CSDL hướng đối tượng db4o (Versant)

Phạm Thị Ngọc Diễm
Bộ môn HTTT – Khoa CNTT & TT

10-2018

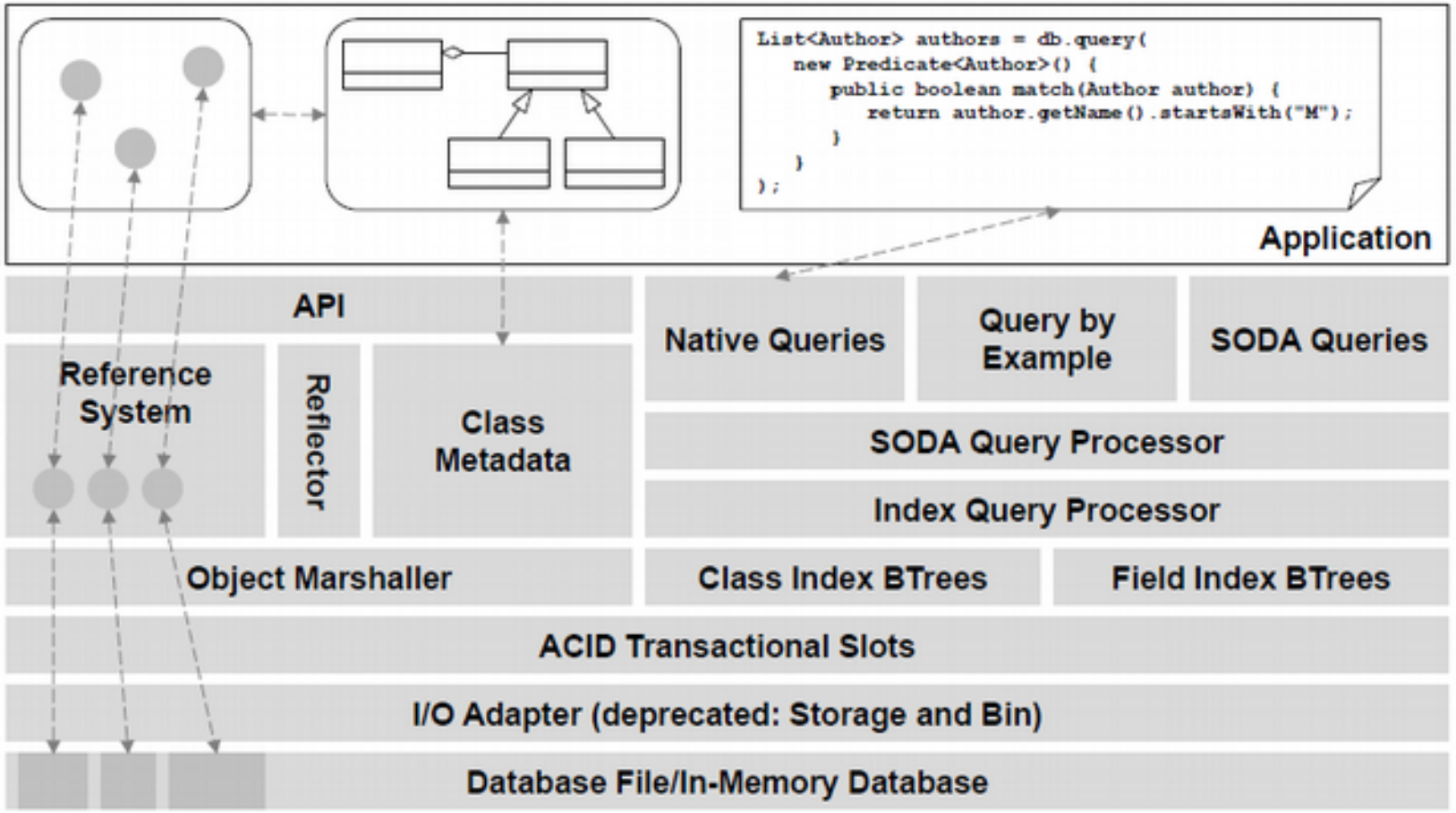
Plan

- Giới thiệu
- ObjectContainer
- Thao tác trên CSDL
- Thay đổi cấu trúc CSDL
- Object Manager Enterprise
- Sự phân cấp đối tượng
- Kích hoạt
- Index
- Giao dịch

