

Object-Oriented Database Management
Chapter 13

Class 09: Object-Oriented Database Management 1

Object DBMSs

- Underlying concepts:
 - Freely sharing data across processing routines creates unacceptable data dependencies
 - All software should be constructed out of standard, reusable components whenever possible
- Beginnings in the 1970s, but no significant commercial use until the 1990s

Class 09: Object-Oriented Database Management 2

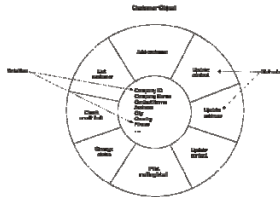
Advanced Database Applications

- Relational DBMSs have proven inadequate for applications that store complex data structures:
 - Computer-aided design (CAD)
 - Computer-aided manufacturing (CAM)
 - Computer-aided software engineering (CASE)
 - Office information systems and multimedia systems
 - Digital publishing
 - Geographic information systems (GIS)

Class 09: Object-Oriented Database Management 3

Anatomy of an Object

- **Object:** a logical grouping of related data and programming logic that represents a real-world thing
 - Like a relational entity except programming logic is stored as an integral part



Class 09: Object-Oriented Database Management

Object Properties

- **Abstraction:** process of identifying essential aspects of an entity and ignoring unimportant properties (e.g. implementation details)
- **Encapsulation:** object contains both the data structure and the operations (methods) that can be used to manipulate it
- **Information Hiding:** separate the external aspects of an object from its internal details; provides data independence
- **Modularization:** object can be constructed and modified independent of the system, provided the public interface is not changed

Class 09: Object-Oriented Database Management

Objects and Attributes

- **Object:** A uniquely identifiable entity that contains both the attributes that describe the state of the "real world" object and the actions (methods) associated with it
 - **Attributes or Instance Variables** record the current state of the object
 - **Complex Attribute** can contain collections or **references**, representing a relationship between objects

Class 09: Object-Oriented Database Management

Object Identity

- **Object Identifier (OID):**
 - System generated
 - Unique to one object
 - Unique across entire system; primary key is unique only within a relation
 - Invariant; cannot be altered during its lifetime
 - Independent of the values of its attributes
 - Invisible to the user (ideally)

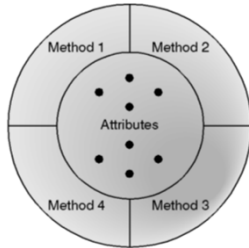
Advantages of OIDs

- Efficient: smaller than textual names or foreign keys
- Fast: points to an actual address
- Cannot be modified by the user
- Independent of content

Methods and Messages

- **Methods**
 - Define the behavior of an object
 - Can be used to change the object's state by modifying attribute values
 - Consists of a name and a body
- **Messages**
 - Means by which objects communicate
 - A request from one object (sender) to another object (receiver) to execute a method

Attributes and Methods



Class 09: Object-Oriented Database Management

10

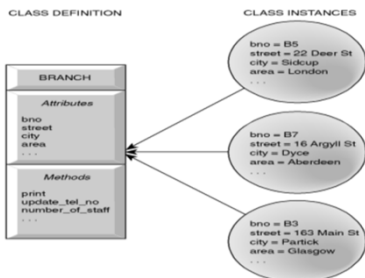
Classes

- **Class**: a grouping of objects that have the same attributes and respond to the same messages
 - Attributes and methods defined once for class
 - Objects in a class called *instances*

Class 09: Object-Oriented Database Management

11

Class Definition and Instances



Class 09: Object-Oriented Database Management

12

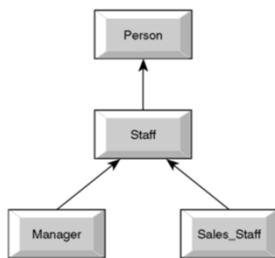
Inheritance, Subclasses and Superclasses

- **Inheritance:** allows one class to be defined as a special case of a more general case
- **Subclass:** special cases of a general class
- **Superclass:** general class
- **Generalization vs Specialization**
- Subclass inherits all properties of Superclass and may define its own

