ITCS 3160

Jing Yang
2010 Fall

Class 16: Object and Object-Relational Databases (ch.11)

References

Slides come from:

• Michael Grossniklaus, Moira Norrie (ETH Zürich): Object-Oriented Databases (Version 2010, lecture notes)
• Beat Signer (Vrije Universiteit Brussel): Introduction to Databases Object and Object-Relational Databases (lecture notes).
Literature


Object-Relational Impedance Mismatch

- Interfacing with SQL is not straightforward
  - data has to be converted between host language and SQL due to the *impedance mismatch*
  - ~30% of the code and effort is used for this conversion!

- The problem gets even worse if we would like to use an object-oriented host language
Object-Oriented Databases

- ODBMSs use the same data model as object-oriented programming languages
  - no object-relational impedance mismatch due to a uniform model
- An object database combines the features of an object-oriented language and a DBMS (*language binding*)
  - treat *data as objects*
    - object identity
    - attributes and methods
    - relationships between objects
  - *extensible type hierarchy*
    - user-defined types, abstract data types
    - single or multiple inheritance
    - overloading, overriding, late binding
  - *declarative query language*

Defining Object-Oriented Databases

- Diverse focus of object-oriented database systems
  - making object-oriented programming languages persistent
  - managing and storing object data
- Many attempts to define object-oriented databases
- The object-oriented database manifesto
  - 13 mandatory features
  - 5 optional characteristics
  - 4 open choices
Complex objects

1. Complex objects
2. Object identity
3. Encapsulation
4. Types and classes
5. Type and class hierarchies
6. Overriding, overloading and late binding
7. Computational completeness
8. Extensibility
9. Persistence
10. Efficiency
11. Concurrency
12. Reliability
13. Declarative query language

Complex objects

- Complex object formed from simpler ones by constructors
- Constructors: tuple (record), set, bag, list, and array
  - Minimum set of constructors: set, list, and tuple
  - Sets: representing collections from the real world
  - Tuples: representing properties of an entity
  - Lists or arrays: capturing order, which occurs in the real world
- Constructor orthogonality: any constructor should apply to any object
- Appropriate operators must be provided for dealing with such objects
Object identity and equality

• Every object has unique and immutable object identifier (OID)
• Sharing of objects through references
• Two objects are identical if they have the same OID
• Two objects are equal if they have the same state

Encapsulation

• Object consists of interface and implementation
• Interface defines signatures of public methods
• Implementation includes object data and methods
• Object state is only modified through public methods
• Object data structure may be exposed for ad-hoc queries
Types and Classes

“systems should offer some form of data structuring mechanism, be it classes or types”

• Types
  – Summarizes common features of a set of objects with the same characteristics
  – corresponds to the notion of an abstract data type
  – data part describes object structure
  – operation part describes object behavior
  – separation of interface and implementation
  – used to check correctness of programs at compile time

• Object classes
  – Used to create and manipulate objects at run time
  – object factory creates new objects
  – The object warehouse keeps the set of objects that are instances of the class
  – Warehouse can be updated and passed

Generalization Hierarchies

• Advantages
  – powerful modeling tool
  – guarantee semantic complexity
  – reuse of specification and implementation

• Inheritance
  – objects of subclass belong automatically to superclass
  – attributes and methods are inherited from superclass
  – subclass can introduce new attributes and methods

• Migration between classes
  – move objects between hierarchy levels
  – object specialization (↓) and generalization (↑)
Generalization Hierarchies (cont’d)

- Substitution inheritance
  - subtype has more operations than supertype
  - subtype can be substituted where supertype is expected
  - based on behavior rather than values
- Inclusion inheritance (Classification)
  - every object of subtype is also object of supertype
  - Example: filled-square is a subclass of square
- Constraint inheritance
  - special case of inclusion inheritance
  - subtype is expressed by constraint on supertype
  - Example: teenager is a subclass of person
- Specialization inheritance (Typing)
  - subtype objects contain more specific information
  - Example: employee and person

Overriding, Overloading and Late Binding

- Method overriding
  - method is redefined in subtype
  - guarantees specialization of methods
  - preserves uniform method interface
- Method overloading
  - effect caused by method overriding
  - various version of a method can exist in parallel
- Late binding
  - appropriate version of overloaded method selected at run time
  - also known as virtual method dispatching
Computational Completeness and Extensibility

• Computational completeness
  – requirement for the method implementation language
  – any computable function can be expressed
  – can be realized through connection with existing language
    “SQL is not complete”
• Extensibility
  – database has a set of predefined types
  – developers can define new types according to requirements
  – no usage distinction between system and user types

Durability and Efficiency

• Persistence
  – data has to survive the program execution
  – orthogonal persistence: each object is allowed to be persistent
  – implicit persistence: user should not have to explicitly move or copy data to make it persistent.
• Secondary storage management
  – index management
  – data clustering
  – data buffering
  – access path selection
  – query optimization
Concurrent Control and Recovery

• Concurrency
  – management of multiple users concurrently interacting
  – atomicity, consistency, isolation and durability
  – serializability of operations

