Semantic Web and Databases

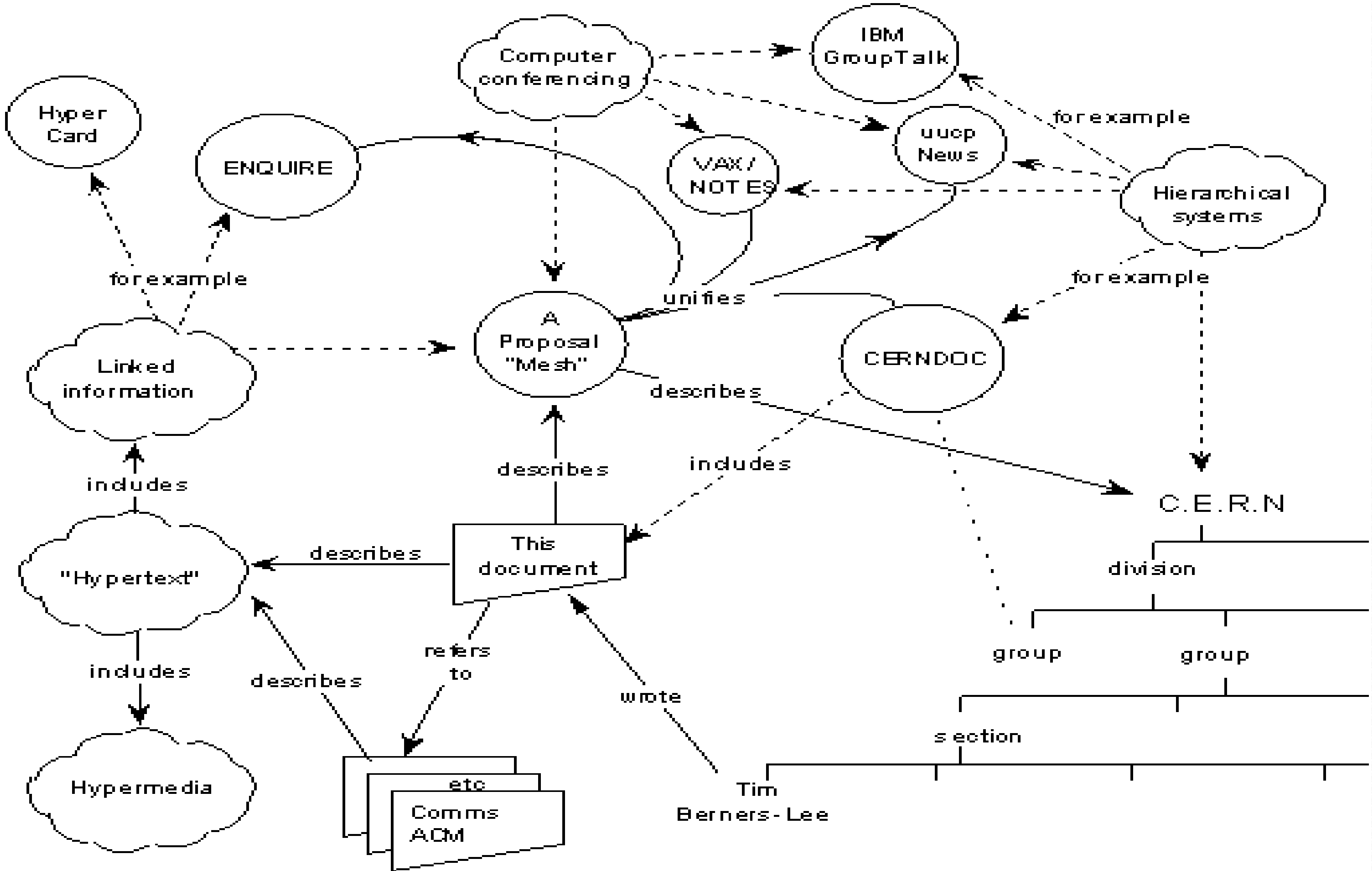
Vilas Wuwongse
Computer Science and
Information Management Program
School of Advanced Technologies
Asian Institute of Technology, Thailand
Email: vw@cs.ait.ac.th

Contents

- Semantic Web
- Ontology
- Ontology Languages
- Semantic Web and Databases
- XML Declarative Description (XDD)
- Semantic Web Modeling
- XML Database Modeling
- Conclusions

Semantic Web

History of the Semantic Web WWW (1989)



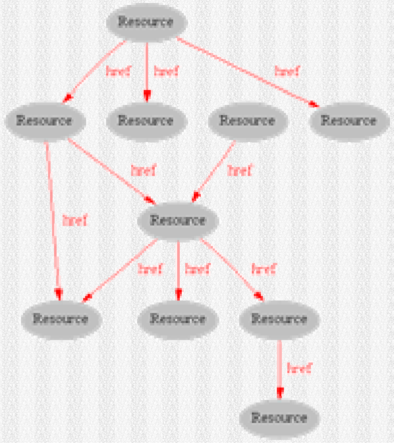
History of the Semantic Web

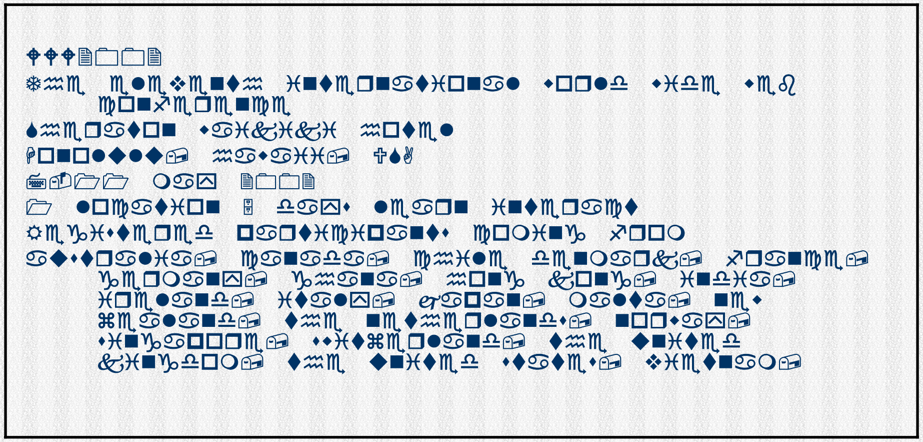
Tim Berners-Lee’s original vision of the Web:

“... a goal of the Web was that, if the interaction between person and hypertext could be so intuitive that the **machine-readable** information space gave an accurate representation of the state of people's thoughts, interactions, and work patterns, then **machine analysis** could become a very powerful management tool, seeing patterns in our work and facilitating our working together through the typical problems which beset the management of large organizations.”

Current Web

- A set of linked resources
- Each resource is identified by a URI
- A resource is understood and consumed by human users
- But, for machines, a resource is merely strings of 0's and 1's





Machine view of a resource

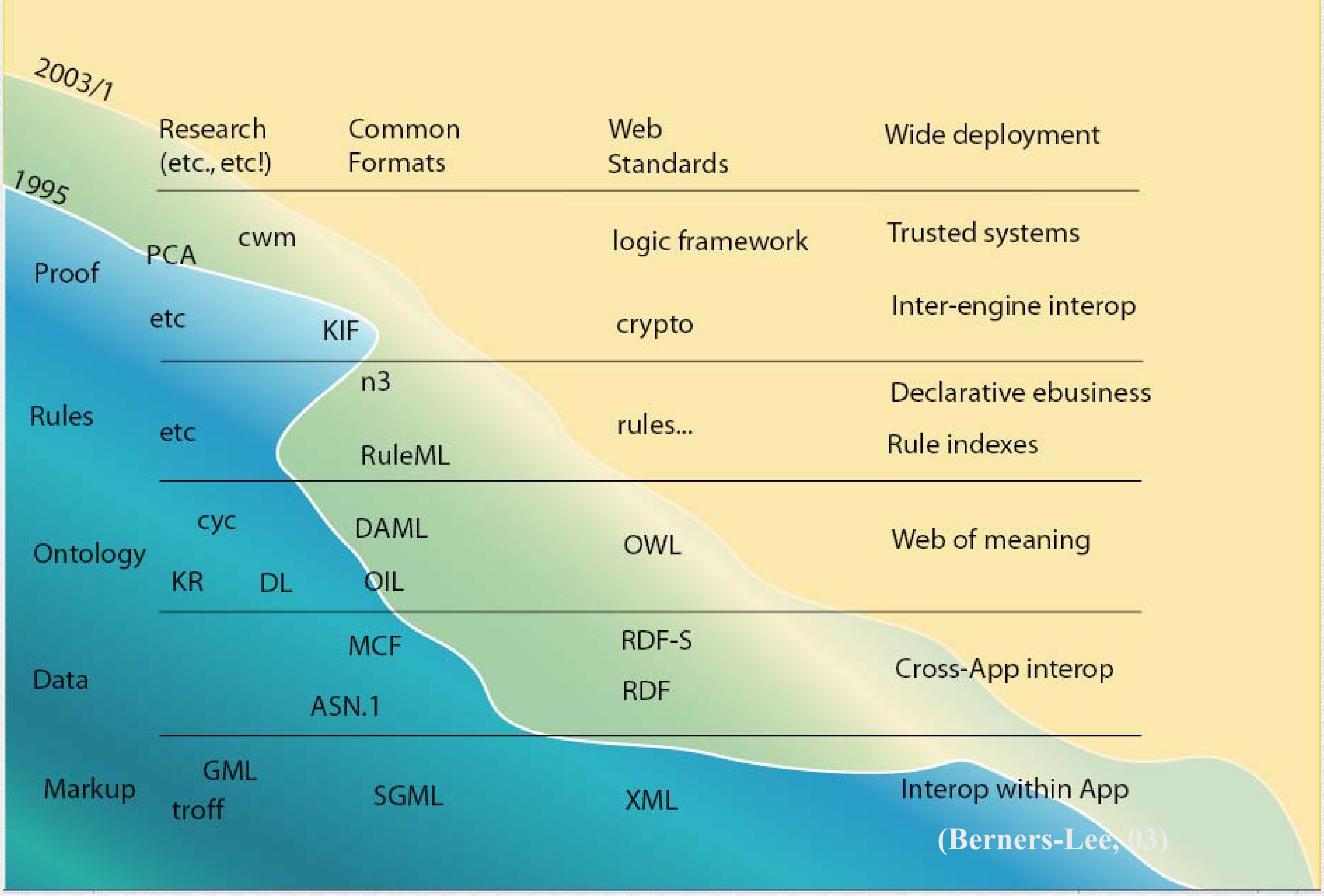
Better Web

- Employment of XML as structural encoding mechanism is a step forward, but not enough
- What does "<author>…</author>" mean to the machine?
- Furthermore, can the machine recognize the equivalence between <author> and <writer>, or the inverse relationship between <parentOf> and <childOf>?
- In addition to XML, a Web resource requires semantic encoding mechanism

