

INFS3101/7100 Ontology and the Semantic Web

Module 5 Complex Objects

Last lecture: Key Terms



- ❖ To represent an ontology need an **ontology representation language**
- ❖ Ontologies can be packaged in modules.
- ❖ A given ontology can be represented in many ways
- ❖ Some ontologies are huge and complex
- ❖ Some ontologies include individuals
- ❖ Some ontologies need complex rules in their representation

Ontology versus model

- ❖ At bottom, an ontology is a data model
- ❖ We already know lots about data modeling
- ❖ What is different?
 - Describes a universe of discourse shared by several systems, therefore conceptually outside of any
 - Only shared objects are modeled, so ontology must be integrated with operational models
 - Assembled from components, so maintenance issues
 - Instances often included, so representation issues

Integrate outside with inside

- ❖ Public face is ontology, resides externally
- ❖ Operational system uses local data model
 - Ontology resides in ontology server
 - Must reconcile local model with ontology
 - Called committing to the ontology
- ❖ Operational data model and procedures must track ontology version changes
 - EG e-books become part of the ontology

Ontology uses standard components

- ❖ EG bookseller world
 - Books-in-Print provides products and descriptions, including ISBN identifiers
 - Secure Electronic Transaction (SET) provides payment services
 - Message types and content come from EDI
 - Bookseller world agency provides specific facilities, ties it all together
- ❖ Component limits available facilities
- ❖ Impact of new version of a component

Ontologies often include instances

- ❖ Operational data models generally schema-only
- ❖ Ontology often includes instances
 - EG Books-in-print ontology includes book instances as well as types
 - Periodic table includes both types and instances
 - As do music, video catalogs
 - And gossip services
- ❖ Instances must be integrated with operational data models
 - Representation issues
- ❖ Some applications have individuals outside the class system
 - EG a credit risk application may take an individual application object and assign it to a class.

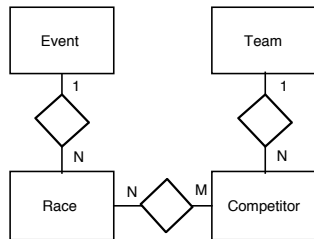
Example

- ❖ I am a person
- ❖ Most people call me Bob
- ❖ My mother calls me Robert
- ❖ The University knows me by a payroll number
- ❖ And also by a student number
- ❖ The Tax Department knows me by tax file no.
- ❖ Queensland Transport by my driver's licence no.
- ❖ All these are representations of the same thing
- ❖ For two to interoperate they must make a correspondence among their representations

Complex Objects

- ❖ Different players must agree on which object is which. **Problem of identity.**
- ❖ Shared objects are often complex: wholes with parts
 - Academic record has results for individual courses
 - A piece of machinery is a whole with parts
 - So is a suite of software
 - Or a sporting event
 - Need to know which parts are associated with which wholes **problem of unity**

Example



- ❖ R(EventID, RaceID, TeamID, CompID)

Identity

- ❖ Identity important in modeling generally
- ❖ Relational systems identify objects by keys
 - Key is a combination of values of attributes that is unique for each object
- ❖ Often use identifiers: product no, student no.
- ❖ Not generally useful in interoperating systems because identifier is private
 - Recall the tax office example or example of me
- ❖ Need candidate keys combining public attributes
 - Name and date of birth
 - Product description: Citric acid 99.9% purity in 200 litre drums

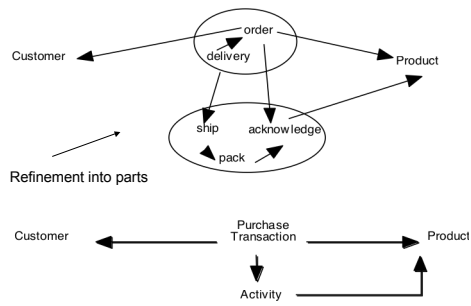
Unity

- ❖ If an object has parts, how do we know which parts belong to which whole?
- ❖ By values of properties. Properties included have the metaproperty **unity**.
 - Sometimes by foreign keys
 - Parts of a student's program record
 - Sometimes they are just in the same list
 - Test cricket team
 - Property here is "being on the list"

Lexical vs Logical Criteria

- ❖ There are different ways to specify unity criteria
- ❖ **Logical criterion**
 - All parts of a whole satisfy (possibly several) predicates
 - All parts of a student record of study are functionally dependent on student number
- ❖ **Lexical criterion**
 - All parts in the same container
 - All parts on same sheet of paper
 - All parts of a student record are attributes in the same tuple
 - The players for a test team are on the list of selected players

Identity and Unity Interact



INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

13

Identity and Unity Interact

- ❖ *Order* and *delivery* part of *purchase transaction*
- ❖ *Ship, pack, acknowledge* part of *activity*
- ❖ One way is to identify parts by adding part identifiers to the identifier for the whole. Also gives a unifying relation.
- ❖ But here the whole does not exist until all the parts exist. Can name the whole by one of the parts (this is called *metonymy*).
- ❖ In this case the unifying relation can be foreign key dependencies.

