OO Data Modeling & OO DBMSs

- Why Object-Oriented Data Modeling?
- Objects, Classes, and Object Identities
- Data Encapsulation
- Relationships:
  - Inheritance
  - Aggregation
  - Association
- OO DBMSs in The Software Market Place
Why OO Data Modeling?

- Conventional data models (e.g., relational) are inadequate
  - Can’t model complex and unstructured data
  - Can’t model processes (dynamic behavior)
  - Doesn’t support reuse
What is Object Orientation?
Why OO?

Potential advantages of OO

Objects can be easier to understand and use

- objects are more intuitive than other data models - they are more like the ‘real world’ things

Reuse of objects - application development

- can lead to higher productivity

Easier to maintain applications - encapsulation of methods

- e.g., a business rule change may require a change to a single method of a single object vs. multiple programs
Objects & Classes

Object

- Conceptual entity (represents a real-world entity)
- Has structure (attributes)
- Exhibits different behavior and capabilities (methods) - able to interact with itself and other objects

Class

- Abstraction of a set of objects that share the same structure (attributes) and behavior (methods)
Objects & Class Definitions

Ex:

Object

Andrew Leung (name)
7/4/1942 (birth date)
3 Main St. (address)
Dragon Restaurant (employer)
Sichuan dishes (specialty)
Cook
Manage kitchen staff
Request supplies

Chef Class

Name: string
Birth-date: DATE
Address: ADDRESS
Employer: RESTAURANT
Specialty: string
Cook
Manage kitchen staff
Request supplies
Object Identity

Each object has a unique identity that is
- Assigned automatically when the object is created
- Destroyed when the object is removed
- Immutable (i.e., cannot be changed)

Major advantages
- Provides transparent identities
- Prevents hard-coding the “key” in applications
- Makes applications more portable and extendable
## Primary Key vs. Object Identities

<table>
<thead>
<tr>
<th>Primary Key:</th>
<th>Object Identity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value based</td>
<td>not value based</td>
</tr>
<tr>
<td>unique within a relation</td>
<td>unique within a database</td>
</tr>
<tr>
<td>user-defined</td>
<td>system assigned</td>
</tr>
<tr>
<td>mutable*</td>
<td>immutable**</td>
</tr>
<tr>
<td>reused</td>
<td>never reused</td>
</tr>
</tbody>
</table>

* *mutable* = *can be changed*

** *immutable* = *cannot be changed*
## Primary Key vs. Object Identities

<table>
<thead>
<tr>
<th>Primary Key</th>
<th>Object Identity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value based</td>
<td>not value based</td>
</tr>
</tbody>
</table>

Example Primary Key: ‘12345’ (CUSTOMERNUMBER)

Example Object Identity (OID): X20 (assigned by the system)
# Object Identities vs. Primary Key

<table>
<thead>
<tr>
<th>Primary Key</th>
<th>Object Identity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value based</td>
<td>not value based</td>
</tr>
<tr>
<td>unique within a relation</td>
<td>unique within a database</td>
</tr>
</tbody>
</table>

Example Primary Key: ‘12345’ (No other tuple in the CUSTOMER relation has this Customer Number)

Example Object Identity (OID): X20 (no other object in the database has this Object ID)
## Object Identities vs. Primary Key

<table>
<thead>
<tr>
<th>Primary Key</th>
<th>Object Identity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value based</td>
<td>not value based</td>
</tr>
<tr>
<td>unique within a relation</td>
<td>unique within a database</td>
</tr>
<tr>
<td>user-defined</td>
<td>system assigned</td>
</tr>
</tbody>
</table>

**Example Primary Key:** ‘12345’ (Customer Number - assigned by the user)

**Example Object Identity (OID):** X20 (assigned by the system)
## Object Identities vs. Primary Key

<table>
<thead>
<tr>
<th>Primary Key</th>
<th>Object Identity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value based</td>
<td>not value based</td>
</tr>
<tr>
<td>unique within a relation</td>
<td>unique within a database</td>
</tr>
<tr>
<td>user-defined</td>
<td>system assigned</td>
</tr>
<tr>
<td>mutable</td>
<td>immutable</td>
</tr>
</tbody>
</table>

Example Primary Key: ‘12345’ (Customer Number) - can be changed by the user

Example Object Identity (OID): X20 (generated by the system) - cannot be changed by the user
### Object Identities vs. Primary Key

<table>
<thead>
<tr>
<th>Primary Key</th>
<th>Object Identity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value based</td>
<td>not value based</td>
</tr>
<tr>
<td>unique within a relation</td>
<td>unique within a database</td>
</tr>
<tr>
<td>user-defined</td>
<td>system assigned</td>
</tr>
<tr>
<td>mutable*</td>
<td>immutable</td>
</tr>
<tr>
<td>reused</td>
<td>never reused</td>
</tr>
</tbody>
</table>

#### Example Primary Key:
User can give Customer Number ‘12345’ to another customer.

#### Example Object Identity (OID):
X20 is never used as the OID for another object.
Data Encapsulation

- The state of an object (the values of its attributes) is entirely encapsulated within the object.
- Encapsulation protects the object from undesirable changes.
- The "user object" must send a message to the owner object in order to get information.

What is your name?
Three kind of relationships between objects within an object-oriented database:

- Inheritance (also called “is a” relationship, or “class hierarchy”)
- Aggregation (also called “part of” relationship)
- Association (1:N, M:N, 1:1)
Class Hierarchy / Inheritance

**Superclass**

**Subclass**

**IS-A**

Single vs. multiple
Total vs. partial
Class Hierarchy / Inheritance

❖ **Single inheritance**
❖ exists when a class has only one immediate (parent) superclass above it in the hierarchy

❖ **Multiple inheritance**
❖ exists when a class has more than one superclass above it
❖ may inherit characteristics from more than one superclass

❖ **Note:** If more than one class/superclass in the hierarchy has a method with the same name, the method in the ‘lowest’ class in the hierarchy is used. (i.e., methods in ‘lower’ classes override methods in ‘higher’ classes)
Aggregation

Part-of relationship (aggregation)

The *whole* object owns the *part* objects
Association

- Has no "owning" relationship: all participants are equally important

- Cardinalities:
  - One-to-one
  - One-to-many
  - Many-to-many

(0, 2) (1, m) (1, 1) (0, m)

(0, 2) (1, m) (1, 1) (0, m)
Work-with   Supervise
The object model has been incorporated into DBMSs in two ways:

**“Pure” OODBMSs:**
- GemStone from Servio Logic;
- ONTOS from Ontos;
- ObjectStore from Object Design

**Object/Relational DMBSs: (Extension of relational DBMS):**
- OpenOODB from Hewlett Packard;
- UniSQL/X from UniSQL