

Object-Oriented Databases

ODMG Standard

- Object Model, Object Definition Language, Object Query Language
- Programming Language Bindings
- Outlook



Development of OODBMS

- Many systems closely related to programming languages
 - Versant, Ontos, ObjectStore, Objectivity (C++)
 - GemStone (Smalltalk)
- Early versions had no query language support
 - ObjectStore had limited selection-based queries
- O₂ developed at INRIA (France) with large funding from European research projects
 - took more of a database approach
 - intended to be language independent
 - lot of research on query languages
 - interests also in interface and development tools support

Standards for Object Data Management

- Object Management Group (OMG)
 - architectures and tools to develop object-oriented systems
 - distributed object management
 - best known for Unified Modeling Language (UML)
- Object Data Management Group (ODMG)
 - data management support
 - complementary to OMG
 - ODMG data model based on OMG object model

Object Data Management Group (ODMG)

- ODMG formed very early in development of OODBMS
- Informal standards body involving all major vendors
 - initiated in 1991 by Rick Cattell of SunSoft
 - initially ODMG comprised five people from OODBMS vendors
- Promote portability and interoperability across products
- Not developing a standard OODBMS product
 - products will vary in terms of languages, tools, interfaces, performance, etc.
 - products may be tailored to application domains, e.g. version management for Computer-Aided Software Engineering (CASE)

ODMG Standard

- Object Model
- Object Definition Language (ODL)
- Object Query Language (OQL)
- Language bindings
 - C++ Binding
 - Smalltalk Binding
 - Java Binding

ODMG Object Model

- Based on the OMG object model
- Basic modelling primitives
 - object unique identifier
 - literal no identifier
- Object state defined by the values carried for a set of properties, i.e. attributes or relationships
- Object behaviour defined by the set of operations that can be executed
- Objects and literals are categorised by their type which defines common properties and common behaviour

Types

- Specification
 - properties, i.e. attributes and relationships
 - operations
 - exceptions
- Implementation
 - language binding
 - a specification can have more than one implementation

Type Specifications

■ Interface

- defines only abstract behaviour
- `interface Employee {...};`

■ Class

- defines both abstract behaviour and abstract state
- `class Person {...};`

■ Literal

- defines abstract state
- `struct Complex { float real; float imaginary; };`

Type Implementation

- Representation
 - data structure
 - derived from type's abstract state by the language binding
- Methods
 - procedure bodies
 - derived from type's abstract behaviour by the language binding
 - also private methods with no counterpart in specification

Subtyping and Inheritance

- Two types of inheritance relationships
- IS-A relationship
 - inheritance of behaviour
 - multiple inheritance, name overloading disallowed
 - `interface Professor : Employee {...};`
- EXTENDS relationship
 - inheritance of state and behaviour
 - single inheritance
 - `class Emp1Pers extends Person : Employee {...};`

Extents

- Extent of a type is the set of all active instances
 - assume class **Person**
 - extent of class **Person** would be the current set of all person objects in the data management system
- Extents can be maintained automatically

Collections

- Supports both collection objects and collection literals
 - set unordered, no duplicates
 - bag unordered, duplicates
 - list ordered, elements can be inserted
 - array ordered, elements can be replaced
 - dictionary maps keys to values
- Collection objects
 - Set<t>, Bag<t>, List<t>, Array<t>, Dictionary<t,v>
- Collection literals
 - set<t>, bag<t>, list<t>, array<t>, dictionary<t,v>

Collections

- Subset containment relation defined only over sets
- Operations union, intersection and difference defined only over sets and bags
- No constraints over collections

Collections

```
interface Collection : Object {
    exception InvalidCollectionType{};
    exception ElementNotFound{ Object element; };
    boolean    is_empty();
    ...
    boolean    contains_element(in Object element);
    void       insert_element(in Object element);
    void       removes_element(in Object element)
                raise (ElementNotFound);
    ...
    Iterator   create_iterator(in boolean stable);
    ...
    boolean    query(in String OQL_predicate,
                    inout Collection result);
};
```

Sets

```
class Set : Collection {
    attribute set<t> value;
    Set      create_union(in Set other_set);
    Set      create_intersection(in Set other_set);
    Set      create_difference(in Set other_set);
    boolean  is_subset_of(in Set other_set);
    boolean  is proper subset of(in Set other set);
    ...
};
```

Bags

```
class Bag : Collection {  
    attribute    bag<t> value ;  
    unsigned long occurrences_of(in Object element) ;  
    Bag         create_union(in Bag other_bag) ;  
    Bag         create_intersection(in Bag other_bag) ;  
    Bag         create_difference(in Bag other_bag) ;  
} ;
```