Semantic Web

- The *Semantic Web* is an extension of the current one, in which information is given well-defined *meaning*, better enabling computers and people to work in cooperation. [Berners-Lee, Hendler and Lassila]
- *Meaning* or *semantics* can be given by having a case-by-case external agreement on a vocabulary
- Or it can be specified by employment of *ontology*
  - Ontology provides a vocabulary of terms
  - New terms can be formed by combining existing ones
  - Meaning of such terms is formally specified
  - Relationships between terms can also be specified

Semantic Wave



(Berners-Lee, 03)

[Hendler]

Ontology

Definition of Ontology

- Webster's Definition

  1 : a branch of metaphysics concerned with the nature and relations of being
  2 : a particular theory about the nature of being or the kinds of existents
- The word ontology is from the Greek *ontos* for being and *logos* for word.
- People use the word **ontology** to mean different things, e.g. glossaries & data dictionaries, thesauri & taxonomies, schemas & data models, and formal ontologies & inference.

Ontology in Computer Science

- John McCarthy first used the term *ontology* in 1980 in the paper: “Circumscription – A Form of Non-Monotonic Reasoning”, Artificial Intelligence, 5:13, 27–39.
- An ontology is
  - *a formal, explicit specification of a shared conceptualization* [Gruber93]
  - *a common vocabulary and agreed upon meanings to describe a domain of interest*
- Meanings of the keywords:
  - *conceptualization*: abstraction of some real-world phenomenon
  - *shared*: acceptance by a community, not restricted to some individuals
  - *specification*: definition
  - *explicit*: crystal-clear declarative meaning
  - *formal*: machine-processability
- In short, an ontology provides
  - a *common vocabulary* of terms
  - declarative definition of the *meaning of the terms* (semantics)
  - a *shared understanding* for people as well as machines

Ontology Languages

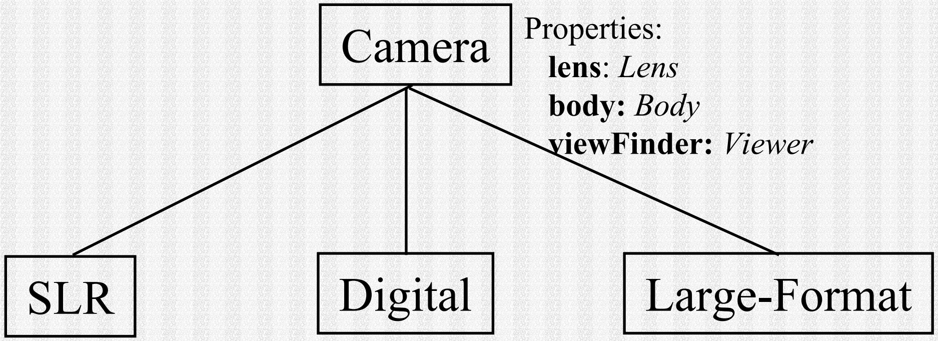
OWL (Web Ontology Language)

- OWL is an XML vocabulary that is used to define classes, their properties, as well as class and property relationships.
- OWL can define
  - Classes
  - Properties
  - Individuals
  - Subclass and other types of class relationships
  - Property relationships
  - Restrictions for property values
  - Individual relationships
- OWL is an extension of RDFS (Resource Description Framework Schema)
- OWL enables machine-processable semantics.

Examples of OWL Vocabulary

- **subClassOf** asserts that one class of items is a subset of another class of items
- **equivalentProperty** asserts that one property is equivalent to another
- **sameIndividualAs** asserts that one instance is the same as another instance
- **maxCardinality** specifies the maximum number of objects satisfying a property

Camera Ontology



Camera

Properties:
lens: *Lens*
body: *Body*
viewFinder: *Viewer*

SLR

Digital

Large-Format

Properties:
viewFinder: *hasValue= #ThroughTheLens*
(SLR is characterized by a viewFinder that is through the lens)

Properties:
body: *allValuesFrom BodyWithNonAdjustableShutterSpeed*
(i.e., no shutter-speed adjusting on LF cameras)

[Costello and Jacobs]

Example of using OWL to define two terms and their relationship

Example: Define the terms "Camera" and "SLR".
State that SLRs are a type of Camera.

These two terms (classes) and their relationship is defined using the OWL vocabulary

```
<owl:Class rdf:ID="Camera"/>
```

```
<owl:Class rdf:ID="SLR">
        <rdfs:subClassOf rdf:resource="#Camera"/>
</owl:Class>
```

[Costello and Jacobs]

Relationship between focal-length and lens size

This OWL element states that focal-length is equivalent to lens size.

```
<owl:DatatypeProperty rdf:ID="focal-length">
        <owl:equivalentProperty rdf:resource="#size"/>
        <rdfs:domain rdf:resource="#Lens"/>
        <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

"focal-length is synonymous with (lens) size"

[Costello and Jacobs]

Summary of OWL Vocabulary: Class Constructors

- **allValuesFrom**: P(x,y) and y=allValuesFrom(C)
- **someValuesFrom**: P(x,y) and y=someValuesFrom(C)
- **cardinality**: cardinality(P) = N
- **minCardinality**: minCardinality(P) = N
- **maxCardinality**: maxCardinality(P) = N
- **intersectionOf**: C = intersectionOf(C1, C2, …)
- **unionOf**: C = unionOf(C1, C2, …)
- **complementOf**: C = complementOf(C1)
- **oneOf**: C = one of(v1, v2, …)

where:

C, C1, C2: OWL descriptions

P: an OWL property

x, y: variables, OWL individuals or OWL data values

N: a number

Summary of OWL Vocabulary: Axioms

subtClassOf: C1 = subClassOf(C2)

equivalentClassOf: C1 = C2

disjointWith: C1 != C2

transitiveProperty: if P(x,y) and P(y,z) then P(x, z)

FunctionalProperty: if P(x,y) and P(x,z) then y=z

InverseOf: if P1(x,y) then P2(y,x)

InverseFunctionalProperty: if P(y,x) and P(z,x) then y=z

equivalentProperty: P1 = P2

subPropertyOf: P1 = subClassOf(P2)

equivalentPropertyOf: P1 = P2

sameIndividualAs: I1 = I2

differentFrom: I1 != I2

where:

C, C1, C2: OWL descriptions

P1, P2: OWL properties

x, y, z: variables, OWL individuals or OWL data values

I1, I2: individuals

Summary of OWL Vocabulary: Axioms

- **subtClassOf**: C1 = subClassOf(C2)
- **equivalentClassOf**: C1 = C2
- **disjointWith**: C1 != C2
- **transitiveProperty**: if P(x,y) and P(y,z) then P(x, z)
- **FunctionalProperty**: if P(x,y) and P(x,z) then y=z
- **InverseOf**: if P1(x,y) then P2(y,x)
- **InverseFunctionalProperty**: if P(y,x) and P(z,x) then y=z
- **equivalentProperty**: P1 = P2
- **subPropertyOf**: P1 = subClassOf(P2)
- **equivalentPropertyOf**: P1 = P2
- **sameIndividualAs**: I1 = I2
- **differentFrom**: I1 != I2

where:

C, C1, C2: OWL descriptions

P1, P2: OWL properties

x, y, z: variables, OWL individuals or OWL data values

I1, I2: individuals

OWL with Rules

- In order to extend the expressive power of OWL, a Semantic Web Rule Language (SWRL) has been proposed
- SWRL combines OWL DL and OWL Lite sublanguages of OWL with the Unary/Binary Datalog sublanguages of RuleML (http://www.ruleml.org), enabling Horn-like rules to be combined with an OWL knowledge base
- The proposed rules are of the form of an implication between an antecedent (body) and consequent (head), where both the antecedent and consequent consist of zero or more atoms
- The atoms can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values

Example of SWRL

```
<swrl:Variable rdf:ID="x1"/>
<swrl:Variable rdf:ID="x2"/>
<swrl:Variable rdf:ID="x3"/>

<ruleml:Imp>

  <ruleml:body rdf:parseType="Collection">
    <swrl:individualPropertyAtom>
      <swrl:propertyPredicate
rdf:resource="&eg;hasParent"/>
      <swrl:argument1 rdf:resource="#x1" />
      <swrl:argument2 rdf:resource="#x2" />
    </swrl:individualPropertyAtom>
    <swrl:individualPropertyAtom>
      <swrl:propertyPredicate
rdf:resource="&eg;hasSibling"/>
      <swrl:argument1 rdf:resource="#x2" />
      <swrl:argument2 rdf:resource="#x3" />
    </swrl:individualPropertyAtom>
    <swrl:individualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;hasSex"/>
      <swrl:argument1 rdf:resource="#x3" />
      <swrl:argument2 rdf:resource="#male" />
    </swrl:individualPropertyAtom>
  </ruleml:body>
```

