Object-Oriented Databases (OODB)
Object Relational Databases (ORDBMS)

Douglas S. Kerr
CIS 671

Object Database Systems (Elmasri & Navathe, Ch 11-13)

• Object-Oriented Database Systems (ODBMS)
• Object-Relational Database Systems (ORDBMS)

Why? Briefly

• Object-Oriented Databases (OODBMS)
  – Add persistence to object-oriented programming languages

• Object Relational Databases (ORDBMS)
  – Add user-defined methods to relational databases
  – Allow grouping of relations into more complex “objects”

Review of Object-Oriented Concepts

Object

• State (value)
• Behavior (operations)
  – Signature or interface: operation name and arguments
  – Method or body: implementation
• Identified by unique Object Identifier (OID)

Class

• Group of similar objects
• Class hierarchies
  – Inheritance
• Persistence must be specified explicitly
  – Via entry point - a named object
  – Via reachability - ∃ sequence of references from named persistent object
Type Hierarchies and Inheritance
(EER — superclass/subclass relationship)

PERSON: Name, Address, Birthdate, Age, SSN

EMPLOYEE: subtype-of PERSON: Salary, HireDate, Seniority
STUDENT: subtype-of PERSON: Major, GPA

EMPLOYEE: Name, Address, Birthdate, Age, SSN, Salary, HireDate, Seniority
STUDENT: Name, Address, Birthdate, Age, SSN, Major, GPA

Object Data Management Group (ODMG)

- Object Model
- Object Definition Language (ODL)
- Object Query Language (OQL)
- Bindings to object-oriented programming languages
  - C++, Java, Smalltalk

ODMG Object Model

- Object
  - Identifier (unique OID)
  - Name (like primary key, optional)
  - Lifetime (persistent or transient)
  - Structure (atomic or collection object)
    - How to construct using type constructors

Example:
4-Input AND Gate Built From Three (3) 2-Input AND Gates

4-Input AND Gate in a Relational Database

Gate Type

<table>
<thead>
<tr>
<th>Gate Type</th>
<th>Description</th>
<th>AbsLogic</th>
<th>PowerControl</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>2AND</td>
<td>C = A &amp; B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4AND</td>
<td>C = A &amp; B &amp; C &amp; D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Gate Instance

<table>
<thead>
<tr>
<th>GateInstance</th>
<th>Gate</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT1</td>
<td>GT2</td>
<td>GT3</td>
</tr>
<tr>
<td>GT2</td>
<td>GT1</td>
<td>GT3</td>
</tr>
<tr>
<td>GT3</td>
<td>GT1</td>
<td>GT2</td>
</tr>
</tbody>
</table>

Wire Instance

<table>
<thead>
<tr>
<th>WireInstance</th>
<th>Wire</th>
<th>Pin1</th>
<th>Pin2</th>
<th>Pin3</th>
<th>Pin4</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>W2</td>
<td>GT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W2</td>
<td>W3</td>
<td>GT2</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td>W4</td>
<td>GT3</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td>W5</td>
<td>GT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W5</td>
<td>W6</td>
<td>GT2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W6</td>
<td>W7</td>
<td>GT3</td>
<td>C</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pin Type

<table>
<thead>
<tr>
<th>Pin Type</th>
<th>Pin</th>
<th>Description</th>
<th>AbsLogic</th>
<th>PowerControl</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT1</td>
<td>PT2</td>
<td>GT1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PT2</td>
<td>PT3</td>
<td>GT2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problems

- The relational model has no notion of a single object.
- There is spatial data present, e.g., location of a gate.
- There will be versions.
- There may also be configurations, i.e., variations of the same design.

Functions (or derived values) may be desired.