Object Definition Language (ODL)

- Specification language to define object type interfaces
- Should support all semantic constructs of object model
- Should be programming language independent
- Compatible to OMG Interface Definition Language (IDL)
- Should be extensible and practical

ODL Example

```
class Article extends Publication
(
  extent Articles
)
{
  exception IllegalPageNumber{ unsigned short pageNumber };
  attribute unsigned short beginPage;
  attribute unsigned short endPage;
  unsigned short getBeginPage();
  void setBeginPage(in unsigned short beginPage)
    raises (IllegalPageNumber);
  unsigned short getEndPage();
  void setEndPage(in unsigned short endPage)
    raises (IllegalPageNumber);
};
```

Relationships

```
class Author {  
    ...  
    relationship set<Publication> authors  
        inverse Publication::authoredBy;  
    ...  
}  
  
class Publication {  
    ...  
    relationship list<Author> authoredBy  
        inverse Author::authors;  
    ...  
}
```

system maintains referential integrity

Persistence

- Persistence by reachability
- Database gives access to global names
 - explicitly named root objects
 - types defined in schema
 - named extents of types

Other Concepts Supported

- Database operations
- Locking and concurrency control
- Transactions
- Access to metadata
- Structured literals and objects for
 - dates
 - times
 - timestamps
 - intervals
 - ...

Object Query Language (OQL)

- Based on ODMG Object Model
- Based on SQL-92
- Not computationally complete
 - rather simple to use query language
- No explicit update operators
 - but can invoke update operations on objects
- Path expressions to navigate complex objects
 - `person.spouse.address.street.name`

Example OQL Query

- Find the titles of all publications by "Moira C. Norrie" with more than one author that were published after 1995

```
select p.title
from   Authors a, a.authors p
where  a.name = "Moira C. Norrie"
and    p.year > 1995
and    count(p.authoredBy) >= 2
```

Collection Expressions

- Universal and existential quantification
 - `for all x in Students : x.studentId > 0`
- Aggregate operators
- Group-by operator
- Order-by operator
- Union, intersection and difference (only sets and bags)
- Conversion
- Flattening a collection of collections
- Special operations for lists

"ODMG 4.0"

- ODMG was dissolved in 2001
- OMG obtained rights to ODMG 3.0 in 2003
- OMG Object Database Technology Working Group (ODBTWG) was founded in 2005 in response to renewed interest in object-oriented databases
- First white paper proposes object calculus based on abstract store model and stack-based queries

Abstract Store Model

- All information represented as <subject, relation, object> triplets
- Hierarchy of store models
 - AS0 basic model
 - AS1 introduces static inheritance
 - AS2 introduces dynamic inheritance (roles)
- Abstract Object Query Language (AOQL)

AS2 Example

```

<S = { <i1, Person, { <i2, name, "Doe" >, <i3, born, 1948 > } >,
      <i4, Person, { <i5, name, "Poe" >, <i6, born, 1975 >} >,
      <i7, Person, { <i8, name, "Lee" >, <i9, born, 1951 >} >,
      <i13, Emp, { <i14, sal, 2500 >, <i15, worksIn, i127 > } >,
      <i16, Emp, { <i17, sal, 1500 >, <i18, worksIn, i128 >} >,
      <i19, Student, { <i20, no, 9 >, <i21, faculty, "CS" > } >
    },
C = { <i40, PersonClass, { <i41, age, (age code)> } >,
      <i50, EmpClass, { <i51, changeSal, (changeSal code) >,
                      <i52, netSal, (netSal code)>} >,
      <i60, StudentClass, { <i61, avgScore, (avgScore code)> } >
    },
R = {i1, i4, i7, i13, i16, i19}, CC = {},
SC = { < i1, i40>, < i4, i40>, < i7, i40>, < i13, i50>,
      < i16, i50>, < i19, i60> },
SS = { < i13, i4>, < i16, i7>, < i19, i7> } >

```

Literature

- R. G. Cattell, Douglas K. Barry, Mark Berler, Jeff Eastman, David Jordan, Craig Russell, Olaf Schadow, Torsten Stanienda, Fernando Velez (Editors): **The Object Data Standard: ODMG 3.0**, *Morgan Kaufmann 2000*
- OMG Object Database Technology Working Group: **Next-Generation Object Database Standardization**, *White Paper, September 2007*

Next Week

ObjectStore and Objectivity/DB

- Application Development
- Model of Persistence
- Advanced Features

