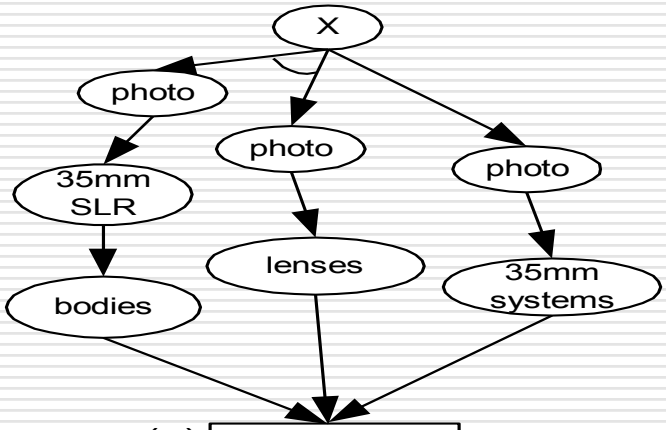
Operators

- **Project** (π)
  - $\pi_{\text{<attribute list><variable list>}}$ (TSR)
    - ⇨ attribute list: {attribute}
    - ⇨ variable list: {$i (path variable), #i (OR variable)}
  - Keeps attributes of resource item and paths of each OR component or OR components on the whole

Project example

'Cameras with only the model and lens_id attributes and the rightmost component':

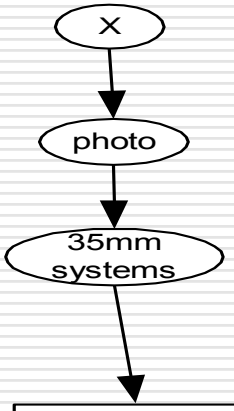
$$\Pi_{<model,\ lens\_id><\#2>}(\text{SLR systems})$$



(a)

SLR systems

brand	model	price	lens_id
Canon	EOS-3	990	1
Nikon	N65	205	2
Pentax	ZX-M	148.5	2
...	...	...	...



(b)

SLR systems

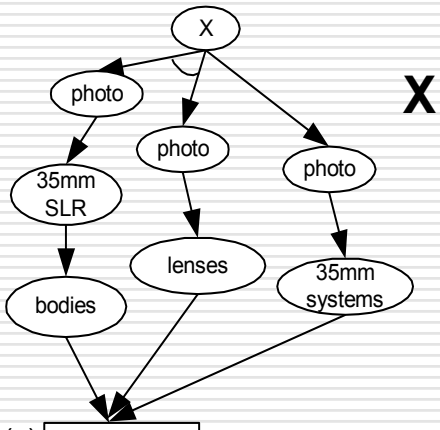
model	lens_id
EOS-3	1
N65	2
ZX-M	2
... ...	...

Operators

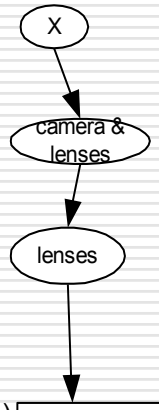
- **Cartesian product** (X)
  - (TSR1) X (TSR2)
  - Combine instances of resources and OR components

Cartesian product example

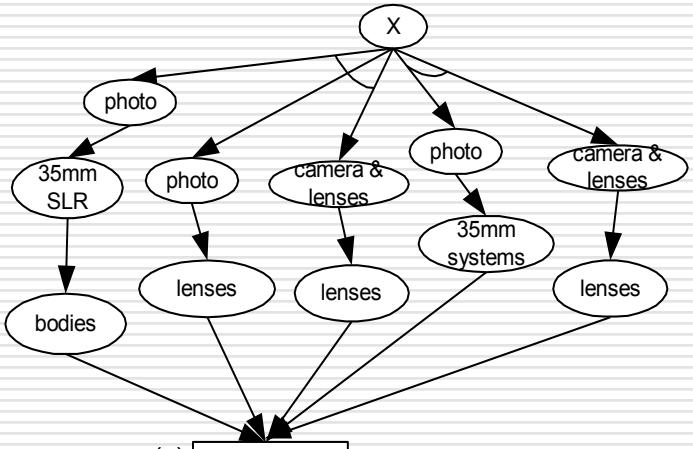
(SLR systems) X (Lenses)



X



=



(a) SLR systems

cbrand	cmodel	cprice	clensid
Canon	EOS-3	990	1
Nikon	N65	205	1
Pentax	ZX-M	148.5	2
...	...	...	...

(b) Lenses

lbrand	lensid	lprice
Sigma	1	200
Tamron	2	100
...	...	...

(c) SLR systems

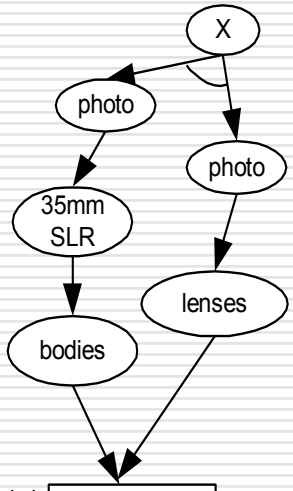
cbrand	cmodel	cprice	clensid	lbrand	lensid	lprice
Canon	EOS-3	990	1	Sigma	1	200
Canon	EOS-3	990	1	Tamron	2	100
Nikon	N65	205	1	Sigma	1	200
Nikon	N65	205	1	Tamron	2	100
Pentax	ZX-M	148.5	2	Sigma	1	200
Pentax	ZX-M	148.5	2	Tamron	2	100
...	...	...	...	...	...	...