• Reliability
  – resiliency to user, software and hardware failures
  – transactions can be committed or aborted
  – restore previous coherent state of data
  – redoing and undoing of transactions
  – logging of operations

Declarative Query Language

• High-level language
  – express non-trivial queries concisely
  – text-based or graphical interface
  – declarative

• Efficient execution
  – possibility for query optimization

• Application independent
  – work on any possible database
  – no need for additional methods on user-defined types
Optional Characteristics and Open Choices

• Optional characteristics
  – multiple inheritance
  – type checking and inference
  – distribution
  – design transactions, long transactions, nested transactions
  – Versions

• Open choices
  – programming paradigm
  – representation system
  – type system
  – uniformity

ODBMS History

• First generation ODBMS
  – 1986
    • G-Base (Graphael, F)
  – 1987
    • GemStone (Servio Corporation, USA)
  – 1988
    • Vbase (Ontologic)
    • Static (Symbolics)

• Second generation ODBMS
  – 1989
    • Ontos (Ontos)
    • ObjectStore(Object Design)
    • Objectivity(Objectivity)
    • Versant ODBMS (Versant Object Technology)
  – 1989
    • The Object-Oriented Database System Manifesto
ODBMS History (cont’d)

- Third generation ODBMS
  - 1990
  - Orion/Itasca (Microelectronics and Computer Technology Cooperation, USA)
  - O2 (Altair, F)
  - Zeitgeist (Texas Instruments)
- Further developments
  - 1991
  - foundation of the Object Database Management Group (ODMG)
  - 1993
  - ODMG 1.0 standard
  - ...

ODMG

- Object Database Management Group (ODMG) founded in 1991 by Rick Cattell
  - standardization body including all major ODBMS vendors
- Define a standard to increase the portability across different ODBMS products
  - Object Model
  - Object Definition Language (ODL)
  - Object Query Language (OQL)
  - language bindings
  - C++, Smalltalk and Java bindings
**ODMG Object Model**

- Basic modeling primitives
  - *object*: unique identifier
  - *Literal* (values): no identifier
- An object's *state* is defined by the values it carries for a set of properties (attributes or relationships)
- An object's *behavior* is defined by the set of operations that can be executed
- Objects and literals are categorized by their *type* (common properties and common behavior)

**Objects**

- Atomic objects
  - user defined
- Collection objects
  - Set<\text{t}>  
  - Bag<\text{t}>  
  - List<\text{t}>  
  - Array<\text{t}>  
  - Dictionary<\text{t,v}>  
- Structured objects
  - Tuples, Date, Interval, Time, Timestamp
Relationships

• *One-to-one, one-to-many or many-to-many relationships with referential integrity maintained by the system*

```cpp
class Assistant {
    ...
    relationship set<ExerciseGroup> leads
        inverse ExerciseGroup::isLeadBy;
    ...
}

class ExerciseGroup {
    ...
    relationship Assistant isLeadBy
        inverse Assistant::leads;
    ...
}
```

Behavior

• Behavior is specified as a set of *operation signatures*

• An operation signature defines
  – name of the operation
  – names and types of arguments
  – type of return value
  – names of exceptions
Inheritance of Behavior

• A subtype may
  – define new behavior in addition to the one defined in its supertypes
  – refine a supertype’s behavior

Inheritance of State and Behavior

• Keyword EXTENDS
  – A subclass inherits all the properties and behavior of its superclass

```csharp
class Professor extends Employee {  
  attribute enum Type{assistant, full, ordinary} rank;  
  relationship worksFor
    inverse Department:hasProfessors;  
  relationship set<Lectures> teaches  
    inverse Session::isTaughtBy;  
}
```
"ODMG 4.0" Standard

• After the ODMG 3.0 standard the group disbanded
  – ODMG Java language binding formed basis for the *Java Data Objects (JDO) specification*
• The OMG *Object Database Technology Working Group (ODBT WG)* was founded in 2005 due to the new interest in object databases
• ODBT WG is now working on a fourth version of an object database standard

Object-Oriented Databases...

• Faster access than RDBMS for many tasks
• However, object databases lack a formal mathematical foundation!
Object-Relational Mapping

- "Automatic" mapping of object-oriented model to relational database
  - developer has to deal less with persistence-related programming
- Hibernate
  - mapping of Java types to SQL types
  - generates the required SQL statements behind the scene
  - standalone framework
- Java Persistence API (JPA)
  - Enterprise Java Beans Standard 3.0
    - use annotations to define mapping
    - javax.persistence package

Object-Relational Databases

- The object-relational data model extends the relational data model
  - introduces complex data types
  - object-oriented features
  - extended version of SQL to deal with the richer type system
- Complex data types
  - new collection types including multiset and arrays
  - attributes can no longer just contain atomic values (1NF) but also collections
  - nest and unnest operations for collection type attributes
  - ER concepts such as composite attributes or multivalued attributes can be directly represented in the object-relational data model
Object-Relational Databases ...

- Since SQL:1999 we can define user-defined types
- Type inheritance can be used for inheriting attributes of user-defined types