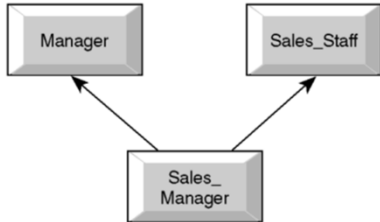
Inheritance Options

- **Multiple Inheritance:** subclass inherits from more than one superclass
 - Can lead to errors if method or variable names are ambiguous
- **Selective Inheritance:** limited properties inherited
- **Overriding:** Subclass redefines a property by using the same name as the inherited one

Single Inheritance



Multiple Inheritance



Class 09: Object-Oriented Database Management 16

Overloading and Polymorphism

- **Overloading** allows the name of a method to be reused within a class definition or across class definitions
- Single message performs different functions depending on which object receives it
- Permits **polymorphism** (having many forms)

Class 09: Object-Oriented Database Management 17

Complex Objects

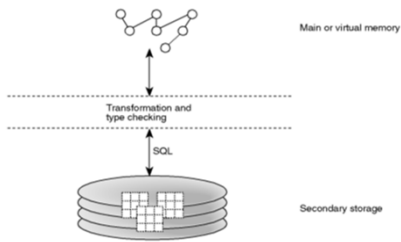
- Objects can be combined into **Complex Objects**
 - Each object can appear independently
 - Complex object can encapsulate the component objects
 - Complex objects can be included in other complex objects

Class 09: Object-Oriented Database Management 18

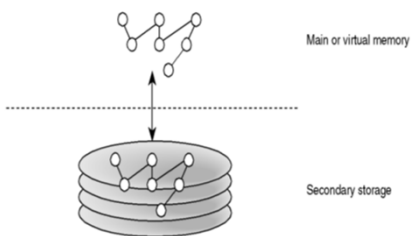
DBMS Generations

- Hierarchical and Network were first generation DBMSs
 - Relational are second generation DBMSs
 - Third generation are Object Oriented (OODBMS) and Object Relational (ORDBMS)
- Note: Objects are really not that new, OO began in the 1970's (Smalltalk at Xerox)

Two-level Storage Model (Relational DBMS)



Single-level Storage Model (OODBMS)

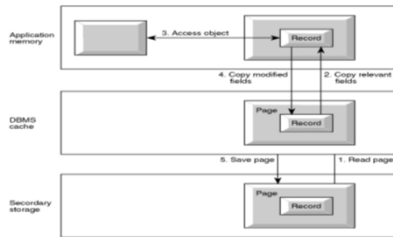


Accessing an Object

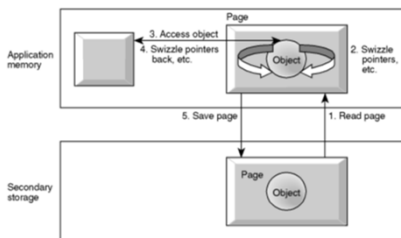
To access object, OODBMS must:

- Determine secondary storage page that contains the object
- Read page and copy into page cache
- Convert object (swizzle references, etc.)
- Application directly references object in memory
- OODBMS must convert object back before storing it

Steps in Accessing an Object, Conventional DBMS



Steps in Accessing an Object, OODBMS



Advantages of OODMSs

- Enriched modeling capabilities
- Extensibility
- No impedance mismatch with OO languages
- More expressive query language
- Support for schema evolution
- Support for long duration transactions
- Applicability to advanced database applications

Disadvantages of OODBMSs

- Lack of universal data model
- Lack of universally accepted standards
- Query optimization compromises encapsulation
- Locking at object level may impact performance
- Complexity
- Lack of support for views
- Lack of support for security

OO Design: Relationships

- 1:1 Relationships: Reference attribute added to each object
- 1:M Relationships: Reference attribute added to object on "many" side, attribute containing a set of references added on the "one" side
- M:N Relationships: Attribute containing set of references added to each object

OO Design: Referential Integrity

- Referential integrity requires that any referenced object must exist
- Techniques to handle referential integrity:
 - Do not allow user to explicitly delete objects
 - Allow user to delete objects when they are no longer required (referenced)
 - Allow user to modify or delete relationships when they are no longer required

OO Design: Behavioral Design

- **Constructor Methods:** generate new instances
- **Destructor Methods:** delete instances when no longer required
- **Access Methods:** return value of an attribute or set of attributes
- **Transform Methods:** change instance state
- **Identifying Methods:** Determine methods required to provide required functions