Operators

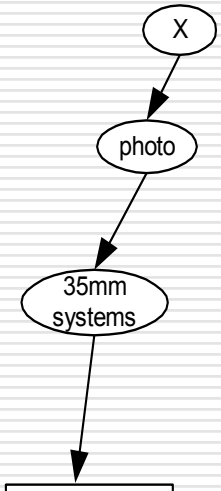
- **Union** (U)
  - (TSR) U (TSR)
  - Union of instances and all OR components
- **Intersection** (∩)
  - (TSR) ∩ (TSR)
  - Intersection of instances and all OR components
- **Difference** (–)
  - (TSR) – (TSR)
  - Instances of the first TSR not present in the second one and all OR components of the first TSR

Union example

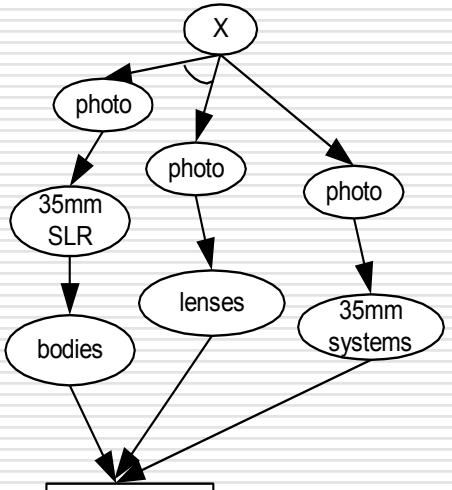
(SLR systems) U (SLR systems)



U



=



(a) SLR systems

cbrand	cmodel	cprice	clensid
Canon	EOS-3	990	1
Nikon	N65	205	1
Pentax	ZX-M	148.5	2
...	...	...	...

(b) SLR systems

cbrand	cmodel	cprice	clensid
Canon	EOS-3	990	1
Nikon	FM2	800	1
Pentax	ZX-M	148.5	2
...	...	...	...

(c) SLR systems

cbrand	cmodel	cprice	clensid
Canon	EOS-3	990	1
Nikon	N65	205	1
Pentax	ZX-M	148.5	2
Nikon	FM2	800	1
...	...	...	...

Prototype

- Interpreter
- Query Execution Engine
- Storage mechanism
  - XML files
  - MySQL RDBMS
    - ⇨ All-edges-in-one-table storage approach
- Graphical Interface

Related work

- Pattern management (PANDA project) (S. Rizzi et al.)
- Inductive databases framework (Tomasz Imielinski et al.)
  - DMQL (Jiawei Han et al.), MINE RULE(R.Meo et al.)
    - ⇨ Descriptive rules
- Tree algebras
  - TAX (H. V. Jagadish et al.)
    - ⇨ Selecting – reconstructing bulk XML data
  - YAT (V. Christophides et al.)
    - ⇨ Tuple-based, not tree-based

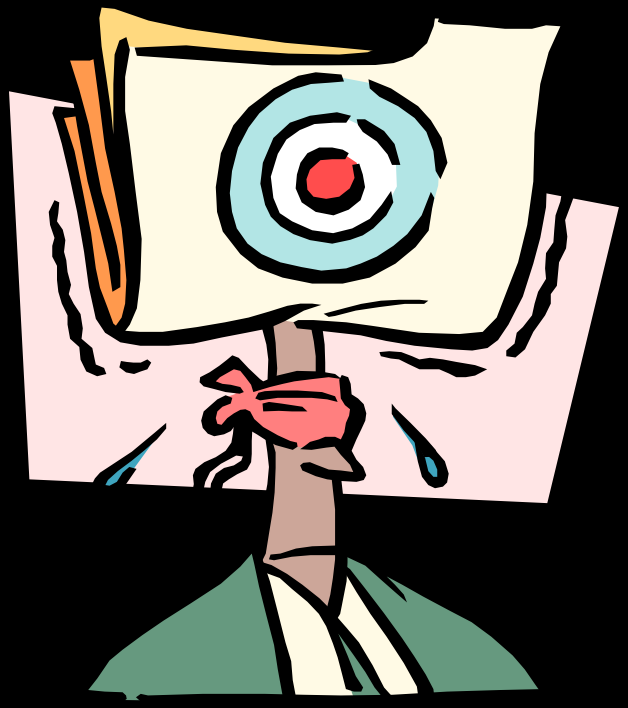
Conclusion

- A model to represent paths as knowledge artifacts (patterns)
  - Catalog schema
  - Tree-Structure Relations (TSRs)
- The PatManQL language:
  - Operators to manipulate paths as patterns and data
- A prototype system

