Database Systems

Object Databases

H. Turgut Uyar Şule Öğüdücü

2005-2012

License

Topics

Object-Orientation

- mismatch between the data model and the software model
  - data: relation, tuple, foreign key, ...
  - software: object, method, ...

Mismatch Example

Example (adding an actor to a movie - SQL definitions)

```
CREATE TABLE MOVIE (ID INTEGER PRIMARY KEY,
    TITLE VARCHAR(80) NOT NULL)

CREATE TABLE PERSON (ID INTEGER PRIMARY KEY,
    NAME VARCHAR(40) NOT NULL)

CREATE TABLE CASTING(
    MOVIEID INTEGER REFERENCES MOVIE,
    ACTORID INTEGER REFERENCES PERSON,
    PRIMARY KEY (MOVIEID, ACTORID)
)
```


Mismatch Example

Example (adding an actor to a movie - SQL operations)

```
INSERT INTO MOVIE(ID, TITLE)
  VALUES(110, 'Sleepy Hollow')

INSERT INTO PERSON(ID, NAME)
  VALUES(26, 'Johnny Depp')

INSERT INTO CASTING(MOVIEID, ACTORID)
  VALUES(110, 26)
```

Mismatch Example

Example (adding an actor to a movie - Java definitions)

```
public class Movie {
    ...
    private List<Person> cast;

    ...
    public void addActor(Person p) {
        this.cast.add(p);
    }
}
```


Mismatch Example

Example (adding an actor to a movie - Java operations)

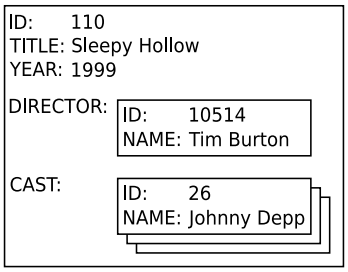
```
Movie m = new Movie("Sleepy Hollow", ...);
Person p = new Person("Johnny Depp", ...);
m.addActor(p);
```


Object Identifiers

- every object has an identifier
  - object identifiers don't change even if object attributes change
- different from primary key
  - primary key is visible (user-defined)
  - value of primary key can change
- correspond to references in programming languages
  - object identifiers can refer to other objects: *containment hierarchy*

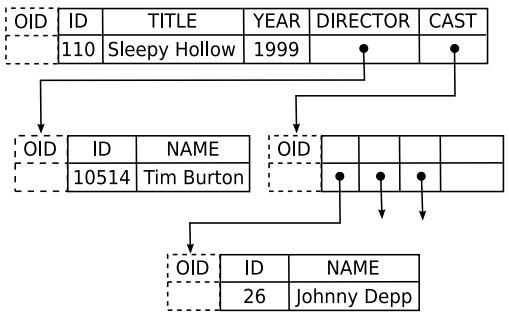
Containment Hierarchy Example

Example



Containment Hierarchy Example

Example



Object Databases

- persistent objects are stored as objects, not as relations
- write: object → internal format (serialization)
- read: internal format → object (deserialization)

db4o

- an object database system that can work embedded
- query using conditions
- query by example
  - create an object of the class to be queried
  - set the desired properties, leave the others blank
  - search for similar objects
- the objects to be updated or deleted have to be retrieved from the database (object identifier)

db4o Interface

- connecting to database (embedded mode):
  `Db4oEmbedded.openFile(filePath)` → `ObjectContainer`
- insert and update:
  `ObjectContainer.store(object)`
- delete:
  `ObjectContainer.delete(object)`

db4o Interface

- all instances of a class:
  `ObjectContainer.query(Class.class)` → `List<Class>`
- query by example:
  `ObjectContainer.queryByExample(Class prototype)`
  → `ObjectSet<Class>`

db4o Interface

- query condition: `Predicate<Class>`
- implement the `match` method:
  `public boolean match(Class object)`
- query:
  `ObjectContainer.query(Predicate<Class> predicate)`
  → `List<Class>`

db4o Examples

Example (connecting to the database)

```
ObjectContainer db = Db4oEmbedded.openFile(
    "imdb.db4o"
);
```


db4o Examples

Example (query: all movies)

```
List<Movie> movies = db.query(Movie.class);
for (Movie movie : movies) {
    ...
}
```

db4o Examples

Example (query by example: movies in 1977)

```
Movie prototype = new Movie(null);
prototype.setYear(1977);
ObjectSet<Movie> movies =
    db.queryByExample(prototype);
while (movies.hasNext()) {
    Movie m = movies.next();
    ...
}
```


db4o Examples

Example (query by condition: movies after 1977)

```
List<Movie> movies = db.query(
    new Predicate<Movie>() {
        public boolean match(Movie movie) {
            return movie.getYear() > 1977;
        }
});
```


db4o Examples

Example (insert)

```
Movie m = new Movie("Casablanca");
m.setYear(1942);
db.store(m);
db.commit();
```


db4o Examples

Example (update)

```
Movie prototype = new Movie("Casablanca");
ObjectSet<Movie> result =
    db.queryByExample(prototype);
Movie found = result.next();
found.setYear(1943);
db.store(found);
db.commit();
```


db4o Examples

Example (delete)

```
Movie prototype = new Movie("Casablanca");
ObjectSet<Movie> result =
    db.queryByExample(prototype);
Movie found = result.next();
db.delete(found);
db.commit();
```