```
<ruleml:head rdf:parseType="Collection">
    <swrl:individualPropertyAtom>
      <swrl:propertyPredicate
rdf:resource="&eg;hasUncle"/>
      <swrl:argument1 rdf:resource="#x1" />
      <swrl:argument2 rdf:resource="#x3" />
    </swrl:individualPropertyAtom>
  </ruleml:head>
</ruleml:Imp>
```

This rule asserts that if x1 hasParent x2, x2 hasSibling x3, and x3 hasSex male, then x1 hasUncle x3.

Problems with SWRL

- SWRL is a mere XMLization of a subset of Horn logic
- SWRL is too verbose and is a not succinct representation of real-world domain data
- Handling of XML data by SWRL is not direct
- Efficient computational mechanism may be difficult to develop

Semantic Web and Databases

DB Contributions to SW

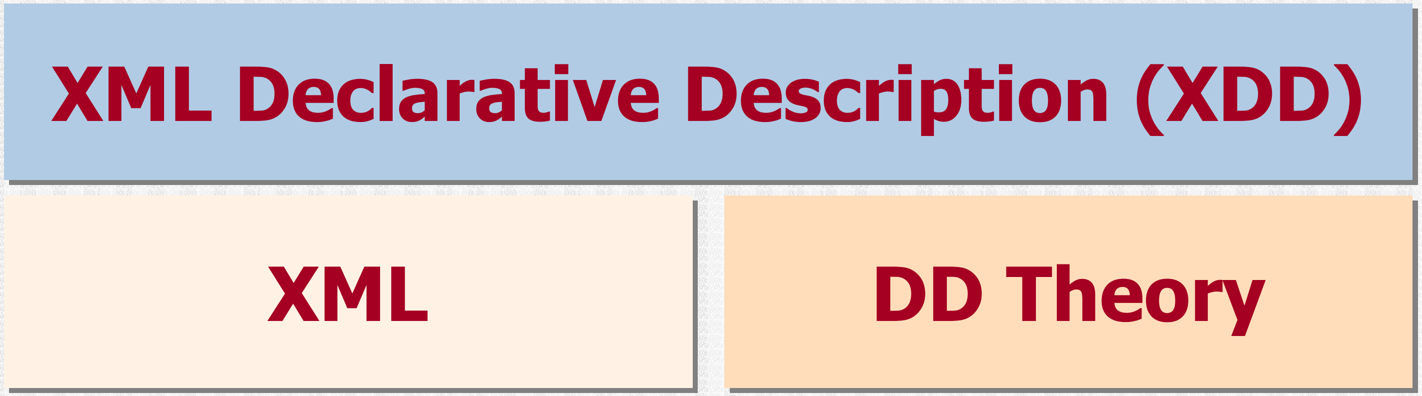
- "Ask not what the Semantic Web can do for you, ask what you can do for the Semantic Web" [Hans-Georg Stork, European Union, http://lsdis.cs.uga.edu/SemNSF]
- DB provides a foundation layer for SW
- Conventional DB techniques could be extended/modified to solve the scalability and performance problems of SW
  - Storage structure for XML documents
  - Indexing for queries
  - Dependencies/constraints
  - Concurrency control
  - Distributed DB
  - Transaction processing
  - Schema/data integration
  - Access control and security

SW Contributions to DB

- Conceptual modeling
- (Semantic) Query formulation and evaluation
- High precision of data services
- Semantics preserving/based schema/data integration/transformation/versioning
- Interoperability of data
- Annotation for multimedia DB
- Metadata-driven data warehouses
- OLAP
- Data mining

XML Declarative Description (XDD)

XDD


XML Declarative Description (XDD)

XML

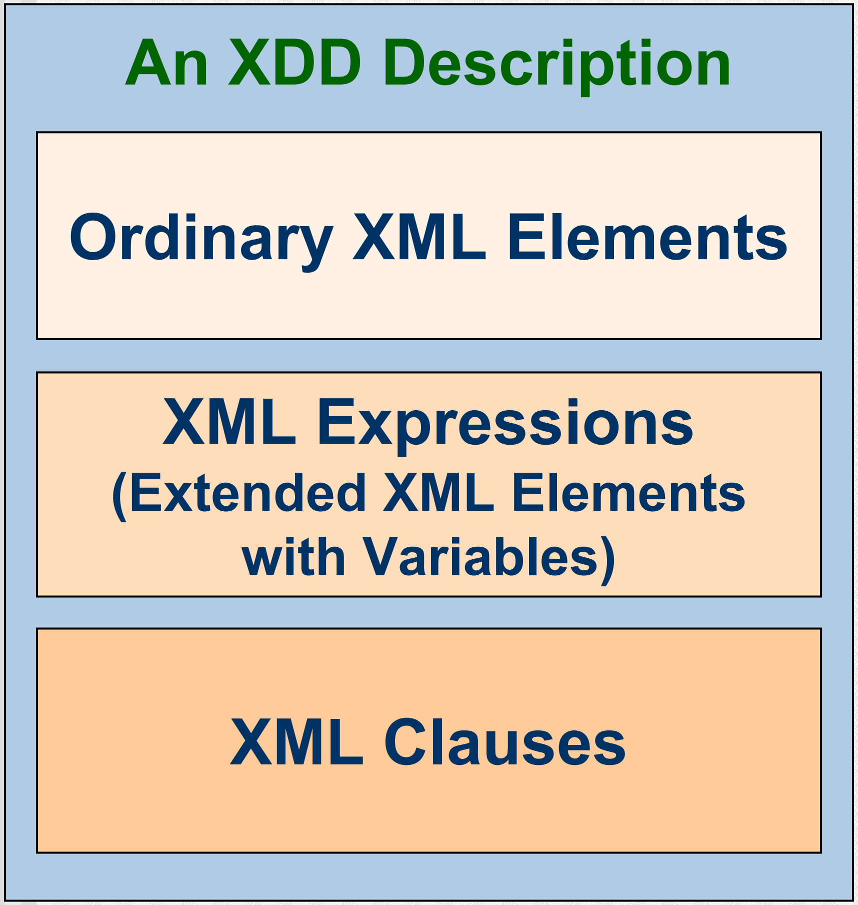
DD Theory

- XDD is unified, XML-based knowledge representation language with
  - well-defined declarative semantics, and
  - a support for general computation and inference mechanisms.
- It employs:
  - XML's nested tree structure as its underlying data structure,
  - Declarative Description theory as a framework to enhance its expressive power.

XDD at a Glance

- Basic modeling elements: ordinary XML elements
  - Capable of representing explicit complex entities and their relationships in a real application domain.
- By means of XML expressions—a generalization of XML elements with variables—and XML clauses with constraints, sets and negations:
  - XDD additionally allows representation of implicit complex entities as well as their classes, relationships, rules, constraints and queries.

XDD Descriptions



An XDD Description

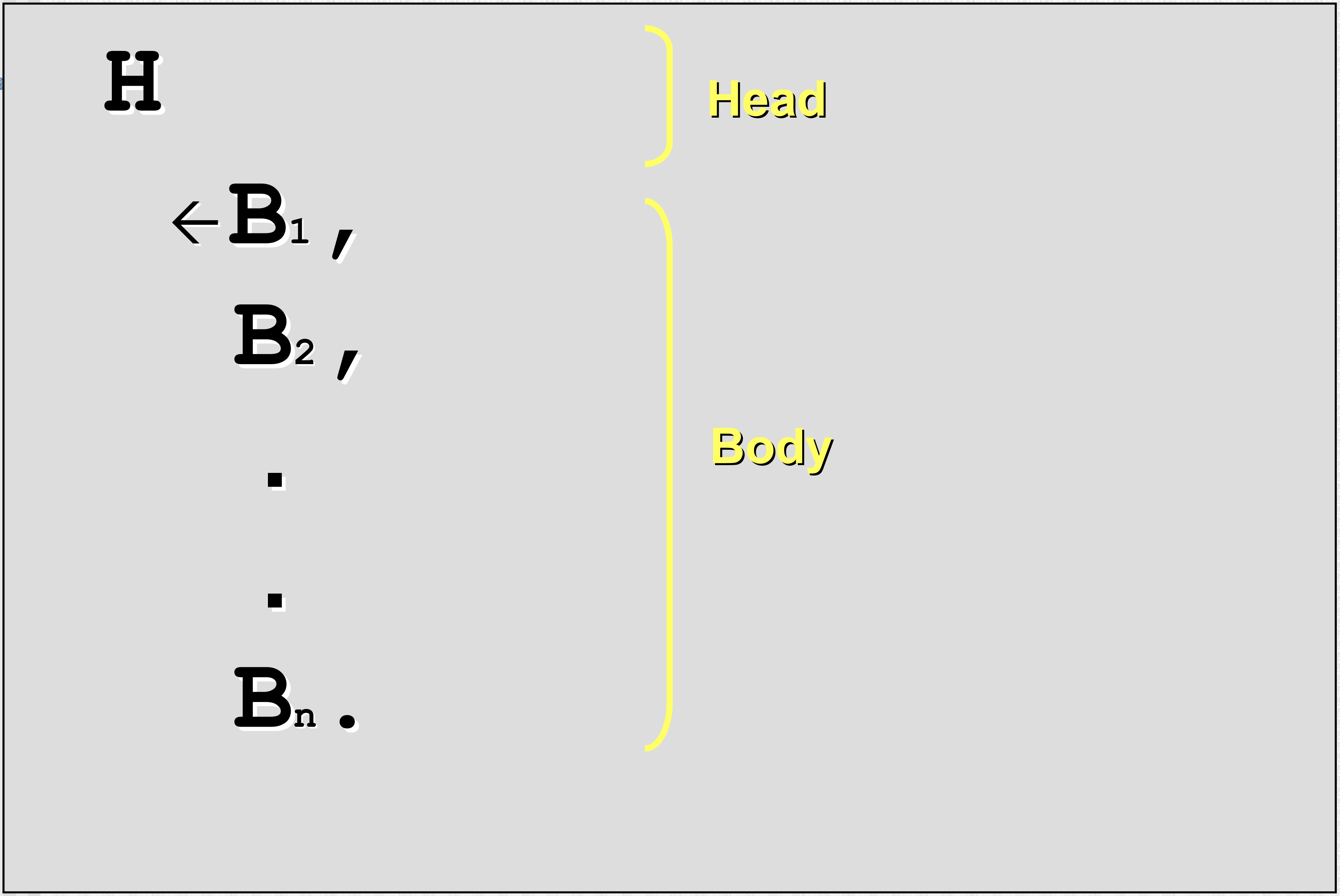
Ordinary XML Elements

XML Expressions
(Extended XML Elements
with Variables)

XML Clauses

- Representing explicit information items in a particular domain and denoting a semantic unit
- Representing implicit information or a set of semantic units
- Modeling integrity constraints, rules, conditional relationships and axioms

XML Clauses



H

$\leftarrow B_1$,

B_2 ,

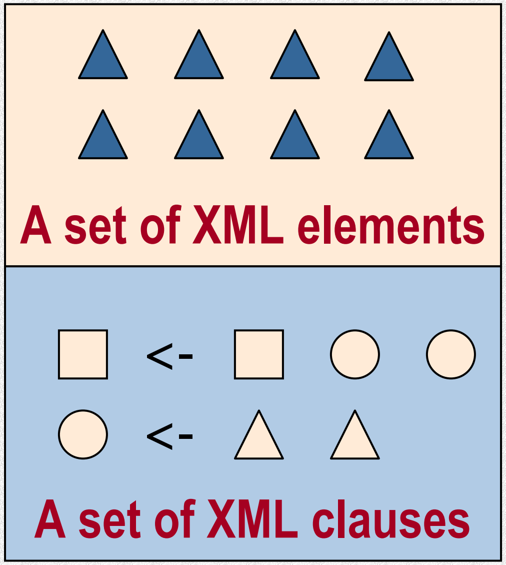
.

.

B_n .

Semantics of an XDD Description

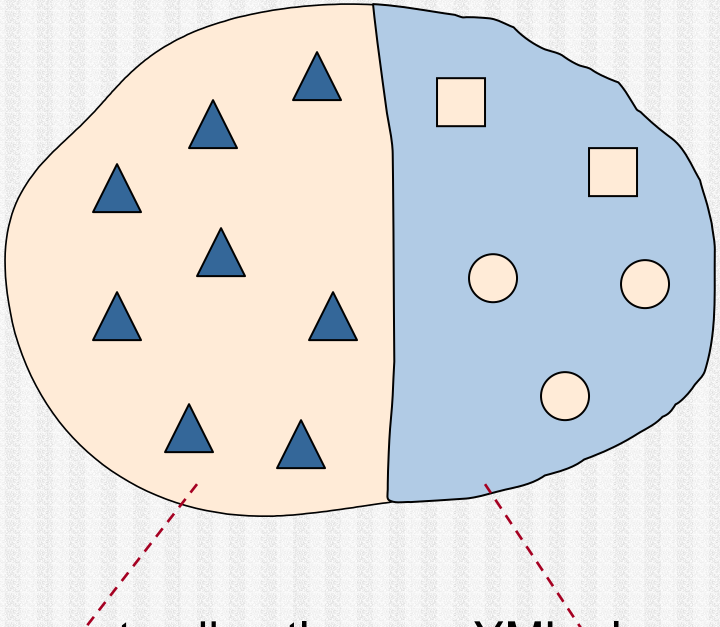
An XDD Description *P*



A set of XML elements

A set of XML clauses

Semantics of *P*



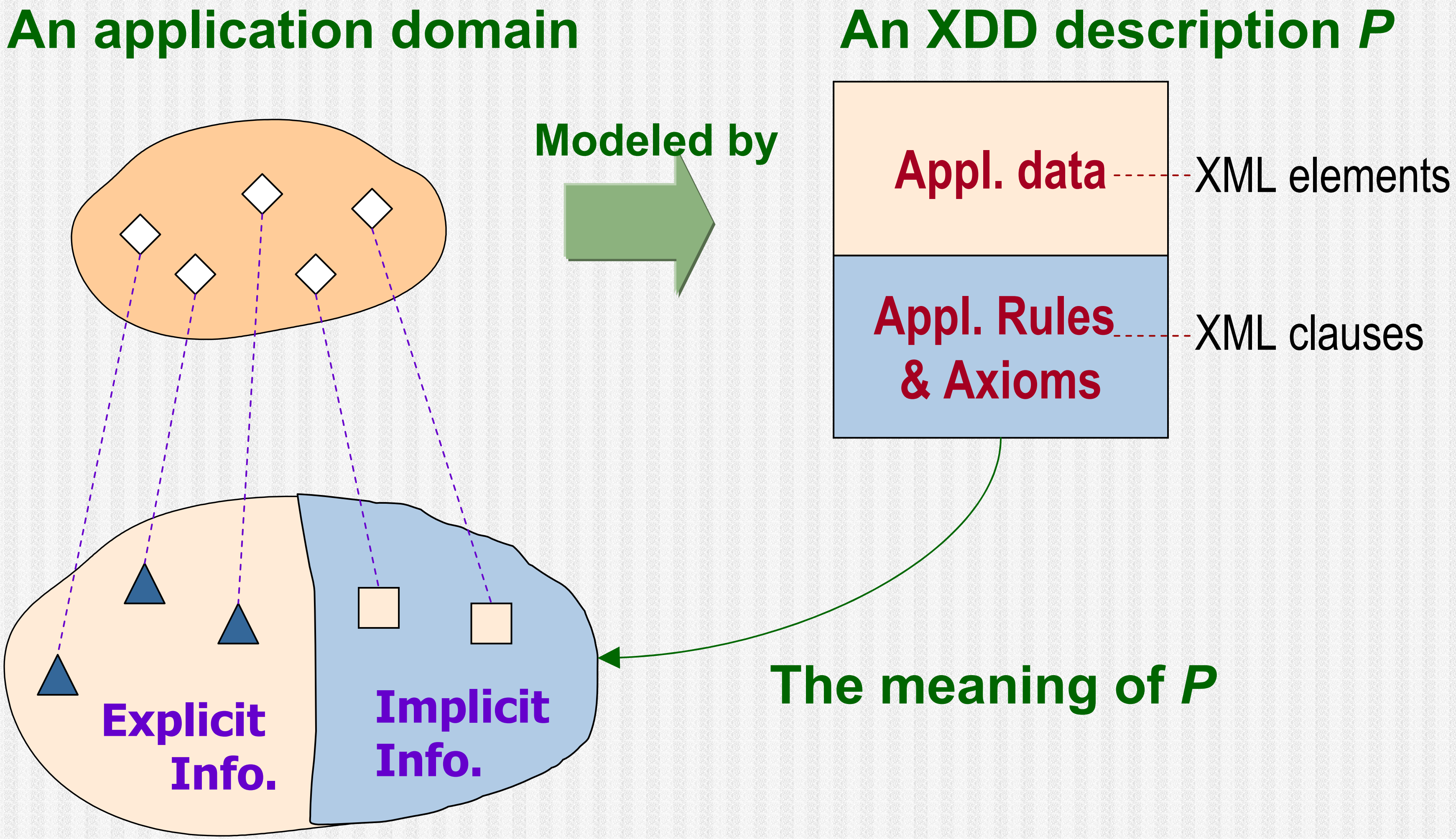
XML elements, directly described by the XML elements in *P*

XML elements, derived from the XML clauses in *P*.

XDD at a Glance

- Enables direct representation of data items, encoded in XML-based application markup languages.
- Extends these languages' expressiveness by facilitation of simple means for succinct and uniform expression of implicit information, rules and conditional relationships.
- Allows their semantics to be determined directly, and also provides efficient computation.

XDD at a Glance



An application domain

An XDD description *P*

Modeled by

Appl. data

XML elements

Appl. Rules & Axioms

XML clauses

Explicit Info.

Implicit Info.

The meaning of *P*

Variable Types in XML Expressions

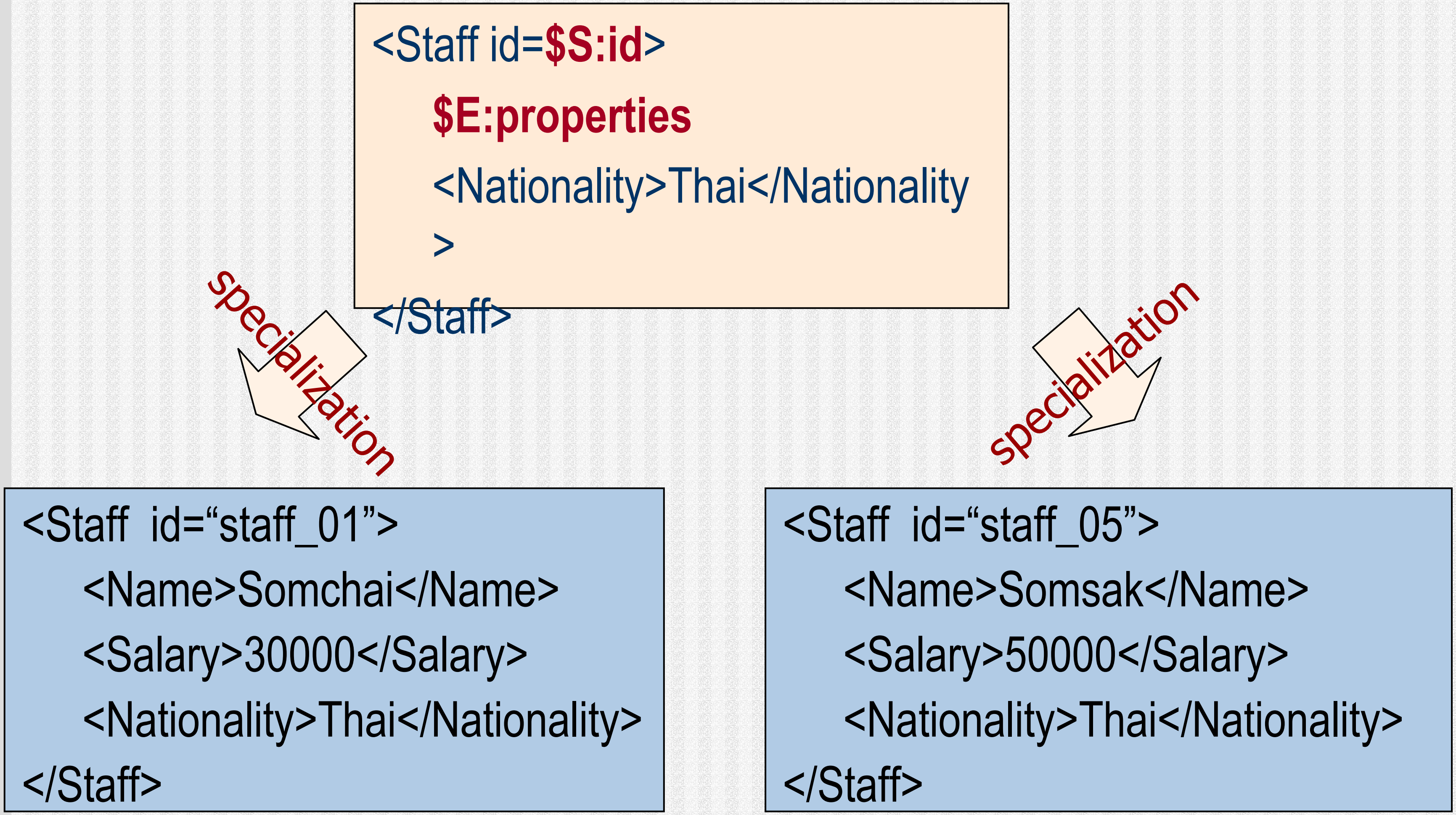
Variable Type	Set of Variables	Variable Names Beginning with	Instantiation to
N-variables: Name-variables	V_N	$N	Element types or attribute names
S-variables: String-variables	V_S	$S	Strings
P-variables: Attribute-value-pair-variables	V_P	$P	Sequences of zero or more attribute-value pairs
E-variables: XML-expression-variables	V_E	$E	Sequences of zero or more XML expressions
I-variable: Intermediate-expression variable	V_I	$I	Parts of XML expressions

Ex: Ground XML Expressions
Staff Information

E1:
```
<Staff  id="staff_01">
        <Name>Somchai</Name>
        <Salary>30000</Salary>
        <Nationality>Thai</Nationality>
</Staff>
```

E2:
```
<Staff  id="staff_05">
        <Name>Somsak</Name>
        <Salary>50000</Salary>
        <Nationality>Thai</Nationality>
</Staff>
```

Ex: Non-ground XML Expression
A Set of Thai Staff



