There is nothing worse than a sharp image of a fuzzy concept.

Ansel Adams

Software to be used in this Chapter...

- Star UML  http://www.mysql.com/products/workbench/
- Visual Paradigm:  http://www.visual-paradigm.com (CE)
Software to be used in this Chapter...

- Microsoft Visio has a UML-like set of diagramming templates for databases
- For Macs OmniGraffle has UML or spreadsheet templates that can be used for ER diagrams
Object-Oriented Modeling

• Becoming increasingly important as
  ▪ Object-Oriented and Object-Relational DBMS continue to proliferate
  ▪ Databases become more complex and have more complex relationships than are easily captured in ER or EER diagrams

Unified Modeling Language (UML)

• Combined three competing methods
• Can be used for graphically depicting
  ▪ Software designs and interaction
  ▪ Database
  ▪ Processes
Object Benefits

- Encapsulate both data and behavior
- Object-oriented modeling methods can be used for both database design and process design
  - Real-World applications have more than just the data in the database; they also involve the processes, calculations, etc. performed on that data to get real tasks done
  - OOM can be used for more challenging and complex problems

Unified Modeling Language (UML)

- UML methodology
  - Used extensively in software design
  - Many types of diagrams for various software design purposes
- UML class diagrams
  - Entity in ER corresponds to an object in UML
UML Classes

• A class is a named description of a set of objects that share the same attributes (states), operations, relationships, and semantics.
  • An object is an instance of a class that encapsulates state and behavior.
    • These objects can represent real-world things or conceptual things.
  • An attribute is a named property of a class that describes a range of values that instances of that class might hold.
  • An operation is a named specification of a service that can be requested from any of a class's objects to affect behavior in some way or to return a value without affecting behavior.

UML Classes

• Attributes have types.
• PK indicates an attribute in the primary key (optional) of the object.
• Methods have declarations: arguments (if any) and return type.
UML Relationships

- An relationship is a connection between or among model elements.
- The UML defines four basic kinds of relationships:
  - Association
  - Dependency
  - Generalization
  - Realization

UML Diagrams

- The UML defines nine types of diagrams:
  - activity diagram
  - class diagram
    - Describes the data and some behavioral (operations) of a system
  - collaboration diagram
  - component diagram
  - deployment diagram
  - object diagram
  - sequence diagram
  - State chart diagram
  - use case diagram
Class Diagrams

- A class diagram is a diagram that shows a set of classes, interfaces, and/or collaborations and the relationships among these elements.

Example: Bar Class

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Attributes</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar</td>
<td>PK Name: string</td>
<td>setName(n)</td>
</tr>
<tr>
<td></td>
<td>Addr: string</td>
<td>setAddr(a)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getName() : string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getAddr() : string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sellsBud() : boolean</td>
</tr>
</tbody>
</table>
Differences from Entities in ER

- Entities can be represented by Class diagrams
- But Classes of objects also have additional operations associated with them

Operations

- Three basic types for database
  - Constructor
  - Query
  - Update
Associations

- An association is a relationship that describes a set of links between or among objects.
- An association can have a name that describes the nature of this relationship. You can put a triangle next to this name to indicate the direction in which the name should be read.

Associations

- An association contains an ordered list of association ends.
  - An association with exactly two association ends is called a binary association
  - An association with more than two ends is called an n-ary association.
Associations: Unary relationships

Person

Employee

Associations: Binary Relationship

Employee

Parking Place

Product Line

Product

Student

Course

Associations: Unary relationships

Person

Employee

Associations: Binary Relationship

Employee

Parking Place

Product Line

Product

Student

Course
Associations: Ternary Relationships

```
Vendor * Supplies * Warehouse
```

Association Classes

```
Student --------| Registers-for | Course
```

```
Registration    ---------
Term           Grade
                  
CheckEligibility()
```

```
Computer Account
acctID
Password
ServerSpace
```

issues 0..1
Derived Attributes, Associations, and Roles

Student
- name
- ssn
- dateOfBirth
  - /age

Course Offering
- term
- section
- time
- location

Course
- crseCode
- crseTitle
- creditHrs

Registers-for
- * participant

Scheduled-for
- 1 /Takes

(Takes = currentDate – dateOfBirth)

 Derived attribute

 Derived role

 Derived association

Generalization

Employee
- empName
- empNumber
- address
- dateHired
- printLabel()

Hourly Employee
- HourlyRate
- computeWages()

Salaried Employee
- Annual Sal
- stockoption
- Contributepension()

Consultant
- contractNumber
- billingRate
- computeFees()
Example: Association

Comparison With E/R Multiplicities

E/R

UML

0..* 0..*

0..* 0..1

0..* 1..1
Association Classes

- Attributes on associations are permitted.
  - Called an *association class*.
  - Analogous to attributes on relationships in E/R.