References

Required Reading: Date

- Chapter 25: Object Databases

Object/Relational Mapping

- software is object-oriented
- database is relational
- map software components to database components

Example: Persist

- wraps a JDBC connection
- translates the object database interface into SQL statements

Persist Interface

- database connection: `Connection connection`
  `Persist(connection)` → `Persist`
- insert:
  `Persist.insert(object)`
- update:
  `Persist.update(object)`
- delete:
  `Persist.delete(object)`

Persist Interface

- query: all instances of a class
  `Persist.readList(Class.class)` → `List<Class>`
- query using SQL: similar to prepared statements
  `Persist.readList(Class.class, String query, params)`
  → `List<Class>`

Persist Examples

Example (database connection)

```
Connection connection =
    DriverManager.getConnection(jdbcURL);
Persist db = new Persist(connection);
```


Persist Examples

Example (query: all movies)

```
List<Movie> movies = db.readList(Movie.class);
for (Movie movie : movies) {
    ...
}
```

Persist Examples

Example (query using SQL: all movies in 1977)

```
List<Movie> movies = db.readList(Movie.class,
    "SELECT * FROM MOVIE WHERE (YEAR = ?)",
    1977);
```


Persist Examples

Example (insert)

```
Movie m = new Movie("Casablanca");
m.setYear(1942);
db.insert(m);
```


Persist Examples

Example (update)

```
List<Movie> movies = db.readList(Movie.class,
    "SELECT * FROM MOVIE WHERE (TITLE = ?)",
    "Casablanca");
Movie found = movies.get(0);
found.setYear(1943);
db.update(found);
```


Persist Examples

Example (delete)

```
List<Movie> movies = db.readList(Movie.class,
    "SELECT * FROM MOVIE WHERE (TITLE = ?)",
    "Casablanca");
Movie found = movies.get(0);
db.delete(found);
```


References

Required Reading: Date

- Chapter 26: Object/Relational Databases

XML

- XML is not a language itself
  - framework for defining languages
- XML-based languages
  - XHTML, DocBook, SVG, MathML, WML, XMI, ...
- XML-related languages
  - XPath, XQuery, XSL Transforms, SOAP, XLink, ...

XML Structure

- an XML document forms a *tree*
- nodes: *elements*
  - root node: *document element*
  - leaves: character data, self-closing elements
- opening/closing tags
- attributes

XML Example

Example (HTML)

```
<html>
<head><title>Foo Bar</title></head>
<body>
  <h1>Welcome to Foo Bar!</h1>
  <p>You can get more information from the
    <a href="http://www.foobar.net/">
      foobar page</a>.</p>
  <img src="logo.jpg" alt="Foo Bar logo" />
</body>
</html>
```


XML Example

Example (DocBook)

```
<book lang="en">
  <title>Foobar Report</title>
  <bookinfo>...</bookinfo>
  <chapter>...</chapter>
  <chapter>...</chapter>
  ...
</book>
```


XML Example

Example (DocBook)

```
  <bookinfo>
    <author>
      <firstname>John</firstname>
      <surname>Foobar</surname>
    </author>
    <date>2007</date>
  </bookinfo>
```


XML Example

Example (DocBook)

```
  <chapter>
    <title>Introduction</title>
    <section>
      <title>Description</title>
      <para>Foobar is ...</para>
    </section>
    ...
  </chapter>
```


XML Example

Example (movies)

```
<movies>
  <movie color="Color">
    <title>Usual Suspects</title>
    ...
  </movie>
  <movie color="Color">
    <title>Being John Malkovich</title>
    ...
  </movie>
  ...
</movies>
```

XML Example

Example (movies)

```
<movie color="Color">
  <title>Usual Suspects</title>
  <year>1995</year>
  <score>8.7</score>
  <votes>35027</votes>
  <director>Bryan Singer</director>
  <cast>
    <actor>Gabriel Byrne</actor>
    <actor>Benicio Del Toro</actor>
  </cast>
</movie>
```


XQuery

- XPath: selecting nodes and data from XML documents
- XQuery: XPath + update operations