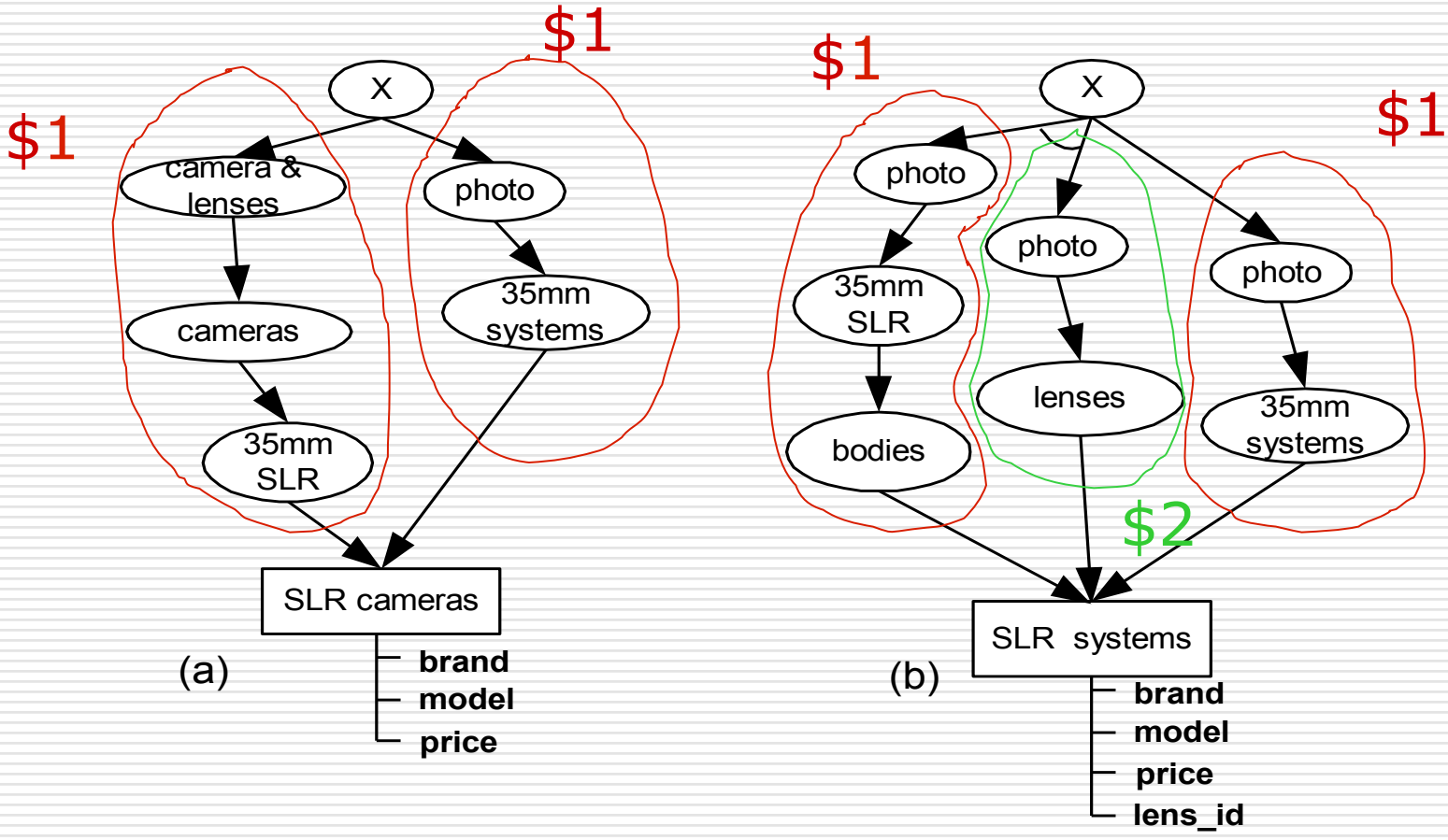
Future Work

- Properties of the Operators
- Restructure operators
- Join operator

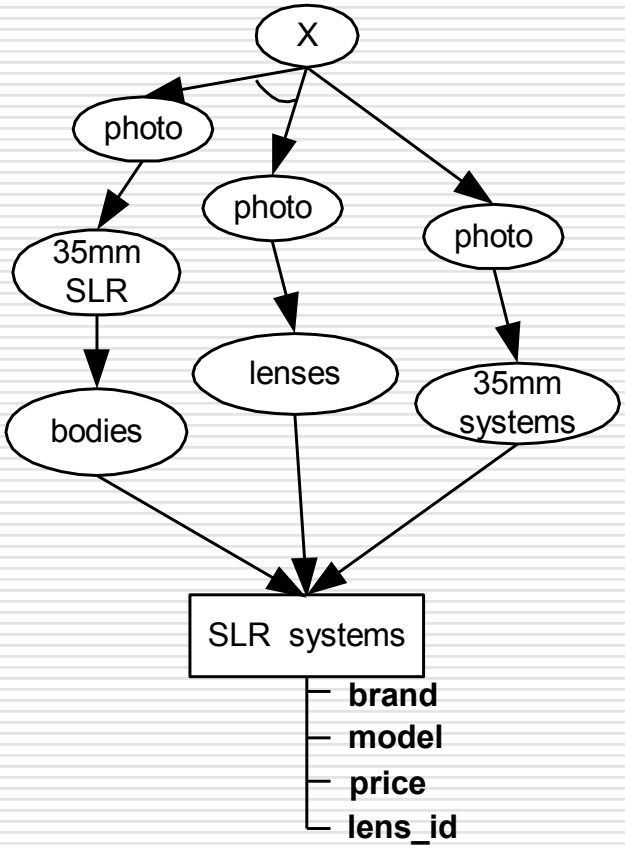
Questions (?)



Tree-Structure Relations (TSRs)



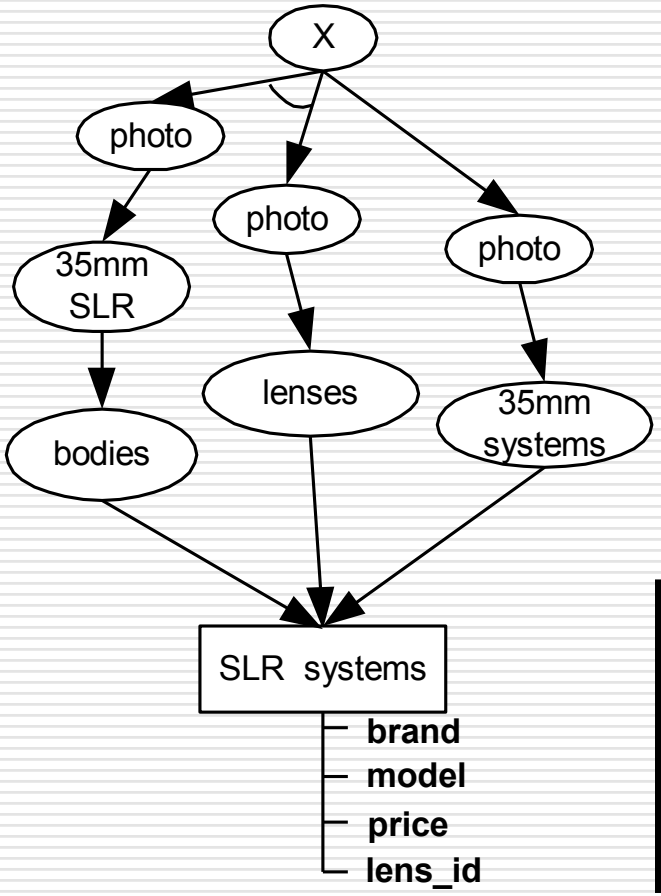
Storage mechanism



- XML file

```
<tsr name="SLR systems">
    <or>
        <and>/photo/35mm SLR/bodies</and>
        <and>/photo/lenses</and>
    </or>
    <or>
        <and>/photo/35mm systems</and>
    </or>
    <item>
        <attribute name="brand" type="…"/>
        <attribute name="model" type="…"/>
        …
        <tuple>…</tuple>
        …
    </item>
</tsr>
```