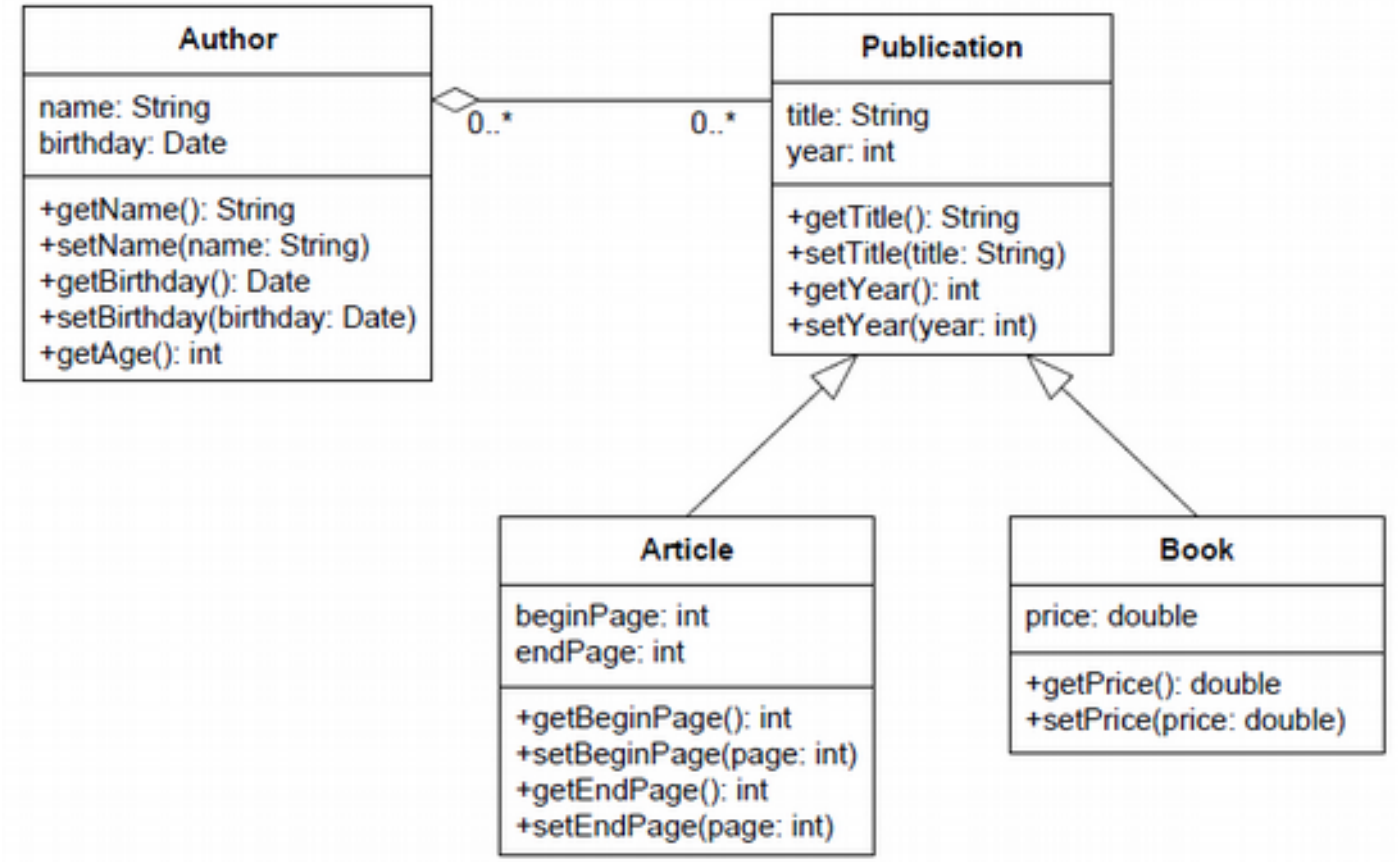
Giới thiệu db4o

- CSDL đối tượng nguồn mở cho Java và .NET
- Lưu trữ trực tiếp các đối tượng Java hoặc .NET
- Kích thước nhỏ, thích hợp cho các ứng dụng Mobile
- Được sử dụng trong các ứng dụng nhờ vào các API
- Sử dụng API bằng cách thêm file JAR hoặc DLL
- Không cần chuyển đổi hoặc mapping
- Không cần thay đổi các class để tạo các đối tượng persistent
- Một dòng code để lưu trữ các đối tượng phức tạp
- Không có quản trị CSDL trong db4o.
- Các ứng dụng phải cài đặt các ràng buộc toàn vẹn dữ liệu
- **Không hỗ trợ chuẩn ODMG**
- Hỗ trợ chế độ làm việc cục bộ hoặc mô hình client/server
- Hỗ trợ mô hình giao dịch ACID