```
<Staff id=$S:id>
    $E:properties
    <Nationality>Thai</Nationality>
</Staff>
```

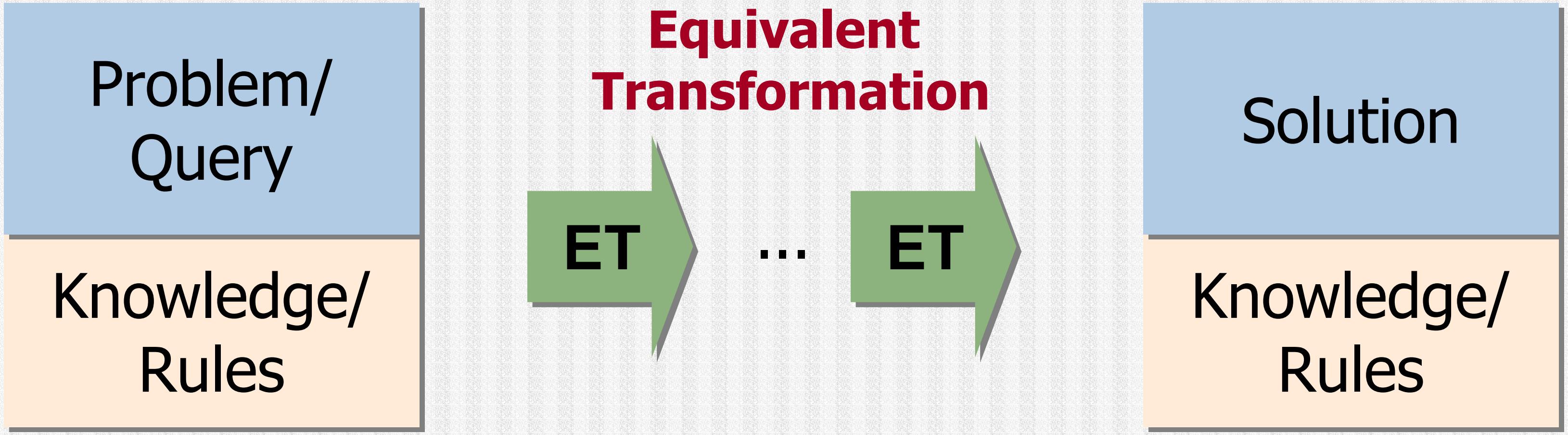
specialization

specialization