Storage mechanism



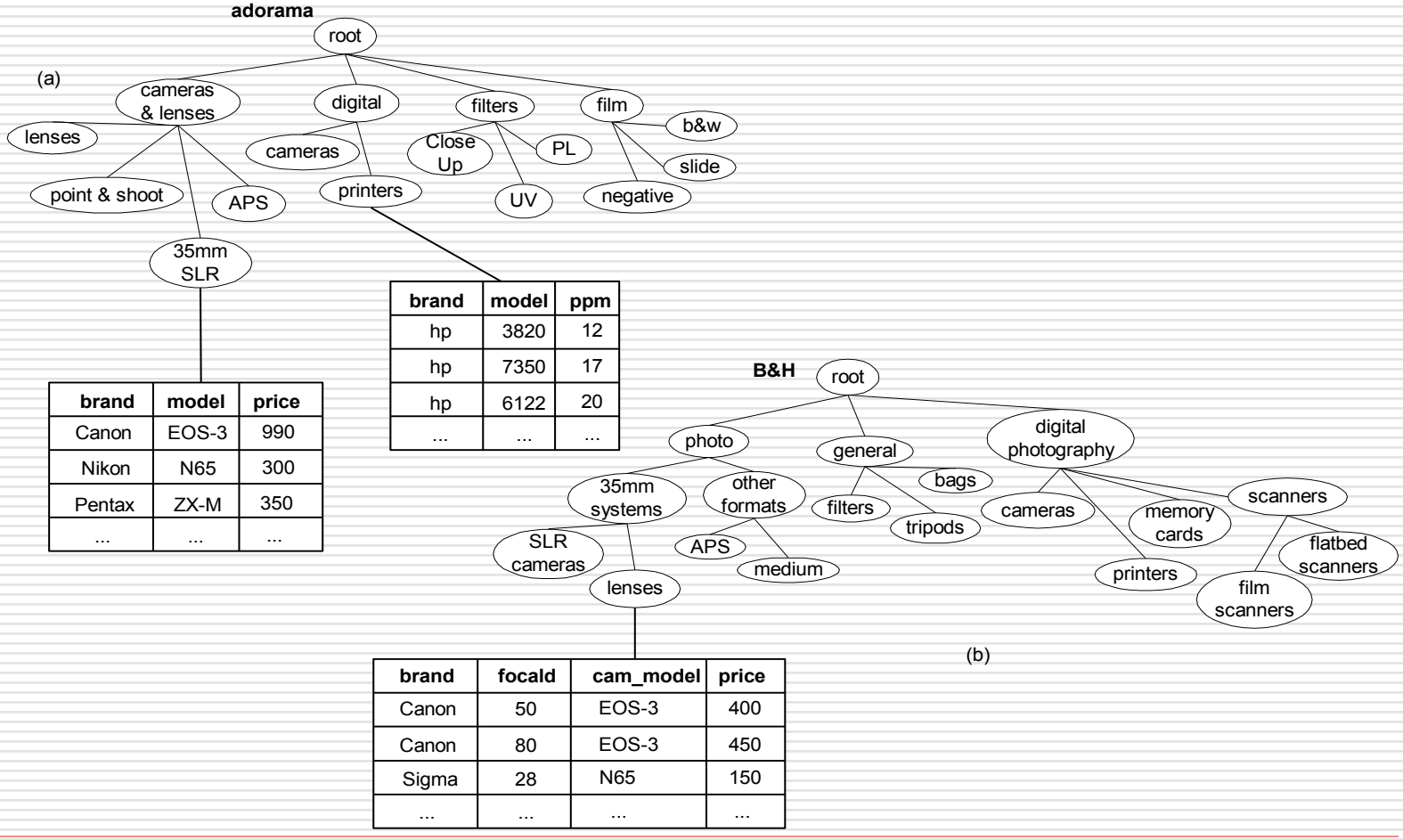
- Database

tid	name	file
1	SLR systems	portal.xml

brand	model	price	lens_id
...	...	...	...

tid	orid	andid	path
1	1	1	/photo/35mm SLR/bodies
1	1	2	/photo/lenses
1	2	1	/photo/35mm systems

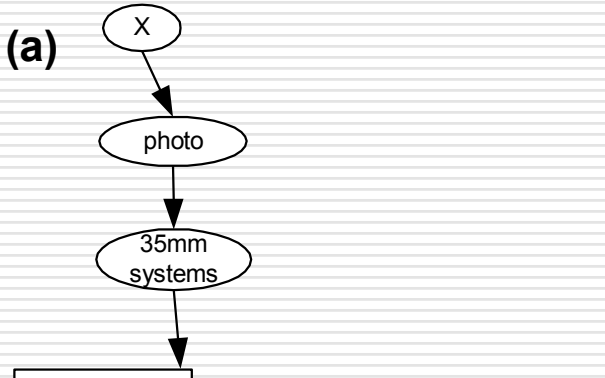
Catalog Schemas examples



Catalog Schema Manipulation

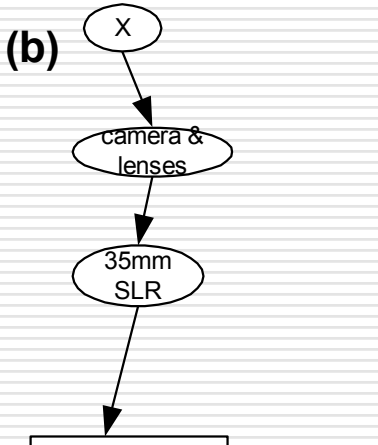
- SLR integrated systems from X – fig. (a)
- SLR cameras from Adorama – fig. (b)
- Lenses from B&H – fig. (c)
- Scenario for X:
  - New lenses out in the market
  - Lenses provided by B&H,
    that fit in Canon bodies provided by Adorama
  - Above SLR systems not present in her stock

Catalog Schema Manipulation



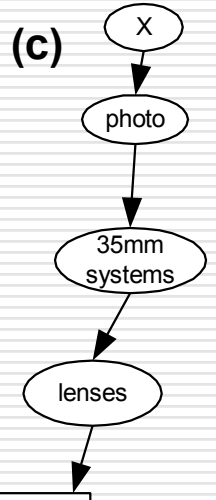
SLR systems

cbrand	cmodel	cprice	lmodel	lprice
Canon	EOS-3	990	100	390
Nikon	N65	205	340	160
...	...	...	...	...