- Examples:
  - PowerCons(GateType) = \sum PowerCons(ComponentGateInstances)
  - AbsLoc(GateInstance) = transformation(ReflLoc(GateInstance), AbsLoc(GateType containing GateInstance))

Problems

Relational is Good For

- Fixed format data
- Transaction processing: simple short transactions
- Query processing
- Concurrency control
- Recovery from crashes

Problems With the Relational Model - 1

- Nonhomogeneous collection of design objects.
- Data Types: images, matrices, vectors; variable length.
- Temporal and/or spatial data.
- Many data types; few instances of each type.

Problems With the Relational Model - 2

- Schemas evolve during design.
- Long running transactions: "checkout a design".
- Versions; design log.

Problems With the Relational Model - 3

- Functions needed:
  - Making a design permanent,
  - Releasing design to production,
  - Archiving design,
  - etc.
- Library of design objects:
  - minimize redundancy.
Object-Oriented Concepts - 1

- Complex Objects:
  - sets,
  - bags,
  - lists,
  - arrays,
  - tuples.

Object-Oriented Concepts - 2

- Object Identity: identify by object id (oid) rather than by an attribute value, never changes.
- Encapsulation: Operations and Data available to user.
- Implementation hidden. No other operations available.

Object-Oriented Concepts - 3

- Types and/or Classes:
  - Type:
    - Interface and Implementation.
    - Types declared, checked at compile-time
  - Class:
    - Instances created at run-time

Object-Oriented Concepts - 4

- Class or Type Hierarchies:
  - Inheritance: both data and function
  - Example: student: undergraduate, graduate
- Overriding, overloading, and late binding:
  - Polymorphism

Object-Oriented Concepts - 5

- Computational Completeness:
- Extensibility:
  - Means to define new types.
  - User types not distinguished from system types.
- Persistence:
  - Not present in OO programming languages.
  - Should be implicit not explicit.

Object-Oriented Concepts - 6

- Secondary Storage Management:
- Concurrency:
- Recovery:
- Ad Hoc Query Facility:
Group of Closely Related Entities and Relationships

Text Example: Research Department with its locations, employees, and projects.

Departments
<table>
<thead>
<tr>
<th>DNAMER</th>
<th>DNUMBER</th>
<th>MGRRSN</th>
<th>MGRRSTARTDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>533445550</td>
<td>33445550</td>
<td>22-May-78</td>
</tr>
</tbody>
</table>

Dept Locations
<table>
<thead>
<tr>
<th>DNUMBER</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Bellaire</td>
</tr>
<tr>
<td>5</td>
<td>Sugarland</td>
</tr>
<tr>
<td>5</td>
<td>Houston</td>
</tr>
</tbody>
</table>

Projects
<table>
<thead>
<tr>
<th>PNAMER</th>
<th>PNNUMBER</th>
<th>DNUMB</th>
<th>LOCATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductX</td>
<td>1</td>
<td>5</td>
<td>Bellaire</td>
</tr>
<tr>
<td>ProductY</td>
<td>2</td>
<td>5</td>
<td>Sugarland</td>
</tr>
<tr>
<td>ProductZ</td>
<td>3</td>
<td>5</td>
<td>Houston</td>
</tr>
</tbody>
</table>

Managers
<table>
<thead>
<tr>
<th>MGRRSN</th>
<th>MGRRSTARTDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>333445555</td>
<td>22-May-78</td>
</tr>
</tbody>
</table>

Operations

• NumberOfEmployeesInDepartment
  (d: Department): integer

• EmployeeAge(e: Employee): integer

Development of Object-based Systems

• Object-Oriented Database Systems
  – An alternative to relational systems
  – Application domains where objects play central role
  – Heavily influenced by object-oriented programming languages
  – An attempt to add DBMS functionality to a programming language environment

Observations from Department Example

• Duplication of data
  – Employee “Wong” information
  – 3 locations “Bellaire”, “Sugarland”, “Houston”
• Give each “object” an “objectID” or OID
• In some sense a reversion to navigation
Development of Object-based Systems - cont.

- Object-Relational Database Systems
  - An attempt to extend relational databases
    • Broader set of applications
  - Provide bridge between relational and object-oriented systems