```
<Staff  id="staff_01">
    <Name>Somchai</Name>
    <Salary>30000</Salary>
    <Nationality>Thai</Nationality>
</Staff>
```

```
<Staff  id="staff_05">
    <Name>Somsak</Name>
    <Salary>50000</Salary>
    <Nationality>Thai</Nationality>
</Staff>
```

Computation with XDD



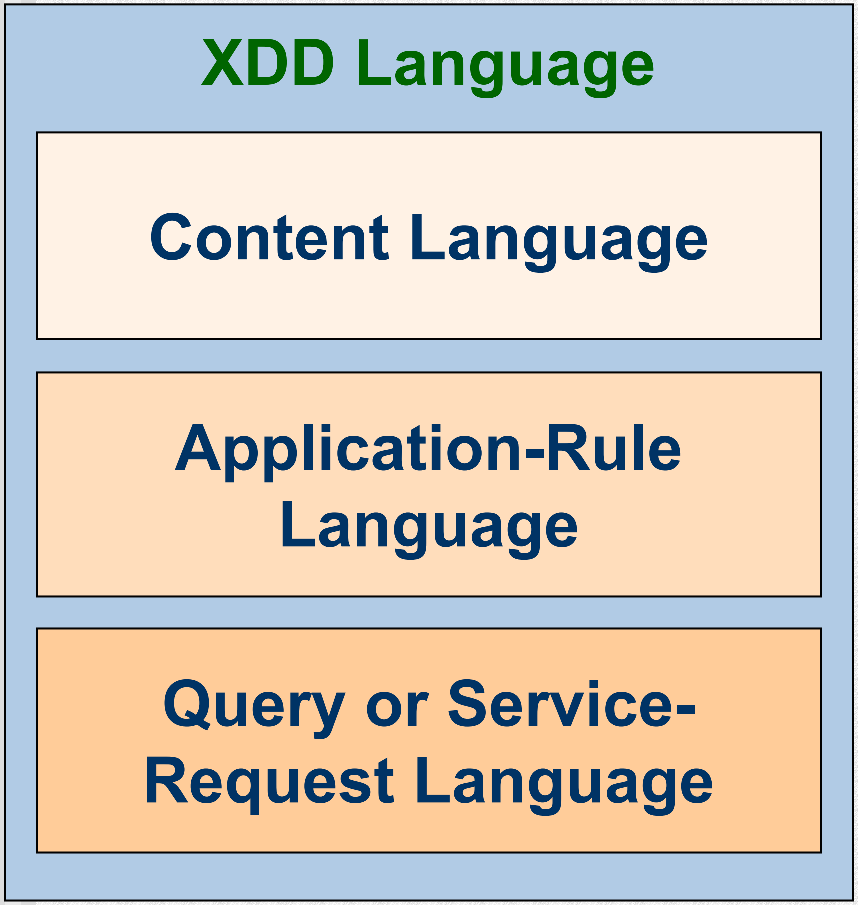
- XDD concentrates on information representation to provide a concise and expressive language with precise and well-defined semantics.
- It achieves efficient manipulation and reasoning by employment of the Equivalent Transformation (ET) computational paradigm.

Semantic Web Modeling

Modeling the Semantic Web

<table>
<tr><th>Semantic-Web Component</th><th colspan="2">Expressed as</th></tr>
<tr><td>1. Constraints on the information-exchange format</td><td colspan="2">XML non-unit clauses</td></tr>
<tr><td>2. Ontologies
■ Concept descriptions
■ Hierarchy of concepts</td><td colspan="2">
XML unit clauses
XML non-unit clauses or the XML specialization system Γ_X</td></tr>
<tr><td>3. Contents
■ Objects
■ Relationships among objects</td><td colspan="2">
XML unit clauses
XML non-unit clauses</td></tr>
<tr><td><b>A resource on the Semantic Web (Contents + Ontologies + Constraints)</b></td><td>Modeled as ==></td><td>An XDD <i>P</i> on Γ_X comprising
<b>XML unit clauses</b>
<b>+</b>
<b>XML non-unit clauses</b></td></tr>
<tr><td><b>The Semantics of the resource</b></td><td>is</td><td>💣(<b><i>P</i></b>)</td></tr>
</table>

Modeling Semantic Web Appl.



XDD Language

Content Language

Application-Rule Language

Query or Service-Request Language

- For modeling application data
- For modeling application rules or logic
- For modeling user’s queries or requests for services

Domain Ontologies and Contents

- A description of domain-specific ontologies and their instances encoded in an ontology language, such as OWL, becomes immediately an XDD description comprising solely ordinary XML elements.
- XML clauses can be employed to define the axiomatic semantics of each ontology modeling primitive which includes a certain notion of implication.
- XML clauses can be used to model arbitrary rules, axioms, constraints and queries.

XDD Description:

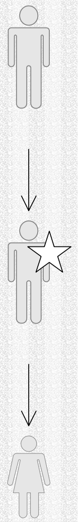
Ontologies and instances

```
C1:   <owl:Class rdf:ID=“Person”>
             <rdfs:label>person</rdfs:label>
      </owl:Class>
C2:   <owl:ObjectProperty rdf:ID=“hasChild”>
             <rdfs:domain rdf:resource=“#Person”/>
             <rdfs:range rdf:resource=“#Person”/>
      </owl:ObjectProperty>
C3:   <owl:ObjectProperty rdf:ID=“hasParent”>
             <owl:inverseOf rdf:resource=“#hasChild”/>
      </owl:ObjectProperty>
```

Application-specific ontology definition expressed in terms of OWL.