SLR cameras

cbrand	cmodel	cprice
Canon	EOS-3	990
Nikon	N65	300
Pentax	ZX-M	350
...	...	...



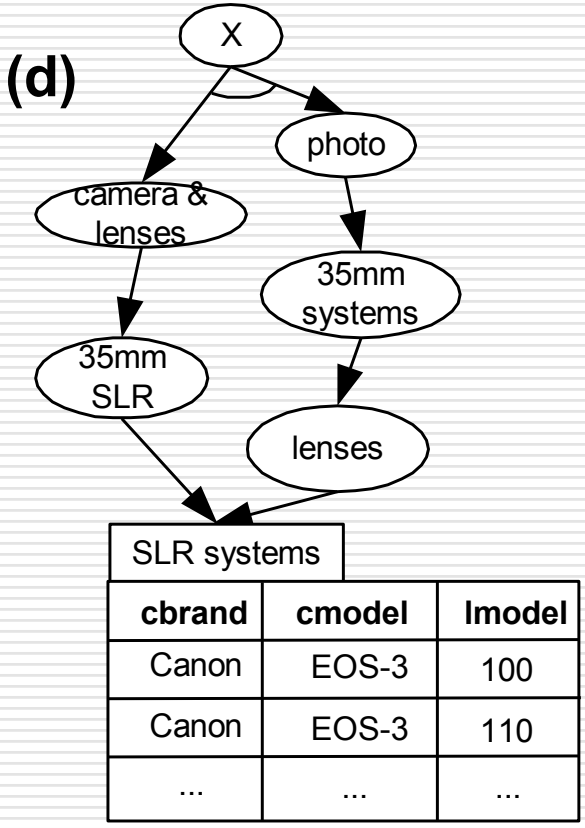
Lenses

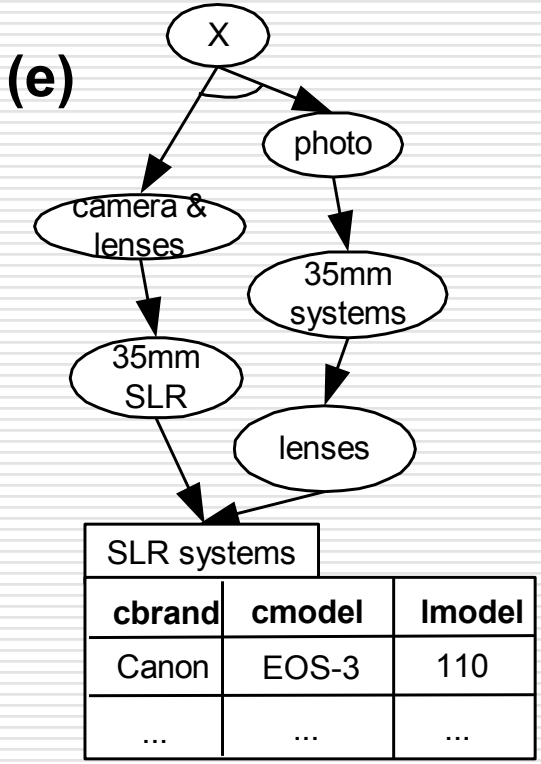
lbrand	lmodel	focald	cam_model	lprice
Canon	100	50	EOS-3	400
Canon	110	80	EOS-3	450
Sigma	340	28	N65	150
...	...	...	...	

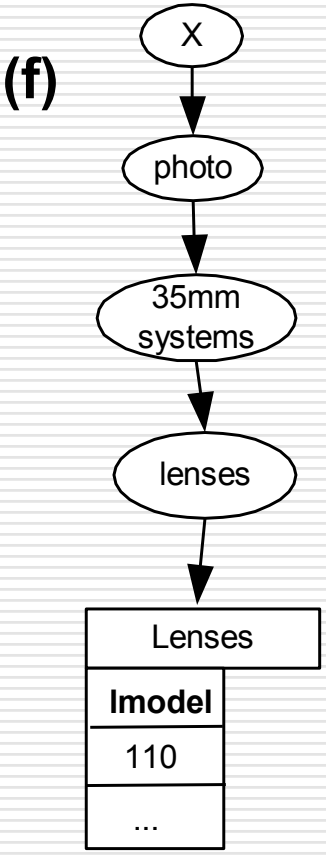
Catalog Schema Manipulation

- Systems with Canon bodies from Adorama and lenses from B&H – fig. (d):
  - $q1 = \Pi_{<cbrand,cmodel,lmodel><>}(\sigma_{<cmodel=cam\_model,\ cbrand="Canon"><>}((SLR\ cameras)\ X\ (lenses)))$
- Systems with Canon bodies from Adorama and lenses from B&H which are not in X's catalog – fig. (e):
  - $q2 = (q1) - \Pi_{<cbrand,cmodel,lmodel><>}(SLR\ cameras)$
- Lenses only without the appropriate camera bodies – fig. (f):
  - $\Pi_{<lmodel><\$2>}(q2)$