Kiến trúc db4o



Ví dụ - Sơ đồ lớp (1)



Ví dụ - Lớp Java

```
public class Author {

  private String name;
  private Date birthday;
  private Set<Publication> pubs;

  public Author() {}

  public Author(String name) {
    this.name = name;
    this.pubs = new
          HashSet<Publication>();
  }

  public String getName() {
    return name;
  }
 …
}
```

```
public class Publication {

  private String title;
  private int year;
  private List<Author> authors;

  public Publication(String title) {
    this.title= title;
    this.authors = new
        ArrayList<Author>();
  }
  public String getTitle() {
    return title;
  }
  public void setTitle(String title){
    this.title = title;
  }
 …
}
```

ObjectContainer

- Đại diện cho các CSDL db4o :
  - Hỗ trợ chế độ file cục bộ hoặc nối kết client đến server db4o.
- Sở hữu một giao dịch:
  - Tất cả các thao tác được thực hiện là giao dịch
  - Giao dịch được bắt đầu khi đối tượng Container được mở.
  - Sau khi commit hoặc rollback, giao dịch kế tiếp được bắt đầu tự động.
- Quản lý các liên kết giữa các đối tượng được lưu trữ và được thể hiện
  - Quản lý định danh đối tượng
  - Cập nhật các đối tượng
- Vòng đời
  - Được mở suốt thời gian mà một chương trình làm việc với nó
  - Tất cả các tham chiếu đến các đối tượng trong bộ nhớ RAM sẽ bị hủy khi ObjectContainer được đóng.

Thao tác trên CSDL

- Mở các thể hiện CSDL db4o : **Db4oEmbedded**

```
// new File("D:/db4o/db4oExample.db4o").delete();

ObjectContainer db =
        Db4oEmbedded.openFile("D:/db4o/db4oExample.db4o");
```

Thao tác trên CSDL

- Lưu đối tượng:
  - Sử dụng phương thức **store** của **ObjectContainer**

```
Publication article = new Publication("Concepts for Content
Management");

Author michael = new Author("Michael Grossniklaus");
Author moira = new Author("Moira C. Norrie");

article.addAuthor(michael);
article.addAuthor(moira);

db.store(article);
```

Thao tác trên CSDL

- Tìm kiếm đối tượng: Db4o hỗ trợ ba kiểu truy vấn.
  - Query by Example (QBE):
    - Đơn giản dựa trên đối tượng template
    - Chọn chính xác
  - Native Query
    - Viết bằng ngôn ngữ lập trình Java hoặc C#
    - Được chuyển sang SODA và được tối ưu hóa
  - Simple Object Data Access (SODA)
    - Truy vấn dựa trên khái niệm đồ thị

Query by Example (QBE)

- Tạo đối tượng template cho db4o sử dụng như một *mẫu*
- db4o sẽ tìm tất cả đối tượng trong CSDL cùng kiểu như *mẫu*.
- Trả về các đối tượng có giá trị các trường của nó trùng khớp với các giá trị của các trường đã cho trong mẫu, trừ các trường có giá trị mặt định.
- Kết quả trả về là một thể hiện của **ObjectSet**.