```
C4:   <Person rdf:about=“Jack”>
             <age>52</age>
             <hasChild rdf:resource= “#John”/>
             <hasAirlineMembership/>
      </Person>
C5:   <Person rdf:about=“John”>
             <age>29</age>
             <hasChild rdf:resource=“#Jill”/>
             <hasAirlineMembership rdf:resource=“#tg9000”/>
      </Person>
C6:   <Person rdf:about=“Jill”>
             <age>7</age>
             <hasAirlineMembership/>
      </Person>
```

Ontology instances (application data)



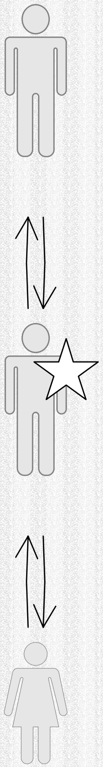
XDD Description: Ontology Axioms

If a property R is an inverse of a property P,
then for any resource X the value of a property P of which is a resource Y,
one can infer that Y also has a property R the value of which is the resource X.

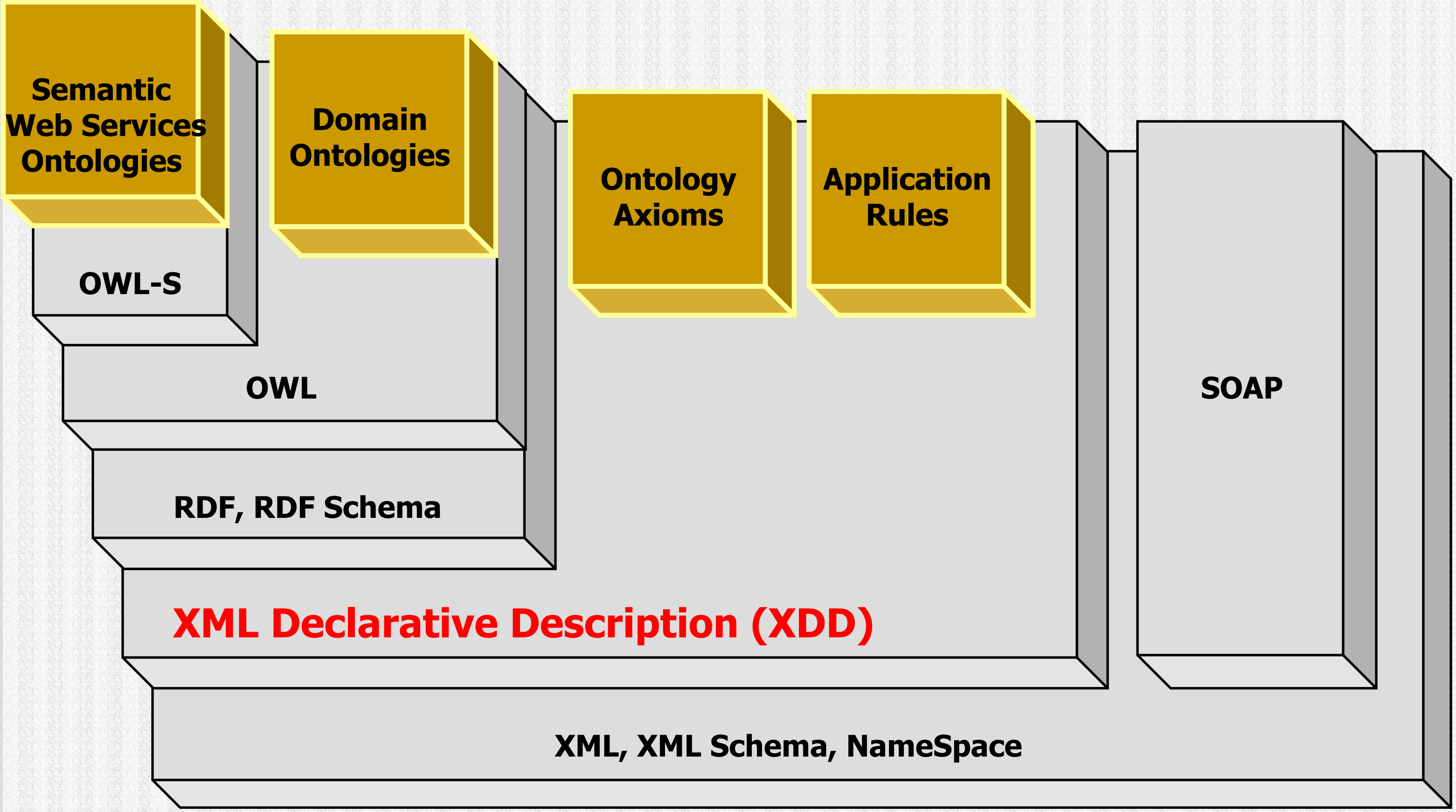
```
C7:  <$N:classB  rdf:about=$S:resourceY>
           $E:instance1Elmt
           <$S:propertyR rdf:resource=$S:resourceX/>
     </$N:classB>
            ←         <owl:ObjectProperty rdf:ID=$S:propertyR>
                            <owl:inverseOf rdf:resource=$S:propertyP/>
                            $E:inversePropertyElmt
                      </owl:ObjectProperty>,
                      <$N:classA  rdf:ID=$S:resourceX>
                            <$S:propertyP rdf:resource=$S:resourceY/>
                            $E:XProperties
                      </$N:classA>,
                      <$N:classB  rdf:ID=$S:resourceY>
                            $E:YProperties
                      </$N:classB>.
```

Derived Information