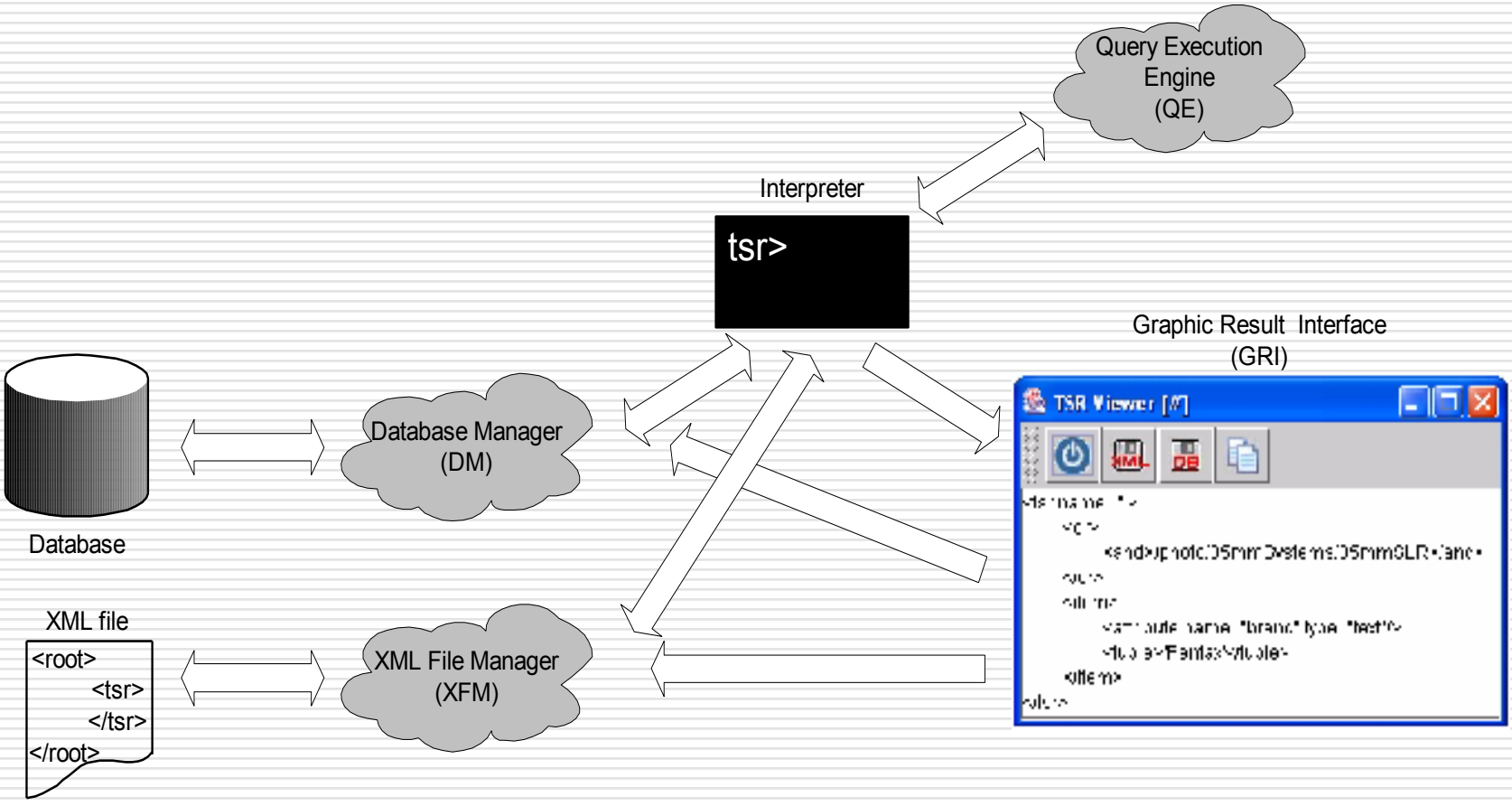
Catalog Schema Manipulation



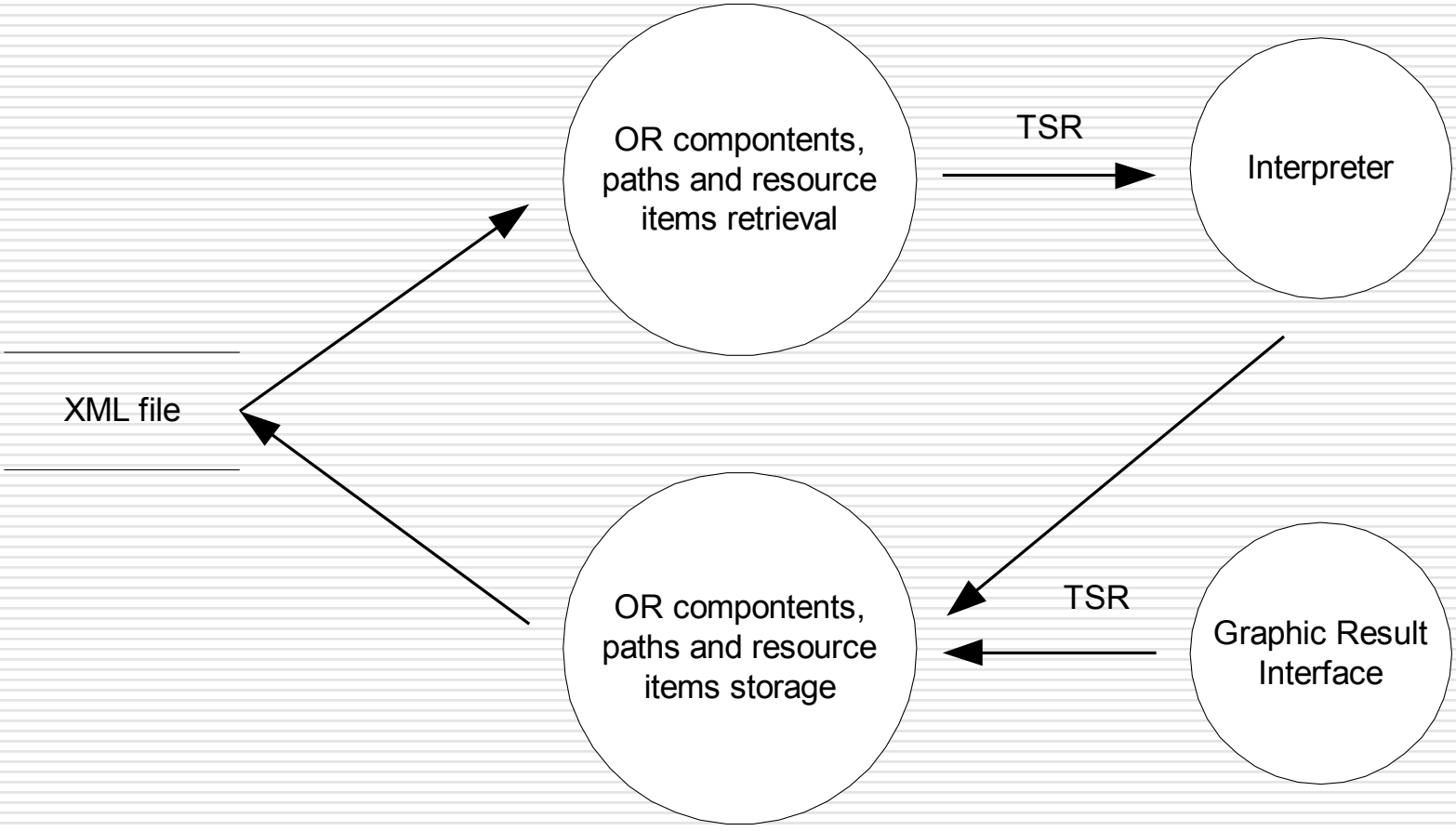