Object-Oriented Applications

- Object-Oriented (OO) applications are written in OO programming languages
- OO languages usually come with pre-defined class structures and methods, which are user extensible
- Some OO languages come with complete development environments
- OO programming can be done with or without an OO database

Object-Oriented Programming

- Messages are used as the vehicle for object interaction
- Message composed of :
 - Identifier of object to receive the message
 - Name of method to be invoked
 - Optional parameter(s)
- Methods are program code that define the *behavior* of the object

Class 09: Object-Oriented Database Management 31

Object-Oriented Languages

- Smalltalk
 - Developed in 1972 by Software Concepts Group (lead by Alan Kay) at Xerox PARC
 - Alan Kay coined phrase "object-oriented"
 - Includes:
 - Language
 - Programming Environment
 - Image File System (sort of an OO database)
 - Extensive Object Library

Class 09: Object-Oriented Database Management 32

Object-Oriented Languages

- Smalltalk (cont.)
 - Innovations included:
 - Bitmap display
 - Windowing system
 - Use of a mouse
 - Xerox never figured out how to market these innovations
 - Smalltalk is still around: www.smalltalk.org

Class 09: Object-Oriented Database Management 33

Object-Oriented Languages

- C++
 - Based on C Language (++ increments a variable by 1 in C)
 - Superset of C
 - Primarily developed by Bjarne Stroustrup at AT&T Bell Laboratories in 1986
 - Classes are implemented as user-defined types (called a *struct* in C)
 - Methods are implemented as member functions of a struct

Object-Oriented Languages

- C++ (cont.)
 - Object purists criticize C++ because programmer can totally ignore the object paradigm
 - C++ aficionados see that flexibility as a huge benefit

Object-Oriented Languages

- Java
 - A simple, portable, general-purpose OO language
 - Developed by Sun Microsystems in 1995
 - Took market by storm, largely because of support for Internet programming (applets)
 - An *interpretive* language (unlike Smalltalk and C++), with “just in time” compilers added to resolve performance issues

Object Persistence

- *Persistence*: the OO property that preserves the state of the object between executions of an application or server shutdown/startup
- Database often used to permanently store objects (OO, Relational, or O-R)
 - Objects must be loaded into memory before use
 - Loading is an indirect process

Object-Relational (O-R) Databases

- Evolved in response to mapping objects to relational databases
- Relational DBMS vendors such as Informix and Oracle added object capability to prevent loss of market share to OO DBMS vendors
- Pure OO databases lack ad hoc query support

O-R Databases

- Advantages
 - Complex data types directly supported
 - Methods can be stored as functions or stored procedures, facilitating reuse and sharing
 - Ad-hoc query capability is fully supported

O-R Databases

- Disadvantages
 - More complex than either OO or relational
 - Objects are *table centric*
 - Relational purists dislike loss of simplicity
 - Object purists not impressed with O-R
 - Lack of class structure and inheritance
 - Object applications are not as data centric as relational applications, so OO DBMS may be a better fit

ORDBMS Advantages

- Reuse comes from ability to extend DBMS to perform common functions centrally
- Sharing comes from embedding object functions in the server, making them available to all applications

ORDBMS Disadvantages

- Complexity and associated increased costs
 - Relational purists believe that essential simplicity and purity are lost on these types of extensions
- Object purists are not attracted to these extensions, arguing that ORDBMS is little more than user-defined data types
- Object applications are not as data centric as relational ones

Early ORDBMSs

- **Postgres ('Post Ingress')**
 - Provides predefined atomic types and allows users to add new atomic and structured types
 - Conversion procedures implemented in a high-level language such as C using a **define procedure** command
- **INHERITS** clause defines table inheritance
 - Multiple inheritance supported
 - Relation automatically inherits all attributes from parents
- Each relation has implicitly defined OID

Early ORDBMSs

- **INGRES Object Management Extension**
 - Allows additional data types and SQL functions to be created.
 - User defined data types can be used anywhere system defined data types can be used
 - New SQL functions can be used in queries and to manipulate system and user defined data types
 - Functions can be written in a language that conforms to 'C' conventions

Which One to Use?