```
<Person rdf:about="John">
    <age>29</age>
    <hasChild rdf:resource="#Jill"/>
    <hasAirlineMembership
        rdf:resource="#tg9000"/>
    <hasParent rdf:resource="#Jack"/>
</Person>
<Person rdf:about="Jill">
    <age>7</age>
    <hasAirlineMembership/>
    <hasParent rdf:resource="#John"/>
</Person>
```



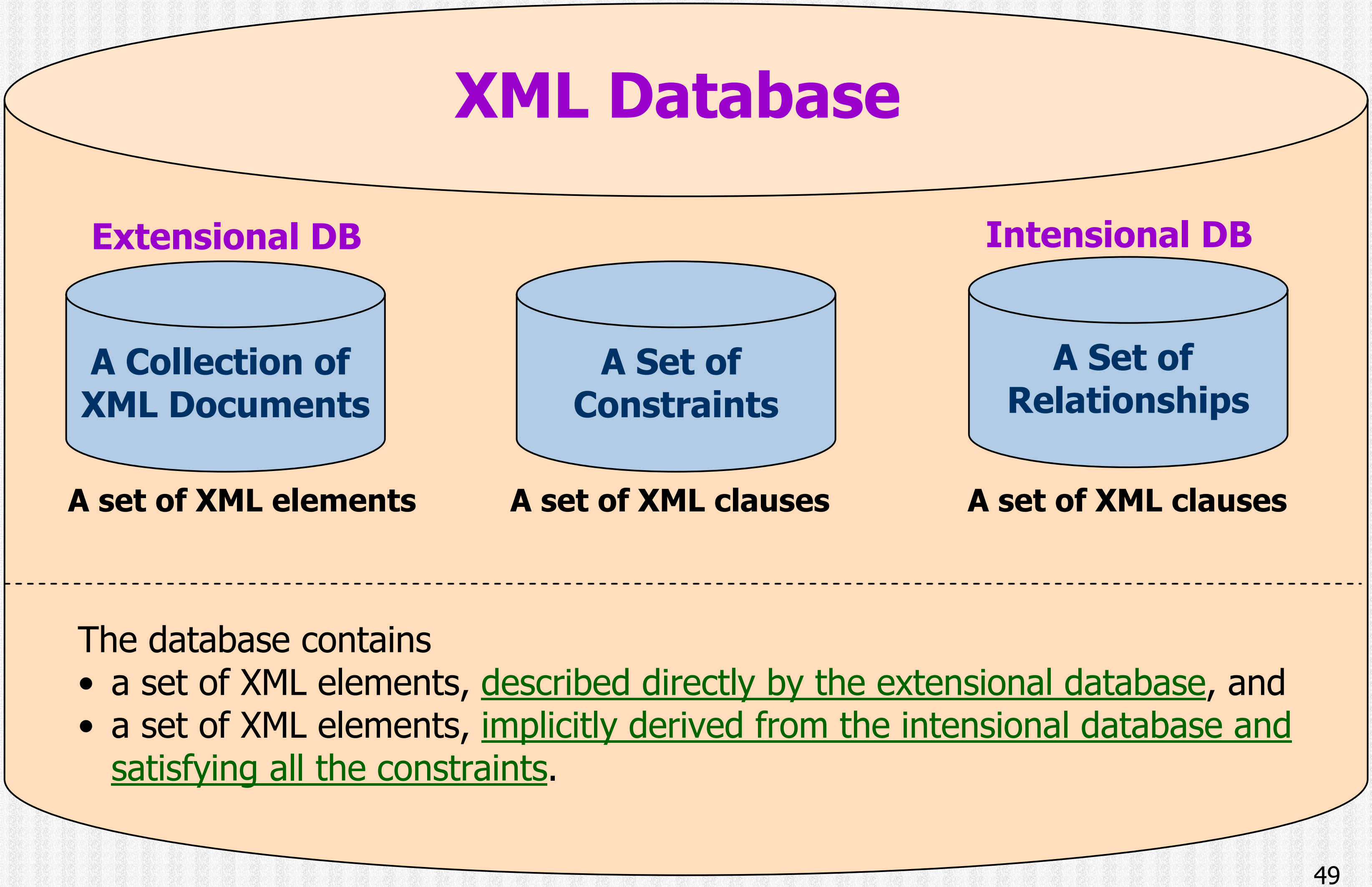
Language Layers with XDD


Semantic Web Services Ontologies
Domain Ontologies
Ontology Axioms
Application Rules
OWL-S
OWL
SOAP
RDF, RDF Schema
XML Declarative Description (XDD)
XML, XML Schema, NameSpace

Language Layers

XML Database Modeling

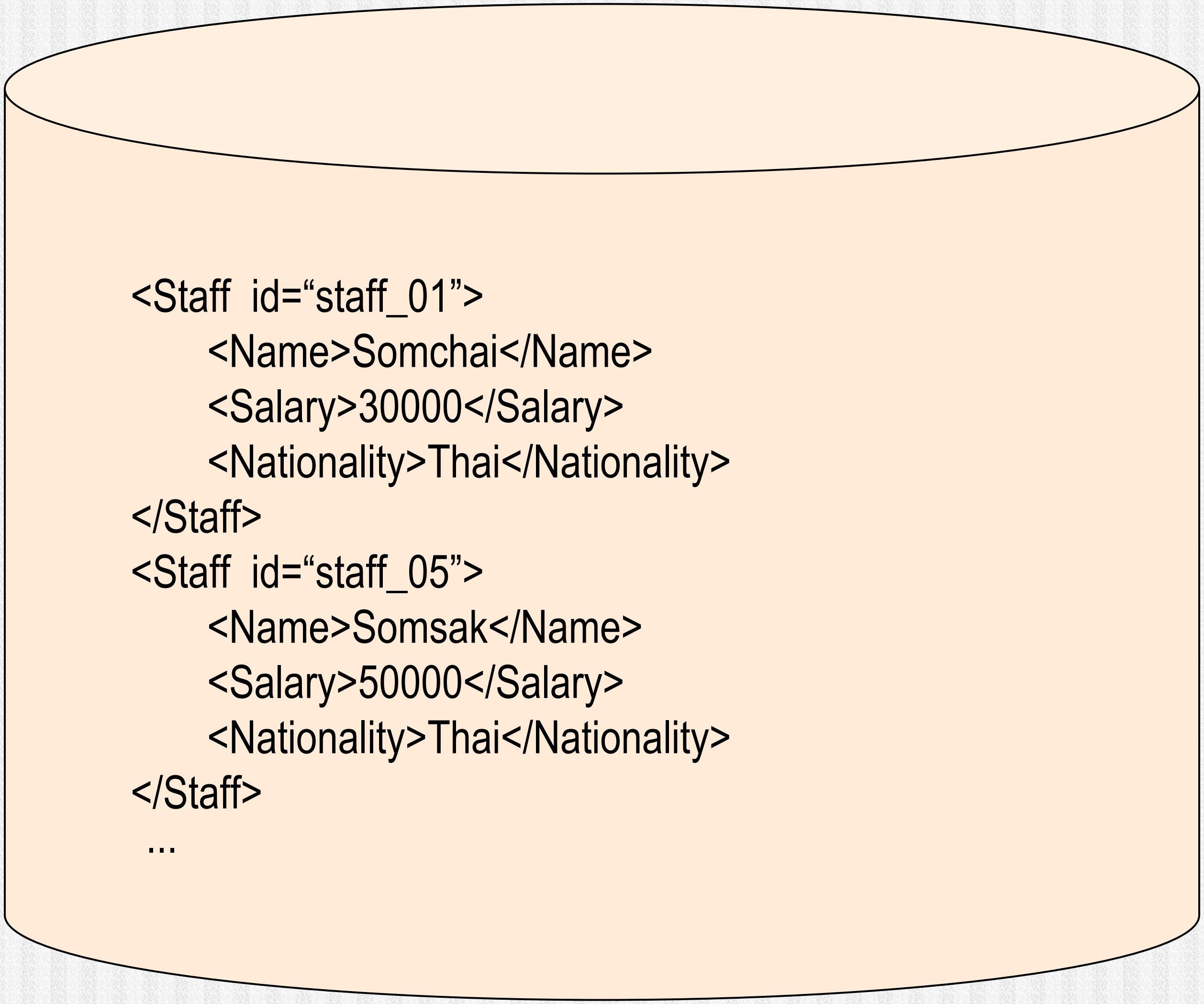
XML Database Modeling



The database contains

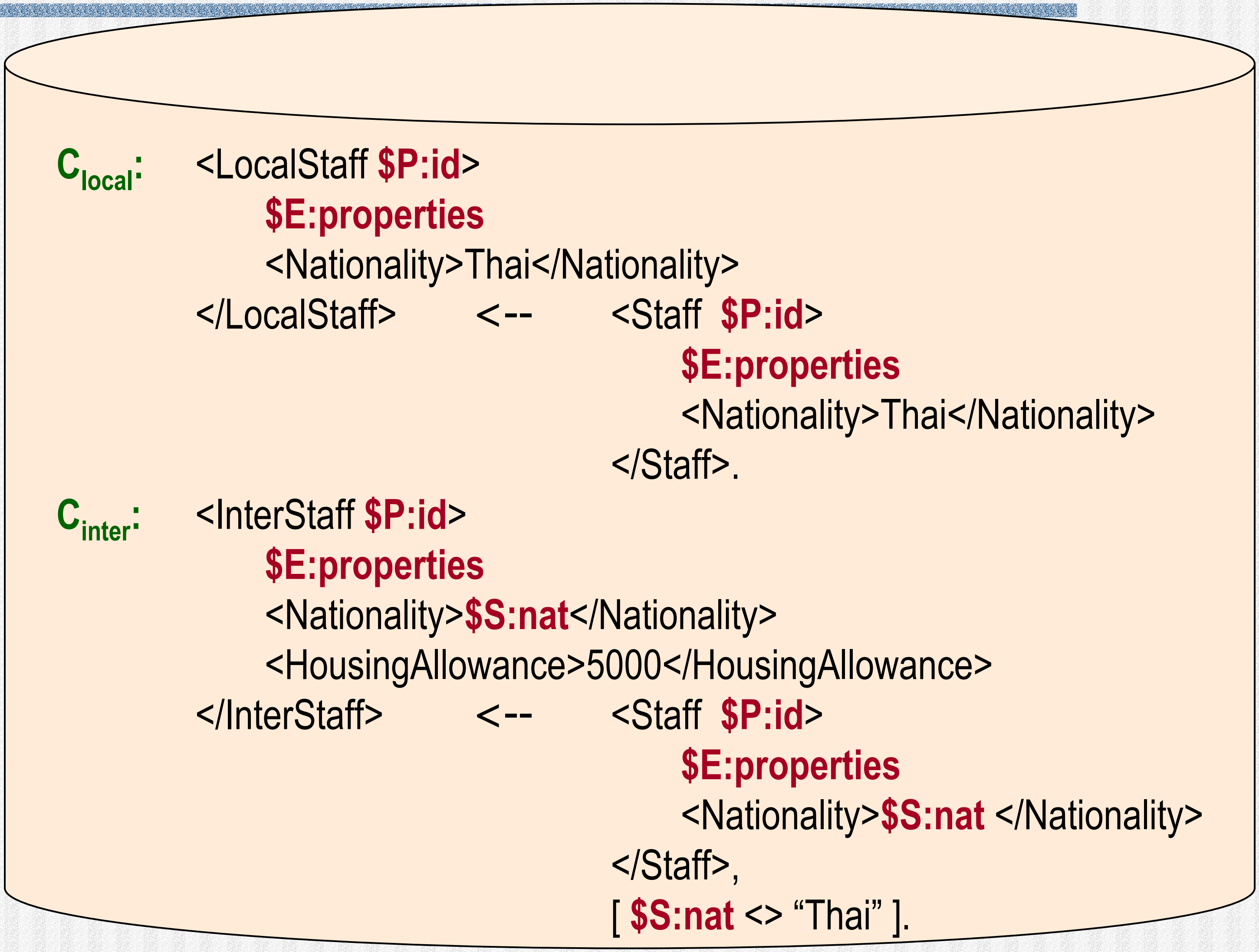
- a set of XML elements, described directly by the extensional database, and
- a set of XML elements, implicitly derived from the intensional database and satisfying all the constraints.

Example 1: Extensional Database



```
<Staff  id=“staff_01”>
      <Name>Somchai</Name>
      <Salary>30000</Salary>
      <Nationality>Thai</Nationality>
</Staff>
<Staff  id=“staff_05”>
      <Name>Somsak</Name>
      <Salary>50000</Salary>
      <Nationality>Thai</Nationality>
</Staff>
 ...
```

Example 2: Intensional Database



C_{local}:

```
<LocalStaff $P:id>
        $E:properties
        <Nationality>Thai</Nationality>
</LocalStaff>      <--      <Staff  $P:id>
                                    $E:properties
                                    <Nationality>Thai</Nationality>
                            </Staff>.
```

C_{inter}:

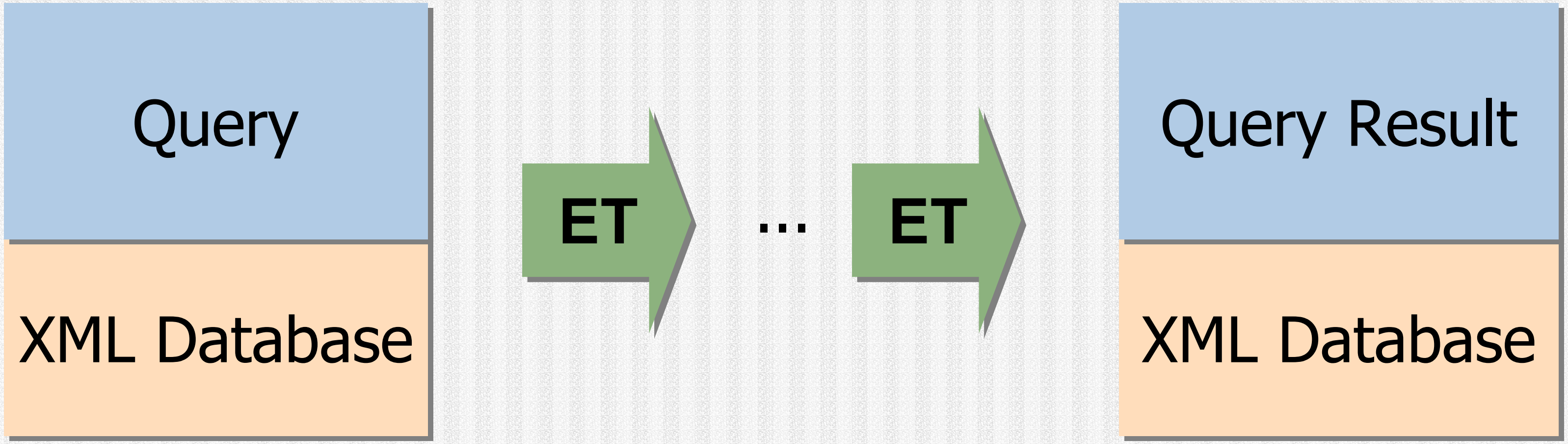
```
<InterStaff $P:id>
        $E:properties
        <Nationality>$S:nat</Nationality>
        <HousingAllowance>5000</HousingAllowance>
</InterStaff>      <--      <Staff  $P:id>
                                    $E:properties
                                    <Nationality>$S:nat </Nationality>
                            </Staff>,
                            [ $S:nat <> “Thai” ].
```

Example 3: Integrity Constraint

C_{con}:

```
<ConstraintViolation  type="SalaryConstraint">
        <LocalStaff  $P:id>
                <Salary>$S:salary</Salary>
        </LocalStaff>
</ConstraintViolation>
                <--        <LocalStaff  $P:id>
                                   $E:e1
                                   <Salary>$S:salary</Salary>
                                   $E:e2
                           </LocalStaff>,
                           [$S:salary > 70000].
```

Query Formulation and Evaluation



- A query about information in an XML database is formulated as an XDD description, comprising one or more XML clauses.
- Evaluation of a query against a database is carried out by means of ET computational mechanism.

Query Formulation

- A query consists of three parts
  - a pattern:
    specification of the document structure
  - a filter:
    specification of selection criteria
  - a constructor:
    specification of the query result

Query Formulation

A query consists of three parts

A pattern:
specification of the document structure

A filter:
specification of selection criteria

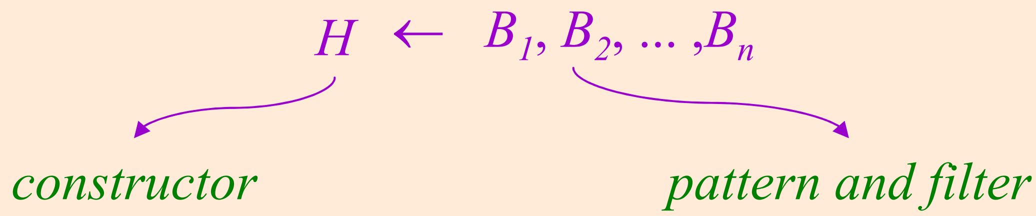
A constructor:
specification of the query result

A Query

Φορμυλατεδ ασ

A set of one or more clauses

Each clause has the form:



$$H \leftarrow B_1, B_2, \ldots ,B_n$$

constructor

pattern and filter

where

- H (constructor) is an XML expression describing the resulting XML elements
- B_i (pattern and filter) is an XML expression, or an XML constraint describing the pattern of XML elements to be selected as well as selection conditions.

Ex: Query Formulation

List names and salaries of all LocalStaff who can earn more than 40000.

```
Q:  <Answer>
          <Name> $S:name </Name>
          <Salary> $S:salary </Salary>
    </Answer>
```

Constructor

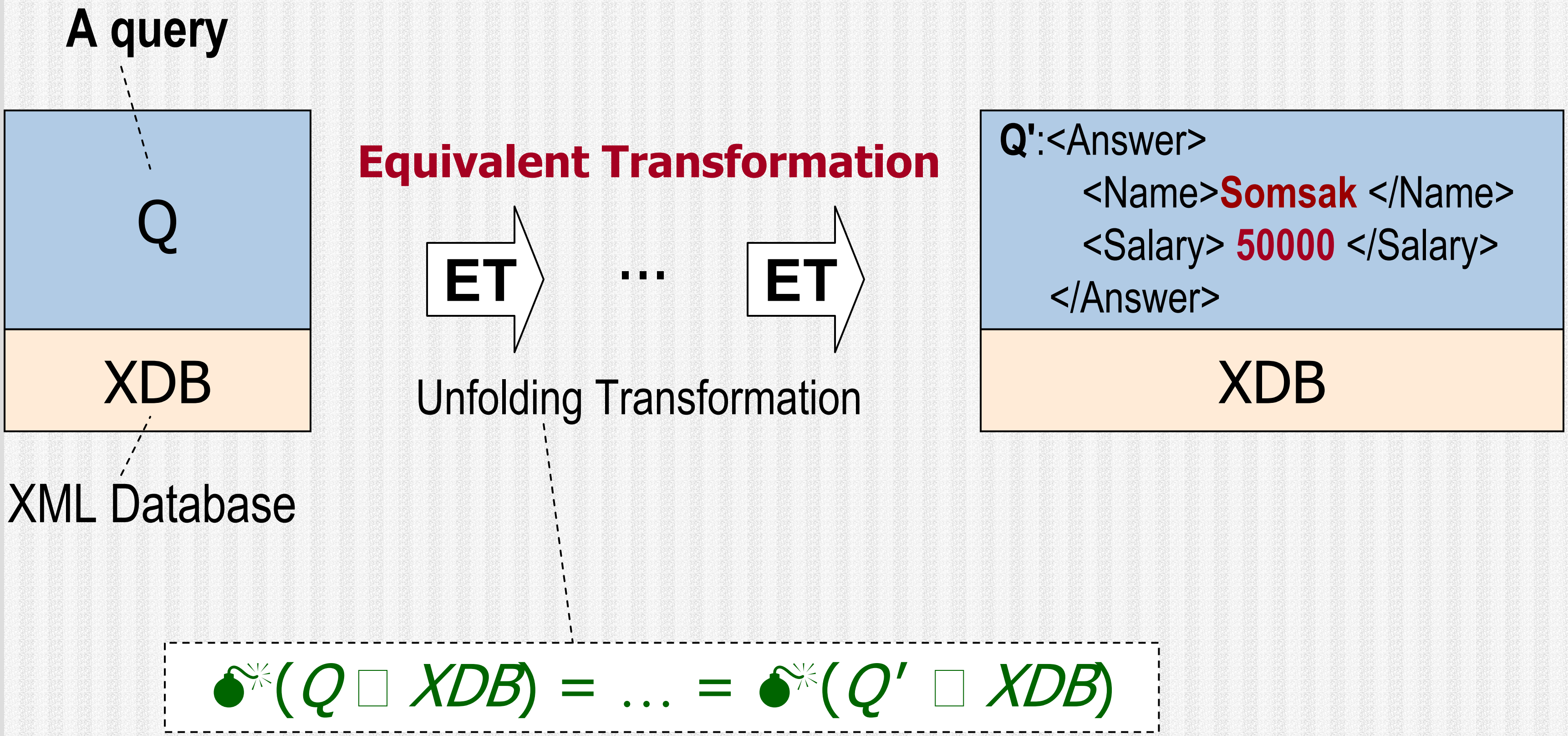
```
          <--    <LocalStaff  id=$S:id>
                       <Name> $S:name </Name>
                       <Salary> $S:salary </Salary>
                       $E:properties
                 </LocalStaff>,
```

Pattern

```
                 [$S:salary > 40000].
```

Filter

Ex: Query Evaluation



A query

Q

XDB

XML Database

Equivalent Transformation

ET … ET

Unfolding Transformation

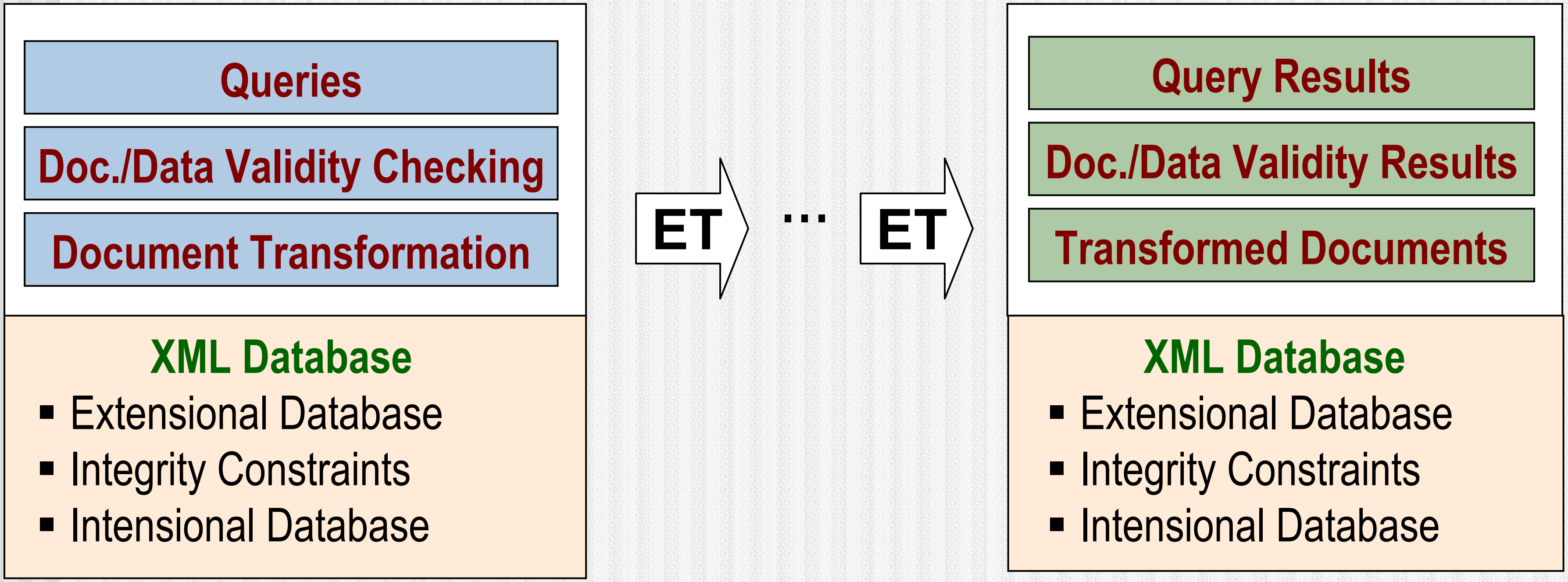
Q':<Answer>
<Name>**Somsak** </Name>
<Salary> **50000** </Salary>
</Answer>

XDB

💣(*Q* □ *XDB*) = … = 💣(*Q′* □ *XDB*)

XDD: A Unified Framework for Modeling XML Databases with DTDs and Constraints

XDD Descriptions can model:



Queries

Doc./Data Validity Checking

Document Transformation

XML Database

- Extensional Database
- Integrity Constraints
- Intensional Database

ET … ET

Query Results

Doc./Data Validity Results

Transformed Documents

XML Database

- Extensional Database
- Integrity Constraints
- Intensional Database

Conclusions

Conclusions

- SW and DB are related, each can contribute to the other
- One of the most important enabling technologies for SW is *ontology*
- Ontology requires an expressive language with efficient computational mechanism
- Such a language can be applied to DB, e.g., conceptual modeling, query processing, schema/data integration, multimedia annotation, DW metadata, data mining
- SWRL = OWL + XMLized subset of Horn logic
- OWL over XDD provides a succinct, expressive OWL+Rules language with efficient computational mechanism