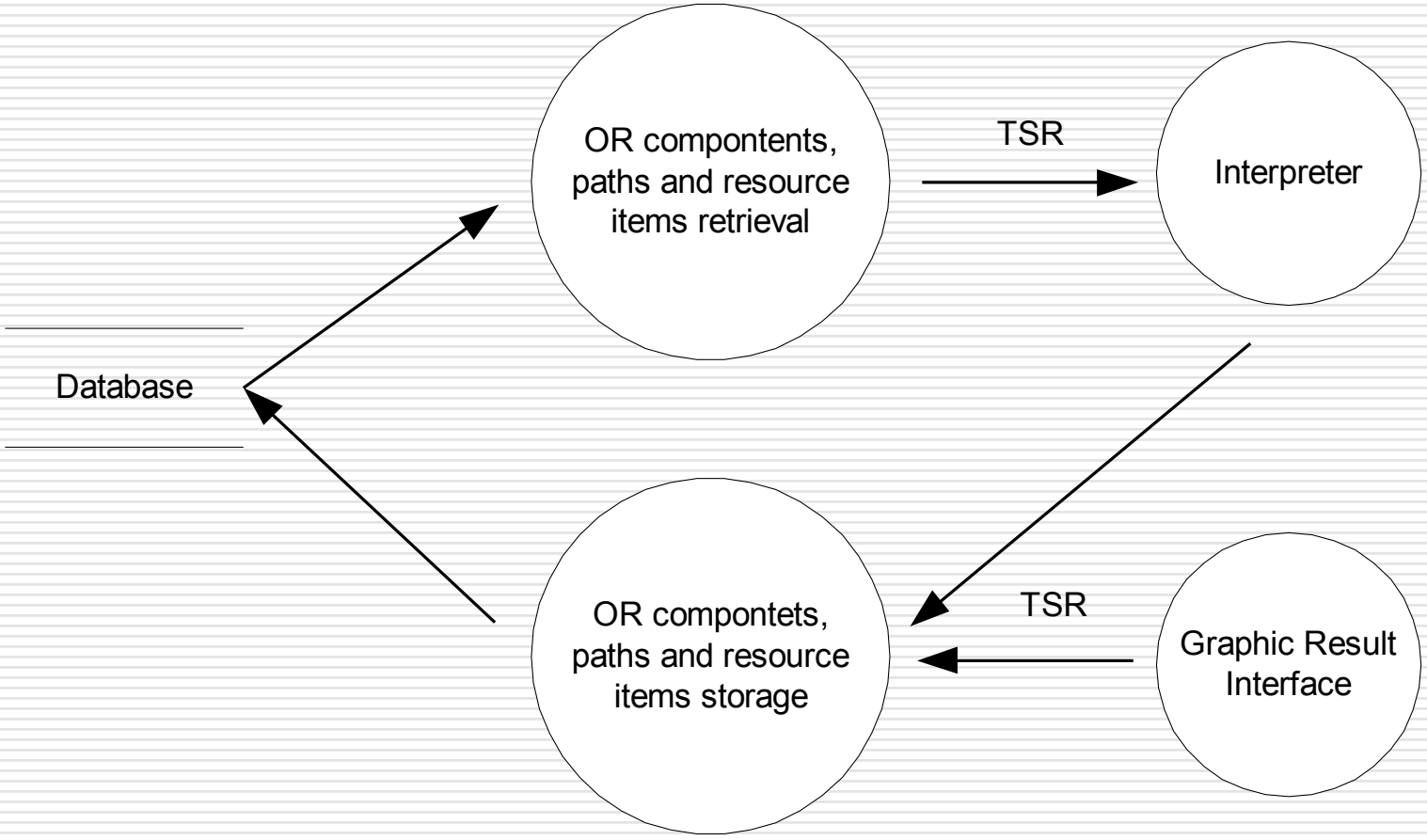
Prototype Architecture



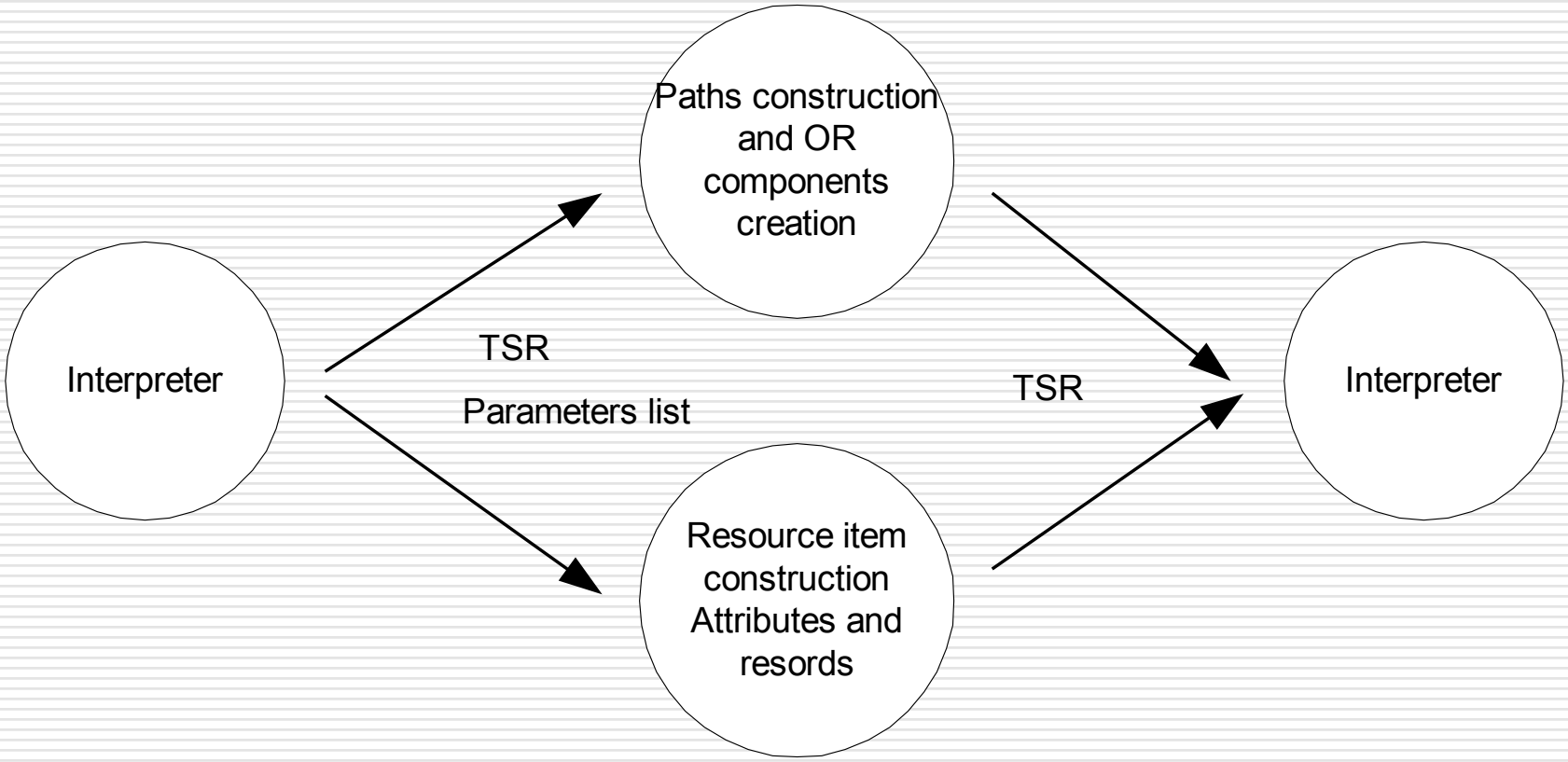
XML File Manager (XFM)