Example: Association Class

```
<table>
<thead>
<tr>
<th>Bar</th>
<th>1..50</th>
<th>0..*</th>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sells</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>price: float</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Subclasses

- Like E/R, but subclass points to superclass with a line ending in a triangle.
- The subclasses of a class can be:
  - *Complete* (every object is in at least one subclass) or *partial*.
  - *Disjoint* (object in at most one subclass) or *overlapping*.

Example: Subclasses

```
<table>
<thead>
<tr>
<th>Beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: string</td>
</tr>
<tr>
<td>manf: string</td>
</tr>
</tbody>
</table>

triangle

<table>
<thead>
<tr>
<th>Ale</th>
</tr>
</thead>
<tbody>
<tr>
<td>color: string</td>
</tr>
</tbody>
</table>
```
Conversion to Relations

• We can use any of the three strategies outlined for E/R to convert a class and its subclasses to relations.
  1. E/R-style: each subclass’ relation stores only its own attributes, plus key.
  2. OO-style: relations store attributes of subclass and all superclasses.
  3. Nulls: One relation, with NULL’s as needed.

Aggregations

• Relationships with implication that the objects on one side are “owned by” or are part of objects on the other side.
• Represented by a diamond at the end of the connecting line, at the “owner” side.
• Implication that in a relational schema, owned objects are part of owner tuples.
Example: Aggregation

Compositions

- Like aggregations, but with the implication that every object is definitely owned by one object on the other side.
- Represented by solid diamond at owner.
- Often used for subobjects or structured attributes.
Example: Composition

Conversion to Relations

- We could store the awards of a beer with the beer tuple.
- Requires an object-relational or nested-relation model for tables, since there is no limit to the number of awards a beer can win.
Example: Composition

<table>
<thead>
<tr>
<th>Bar</th>
<th>1..1 Won 0..1</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: string</td>
<td></td>
<td>street: string</td>
</tr>
<tr>
<td>phone: int</td>
<td></td>
<td>city: string</td>
</tr>
</tbody>
</table>

Conversion to Relations

- Since a bar has at most one address, it is quite feasible to add the street, city, and zip attributes of Addr to the Bars relation.
- In object-relational databases, Addr can be one attribute of Bars, with structure.
Figure 7.16
The COMPANY conceptual schema in UML class diagram notation.
Chapter 10 Outline

- The Role of Information Systems in Organizations
- The Database Design and Implementation Process
- Use of UML Diagrams as an Aid to Database Design Specification
- Rational Rose: A UML-Based Design Tool
- Automated Database Design Tools

Practical Database Design Methodology and Use of UML Diagrams

- Design methodology
  - Target database managed by some type of database management system
- Various design methodologies
- Large database
  - Several dozen gigabytes of data and a schema with more than 30 or 40 distinct entity types
The Role of Information Systems in Organizations

- Organizational context for using database systems
  - Organizations have created the position of database administrator (DBA) and database administration departments
  - Information technology (IT) and information resource management (IRM) departments
    - Key to successful business management

The Role of Information Systems in Organizations (cont’d.)

- Database systems are integral components in computer-based information systems
- Personal computers and database system-like software products
  - Utilized by users who previously belonged to the category of casual and occasional database users
- **Personal databases** gaining popularity
- Databases are distributed over multiple computer systems
  - Better local control and faster local processing
The Role of Information Systems in Organizations (cont’d.)

- Data dictionary systems or information repositories
  - Mini DBMSs
  - Manage meta-data

- High-performance transaction processing systems require around-the-clock nonstop operation
  - Performance is critical

The Information System Life Cycle

- Information system (IS)
  - Resources involved in collection, management, use, and dissemination of information resources of organization
The Information System Life Cycle

- **Macro life cycle**
  - Feasibility analysis
  - Requirements collection and analysis
  - Design
  - Implementation
  - Validation and acceptance testing
  - Requirements collection and analysis

The Information System Life Cycle (cont’d.)

- The database application system life cycle: **micro life cycle**