XPath

- path of nodes to find: chain of location steps
  - starting from the root (absolute)
  - starting from the current node (relative)
  - location steps are separated by / symbols

Example

- `/movies/movie`
- `cast/actor` or `./cast/actor`
- `../../year`

Location Steps

- location step structure:
  `axis::node_selector[predicate]`
- axis: where to search
- selector: what to search
- predicate: under which conditions

Axes

- `child`: all children, one level (default axis)
- `descendant`: all children, recursively (shorthand: `//`)
- `parent`: parent node, one level
- `ancestor`: parent nodes, up to document element
- `attribute`: attributes (shorthand: `@`)
- `following-sibling`: siblings that come later
- `preceding-sibling`: siblings that come earlier
- ...

Node Selectors

- node tag
- node attribute
- node text: `text()`
- all children: `*`

XPath Examples

Example

- names of all directors:
  `/movies/movie/director/text()`
  `//director/text()`
- all actors in this movie:
  `./cast/actor`
  `.//actor`
- colors of all movies:
  `//movie/@color`
- scores of movies after this one:
  `./following-sibling::movie/score`

XPath Predicates

- testing node position: `[position]`
- testing existence of a child: `[child_tag]`
- testing value of a child: `[child_tag="value"]`
- testing existence of an attribute: `[@attribute]`
- testing value of an attribute: `[@attribute="value"]`

XPath Examples

Example

- the title of the first movie:
  `/movies/movie[1]/title`
- all movies in the year 1997:
  `movie[year="1997"]`
- black-and-white movies:
  `movie[@color="BW"]`

Example: Oracle Berkeley DBXML

- an embedded XML database
- stores XML documents
- manipulates data using XQuery
- can be used via its own client
- has bindings for several languages

DBXML Interface

- creating a database:
  - create an `XmlManager` object
  - `XmlManager.createContainer(name)` → `XmlContainer`
  - put a document element:
    `XmlContainer.putDocument(namespace, xml_string, configuration)`
- connecting to an existing database:
  - create an `XmlManager` object
  - if `XmlManager.existsContainer(name) != 0`
  - `XmlManager.openContainer(name)` → `XmlContainer`

DBXML Interface

- `XmlManager.createQueryContext()` → `XmlQueryContext`
- `XmlQueryContext.setNamespace(namespace, URL)`
- query string: `collection(name)/xpath_expression`
- running the query:
  `XmlManager.query(query, context)` → `XmlResults`
- each element of the `XmlResults` iterator is an `XmlValue`
- `getFirstChild()`, `getLastChild()`, `getNextSibling()`, ...
- character data: `getNodeValue()` → `String`
- attributes: `XmlValue.getAttributes()` → `XmlResults`

DBXML Examples

Example (database connection)

```
db = new XmlManager();
XmlContainer container = null;
if (db.existsContainer("imdb.dbxml") != 0)
  container = db.openContainer("imdb.dbxml");
else {
  container = db.createContainer("imdb.dbxml");
  container.putDocument("movies",
      "<movies />",
      (XmlDocumentConfig) null);
}
```


DBXML Examples

Example (converting a movie object into an XML string)

```
public static String toXml(Movie movie) {
  StringBuffer buffer = new StringBuffer();
  buffer.append("<movie>");
  buffer.append("<title>"
    + movie.getTitle() + "</title>");
  buffer.append("<year>"
    + movie.getYear().toString() + "</year>");
  buffer.append("</movie>");
  return buffer.toString();
}
```


DBXML Examples

Example (converting an XML node into a movie object)

```
private static Movie fromNode(XmlValue node)
        throws XmlException {
  XmlValue tn = node.getFirstChild();
  String title =
      tn.getFirstChild().getNodeValue();
  XmlValue yn = tn.getNextSibling();
  String yearValue =
      yn.getFirstChild().getNodeValue();
  Integer year = Integer.parseInt(yearValue);
  Movie movie = new Movie(title);
  movie.setYear(year);
  return movie;
}
```


DBXML Examples

Example (query: all movies)

```
XmlQueryContext context = ...;
context.setNamespace(...);
String query =
  "collection(\"imdb.dbxml\")/movies/movie";
XmlResults results = db.query(query, context);
if (results.hasNext()) {
  XmlValue node = results.next();
  Movie movie = fromNode(node);
  ...
}
results.delete();
```


DBXML Examples

Example (insert)

```
Movie m = new Movie("Casablanca");
m.setYear(1942);

XmlQueryContext context = ...;
context.setNamespace(...);
String query = "insert nodes " + toXml(m)
  + " into collection(\"imdb.dbxml\")/movies";
XmlResults results = db.query(query, context);
results.delete();
```


References

Required Reading: Date

- Chapter 27: The World Wide Web and XML