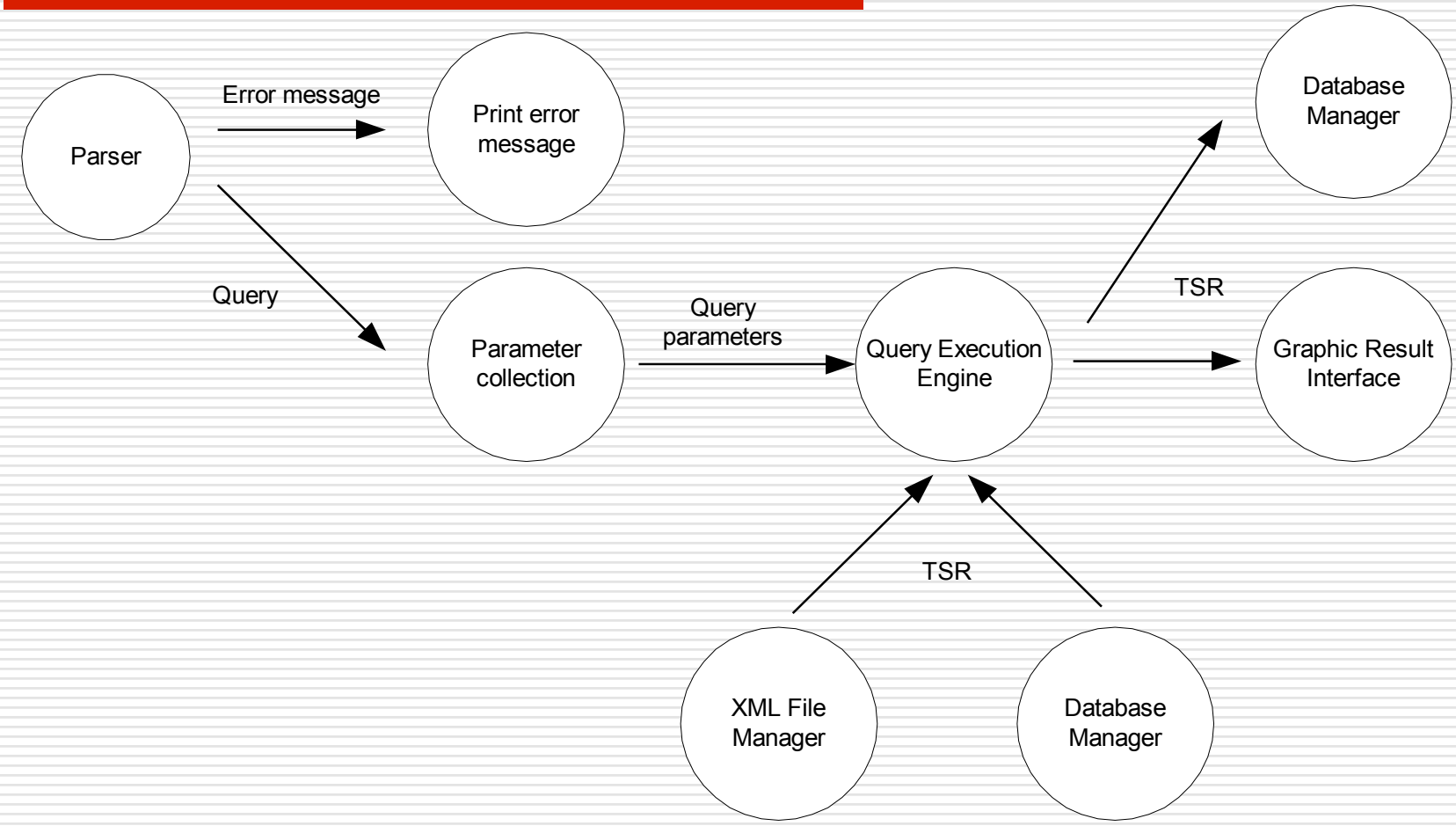
Database Manager (DM)



Query Execution Engine (QE)



Interpreter



Graphic Result Interface (GRI)