  - System definition
  - Database design
  - Database implementation
  - Loading or data conversion
The Information System Life Cycle (cont’d.)

- Application conversion
- Testing and validation
- Operation
- Monitoring and maintenance

The Database Design and Implementation Process

- Design logical and physical structure of one or more databases
  - Accommodate the information needs of the users in an organization for a defined set of applications
- Goals of database design
  - Very hard to accomplish and measure
- Often begins with informal and incomplete requirements
Main phases of the overall database design and implementation process:

1. Requirements collection and analysis
2. Conceptual database design
3. Choice of a DBMS
4. Data model mapping (also called logical database design)
5. Physical database design
6. Database system implementation and tuning
The Database Design and Implementation Process (cont’d.)

- Parallel activities
  - Data content, structure, and constraints of the database
  - Design of database applications
- Data-driven versus process-driven design
- Feedback loops among phases and within phases are common

The Database Design and Implementation Process (cont’d.)

- Heart of the database design process
  - Conceptual database design (Phase 2)
  - Data model mapping (Phase 4)
  - Physical database design (Phase 5)
  - Database system implementation and tuning (Phase 6)
Phase 1: Requirements Collection and Analysis

- **Activities**
  - Identify application areas and user groups
  - Study and analyze documentation
  - Study current operating environment
  - Collect written responses from users

Phase 1 (cont’d.)

- **Requirements specification techniques**
  - Oriented analysis (OOA)
  - Data flow diagrams (DFDs)
  - Refinement of application goals
  - Computer-aided
Phase 2: Conceptual Database Design

- Phase 2a: Conceptual Schema Design
  - Important to use a conceptual high-level data model
  - Approaches to conceptual schema design
    - Centralized (or one shot) schema design approach
    - View integration approach

Phase 2: (cont’d.)

- Strategies for schema design
  - Top-down strategy
  - Bottom-up strategy
  - Inside-out strategy
  - Mixed strategy

- Schema (view) integration
  - Identify correspondences/conflicts among schemas:
    - Naming conflicts, type conflicts, domain (value set) conflicts, conflicts among constraints
  - Modify views to conform to one another
  - Merge of views and restructure
Phase 2: (cont’d.)

- Strategies for the view integration process
  - Binary ladder integration
  - N-ary integration
  - Binary balanced strategy
  - Mixed strategy

- Phase 2b: Transaction Design
  - In parallel with Phase 2a
  - Specify transactions at a conceptual level
  - Identify input/output and functional behavior
  - Notation for specifying processes

Phase 3: Choice of a DBMS

- Costs to consider
  - Software acquisition cost
  - Maintenance cost
  - Hardware acquisition cost
  - Database creation and conversion cost
  - Personnel cost
  - Training cost
  - Operating cost

- Consider DBMS portability among different types of hardware
Phase 4: Data Model Mapping (Logical Database Design)

- Create a conceptual schema and external schemas
  - In data model of selected DBMS
- Stages
  - System-independent mapping
  - Tailoring schemas to a specific DBMS

Phase 5: Physical Database Design

- Choose specific file storage structures and access paths for the database files
  - Achieve good performance
- Criteria used to guide choice of physical database design options:
  - Response time
  - Space utilization
  - Transaction throughput
Phase 6: Database System Implementation and Tuning

- Typically responsibility of the DBA
  - Compose DDL
  - Load database
  - Convert data from earlier systems
- Database programs implemented by application programmers
- Most systems include monitoring utility to collect performance statistics

Use of UML Diagrams as an Aid to Database Design Specification

- Use UML as a design specification standard
- Unified Modeling Language (UML) approach
  - Combines commonly accepted concepts from many object-oriented (O-O) methods and methodologies
  - Includes use case diagrams, sequence diagrams, and statechart diagrams
UML for Database Application Design

- Advantages of UML
  - Resulting models can be used to design relational, object-oriented, or object-relational databases
  - Brings traditional database modelers, analysts, and designers together with software application developers

Different Types of Diagrams in UML

- Structural diagrams
  - Class diagrams and package diagrams
  - Object diagrams
  - Component diagrams
  - Deployment diagrams