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

14

Use of Independent Classes

- ❖ There are generally some classes which are the targets of dependencies but never the source
 - Customer, product
- ❖ There are the more stable parts of the system. Called independent classes. The other classes are dependent.
- ❖ Dependent classes can be identified by the independent classes they depend on.
 - ER weak entities work like this

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

15

Unity and Identity Interact

- ❖ Some instances are individuals
 - Customers
- ❖ But some are better thought of as types
 - Products
 - There are lots of instances of
 - Citric acid 99.9% pure 400 litre drums
 - Mazda 3 SP23 black automatic
 - Panasonic Viera plasma television
 - Staedtler stick 430M red pen
 - The instances are identical

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

16

Instances and Individuals

- ❖ How do we distinguish instances of these types?
 - Mazda 3 SP23 black automatic
 - Panasonic Viera plasma television
- ❖ These generally have serial number plates attached to them.
- ❖ But Staedtler stick 430M red pens are indistinguishable.
- ❖ And Citric acid 99.9% pure 400 litre drums is very strange. If you empty them into a tank, how do you tell which acid in the tank came from which drum?

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

17

Instances and Individuals

- ❖ Citric acid lacks unity. Can't tell which part belongs to which whole. If no unity, no identity. Called a *bulk type*.
- ❖ Pens have unity, are anonymous (no identity). So are treated as bulk.
- ❖ Cars and plasma TVs have both unity and identity (the serial number plates). Individual has both unity and identity. Type whose instances are individuals called *countable type*.

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

18

Bulk Types

- ❖ A container can hold some of a bulk type.
- ❖ If the container is identified, the contents have a pseudo-identity. But drums of citric acid may be anonymous, so don't give pseudo-identity.
- ❖ Orders etc are containers.
- ❖ Value is a bulk type. An invoice or account is a container.
- ❖ Identity depends on context. "This" drum is identified indexically as long as you can keep track of it. (*Indexical* - this, that, I, my, you, your, here, now)

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

19

Identifying Classes

- ❖ How do we identify a class?
- ❖ Brute facts often by predicates
 - If it looks like a duck and quacks like a duck, then it's a duck.
- ❖ But classes of institutional facts in information systems are generally identified lexically.
 - An invoice has "Invoice" written on it, while a credit note has "credit note" written on it.
 - Records stored in the "Student" table are records of students.

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

20

Identifying a system

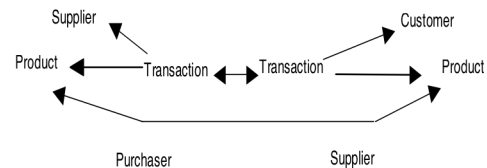
- ❖ A system has lots of interacting parts, each of which is identified.
- ❖ The system as a whole is often identified implicitly, indexically (this one, the system running on these computers, etc.)
- ❖ We could attach a system identifier to all the parts (called a terminal object by some people)
- ❖ But generally don't because within a single system we don't need to.

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

21

Interoperating Systems



Need to coordinate identifiers for Product and Transaction
Each system represented by instance in the other, so each system
Needs to be identified. Expanded context changes identification requirements

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

22

Bilateral Identity Doesn't Scale

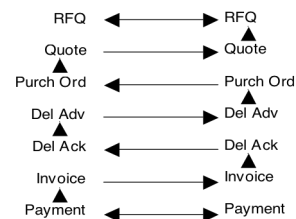
- ❖ I am a person
- ❖ Most people call me Bob
- ❖ My mother calls me Robert
- ❖ The University knows me by a payroll number
- ❖ And also by a student number
- ❖ The Tax Department knows me by tax file no.
- ❖ Queensland Transport by my driver's licence no.
- ❖ If dozens or hundreds of players, doesn't scale
- ❖ Need global identification scheme, like ISBN
- ❖ Need roles to perform the speech act of assigning the global identifier
- ❖ Can an object have more than one global identifier?
- ❖ Is this a problem?
- ❖ If so, how to control?

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

23

Transactions



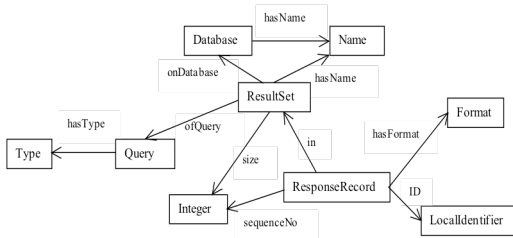
- ❖ Messages are speech acts
- ❖ Both sides keep copies of institutional facts
- ❖ Whole identified metonymically

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

24

Z39.50 Comment



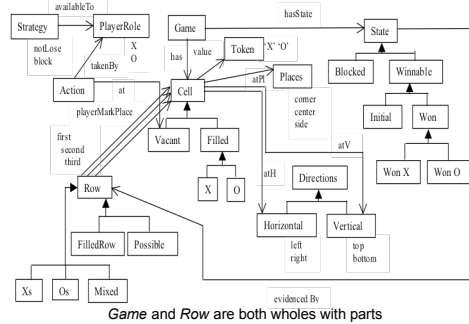
❖ Result set a whole with parts

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

25

Tic Tac Toe comment



Game and Row are both wholes with parts

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

26

Summary: Key Terms



- ❖ Complex objects need both **identity** and **unity**. Identification and unification can be **logical** or **lexical**. **Independent** classes can help identify **dependent** classes. A whole can be identified **metonymically**. Identity and unity depend on **context**. Can be **indexical**.
- ❖ **Countable** types have both identity and unity. **Bulk** types lack one. **Containers** can give **pseudo-identity** to a bulk type.

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

27

Resources

- ❖ **Essential**
 - Notes chapter 5 Complex Objects
- ❖ **Further**
 - Gangemi et. al. (2001) Understanding top-level ontological distinctions *Proc. of IJCAI 2001 workshop on Ontologies and Information Sharing* on course web site

INFS3101/INFS7100 week 4, 22 November, 2005

Bob Colomb

28