Different Types of Diagrams in UML (cont’d.)

- Behavioral diagrams
  - Use case diagrams
  - Sequence diagrams
  - Collaboration diagrams
  - Statechart diagrams
  - Activity diagrams

Figure 10.7
The use case diagram notation.
Different Types of Diagrams in UML (cont’d.)

**Figure 10.9**
The sequence diagram notation.

---

Different Types of Diagrams in UML (cont’d.)

**Figure 10.10**
The statechart diagram notation.
Modeling and Design Example: UNIVERSITY Database

Figure 10.11
A sample statechart diagram for the UNIVERSITY database.

Figure 10.12
A sequence diagram for the UNIVERSITY database.
Rational Rose: A UML-Based Design Tool

- **Rational Rose for database design**
  - Modeling tool used in the industry to develop information systems

- **Rational Rose data modeler**
  - Visual modeling tool for designing databases
  - Provides capability to:
    - **Forward engineer** a database
    - **Reverse engineer** an existing implemented database into conceptual design
Data Modeling Using Rational Rose
Data Modeler

- **Reverse engineering**
  - Allows the user to create a conceptual data model based on an existing database schema specified in a DDL file

- **Forward engineering and DDL generation**
  - Create a data model directly from scratch in Rose
  - Generate DDL for a specific DBMS

Data Modeling Using Rational Rose
Data Modeler (cont’d.)

- **Conceptual design in UML notation**
  - Build ER diagrams using class diagrams in Rational Rose

  **Identifying relationships**
  - Object in a child class cannot exist without a corresponding parent object

  **Non-identifying relationships**
  - Specify a regular association (relationship) between two independent classes
Data Modeling Using Rational Rose
Data Modeler (cont’d.)

- Converting logical data model to object model and vice versa
  - Logical data model can be converted to an object model
  - Allows a deep understanding of relationships between conceptual and implementation models

Data Modeling Using Rational Rose
Data Modeler (cont’d.)

- Synchronization between the conceptual design and the actual database
- Extensive domain support
  - Create a standard set of user-defined data types
- Easy communication among design teams
  - Application developer can access both the object and data models
Automated Database Design Tools

- Many CASE (computer-aided software engineering) tools for database design
- Combination of the following facilities
  - Diagramming
  - Model mapping
  - Design normalization

Automated Database Design Tools (cont’d.)

- Characteristics that a good design tool should possess:
  - Easy-to-use interface
  - Analytical components
  - Heuristic components
  - Trade-off analysis
  - Display of design results
  - Design verification
Automated Database Design Tools (cont’d.)

- Variety of products available
  - Some use expert system technology

<table>
<thead>
<tr>
<th>Company</th>
<th>Tool</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embarcadero Technologies</td>
<td>ER/Studio</td>
<td>Database modeling in ER and IDEFix</td>
</tr>
<tr>
<td></td>
<td>DBArtisan</td>
<td>Database administration and space and security management</td>
</tr>
<tr>
<td>Oracle</td>
<td>Developer 2000 and</td>
<td>Database modeling, application</td>
</tr>
<tr>
<td></td>
<td>Designer 2000</td>
<td>development</td>
</tr>
<tr>
<td>Persistence Inc.</td>
<td>PowerTier</td>
<td>Mapping from O-O to relational model</td>
</tr>
<tr>
<td>(Computer Associates)</td>
<td>Platinum ModelMart,</td>
<td>Data, process, and business component modeling</td>
</tr>
<tr>
<td></td>
<td>ERwin, BPwin, AllFusion Component Modeler</td>
<td></td>
</tr>
<tr>
<td>Popkin Software</td>
<td>Telelogic System Architect</td>
<td>Data modeling, object modeling, process modeling, structured analysis/design</td>
</tr>
<tr>
<td>Rational (IBM)</td>
<td>Rational Rose</td>
<td>Modeling in UML and application</td>
</tr>
<tr>
<td></td>
<td>XDE Developer Plus</td>
<td>generation in C++ and Java</td>
</tr>
<tr>
<td>Resolution Ltd.</td>
<td>XCase</td>
<td>Conceptual modeling up to code maintenance</td>
</tr>
<tr>
<td>Sybase</td>
<td>Enterprise Application Suite</td>
<td>Data modeling, business logic modeling</td>
</tr>
<tr>
<td>Visio</td>
<td>Visio Enterprise</td>
<td>Data modeling, design and reengineering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Visual Basic and Visual C++</td>
</tr>
</tbody>
</table>

Summary

- Six phases of the design process
  - Commonly include conceptual design, logical design (data model mapping), physical design
- UML diagrams
  - Aid specification of database models and design
- Rational Rose and the Rose Data Modeler
  - Provide support for the conceptual design and logical design phases of database