Query by Example (QBE)

- Ví dụ:
  - Tìm các tác giả tên "**Moira C. Norrie**"

```
Author proto = new Author("Moira C. Norrie");
ObjectSet<Author> authors = db.queryByExample(proto);
for (Author author: authors) {
System.out.println(author.getName()); }
```

  - Tìm **tất cả** các publication

```
Publication pb=new Publication(null);
ObjectSet<Publication> pubs = db.queryByExample(pb);
for (Publication publication: pubs) {
        System.out.println(publication.getTitle());}
```

```
ObjectSet<Publication> pubs = db.queryByExample(Publication.class);
for (Publication publication: pubs) {
        System.out.println(publication.getTitle());}
```

Query by Example (QBE)

- **Hạn chế:**
  - Db4o phải duyệt qua tất cả các thành phần của đối tượng template.
  - Không thực hiện được các truy vấn nâng cao và phức tạp với các phép toán AND, OR, NOT, …
  - Không thể tìm kiếm với điều kiện như giá trị 0, chuỗi rỗng "" hay tham chiếu null.
  - Cần có hàm tạo với các thuộc tính không được khởi tạo.

  => Native Query

Native Query

- Được viết trong ngôn ngữ lập trình Java/C#
- Cung cấp khả năng chạy một hoặc nhiều dòng code trên tất cả các thể hiện của một lớp.
- Biểu thức truy vấn **nên trả về true** để đánh dấu những thể hiện cụ thể như một phần của tập kết quả.
- Db4o "thử" tối ưu hóa (nếu có thể) biểu thức truy vấn native và thực thi chúng trên index

  => tối ưu hóa đang trong quá trình phát triển
- Truy vấn Native là truy vấn chính và nên sử dụng để truy vấn CSDL từ ứng dụng.

Native Query

- Ví dụ

```
ObjectSet<Publication> naPubs = db.query(new Predicate<Publication>()
{
          public boolean match(Publication publication) {
               return publication.getYear() >= 1990 &&
            publication.getYear()<2000 ||
            publication.getTitle().equals("Db4o");
      }
     } );

for (Publication publication: naPubs) {
     System.out.println(publication.getTitle());
}
```

Native Query

- Ví dụ : sắp xếp

```
Comparator<Publication> personCmp=
      new Comparator<Publication>() {
      public int compare(Publication o1, Publication o2) {
          return o1.getTitle().compareTo(o2.getTitle());
                }
            };
ObjectSet<Publication> naPubs = db.query( new
Predicate<Publication>() {
    public boolean match(Publication publication) {
    return publication.getYear() >= 1990; } },personCmp );
for (Publication publication: naPubs) {
                  publication.printPub();
    }
```

Native Query

- Lớp **Predicate<ExtentType>**
- Lớp cơ bản để tạo các truy vấn Native
- Một lớp mở rộng (**<ExtentType>**) Predicate phải cài đặt một phương thức với các qui ước sau:
  - Tên phương thức phải là match() (Java) /Match() (C#)
  - Phải được khai báo public
  - Kiểu trả về phải là boolean
  - Có 1 tham số, kiểu của nó chính là lớp mở rộng **<ExtentType>**
  - Phương thức trả về **true** cho tất cả các thể hiện của lớp mở rộng bao gồm trong kết quả của truy vấn. Cho tất cả các thể hiện không bao gồm trong kết quả, match() trả về false.

Native Query

- **Hạn chế**:

  db4o sẽ phân tích và chuyển đổi truy vấn native sang truy vấn SODA

  ⇒ Không thể thực hiện với một vài truy vấn

  ⇒ Khởi tạo các đối tượng persistent để thật sự thực thi các câu truy vấn native.

  ⇒ Phân tích các câu truy vấn native để việc khởi tạo là ít nhất.

Native Query

- **Tối ưu hóa truy vấn Native** : công cụ này hỗ trợ các cấu trúc sau:
  - Các hằng về thời gian biên dịch
  - Truy xuất thành viên đơn giản
  - Các phép so sánh trên các kiểu cơ bản
  - equals() trên các kiểu cơ bản và String
  - contains(), startsWith(), endsWith() trên String
  - Biểu thức số học
  - Biểu thức boolean
  - Truy xuất trường tĩnh
  - Truy xuất mảng
  - Lời gọi phương thức tĩnh
  - Phương thức/một kết hợp tạo thành từ các cấu trúc trên

- Không hỗ trợ đa hình và phương thức nhiều dòng.

SODA

- Truy vấn SODA là truy vấn cấp thấp, cho phép truy cập trực tiếp đến các nút của đồ thị truy vấn.
- SODA sử dụng String để xác định các field

⇒ không an toàn kiểu cũng như kiểm tra thời gian biên dịch và viết khá dài dòng.

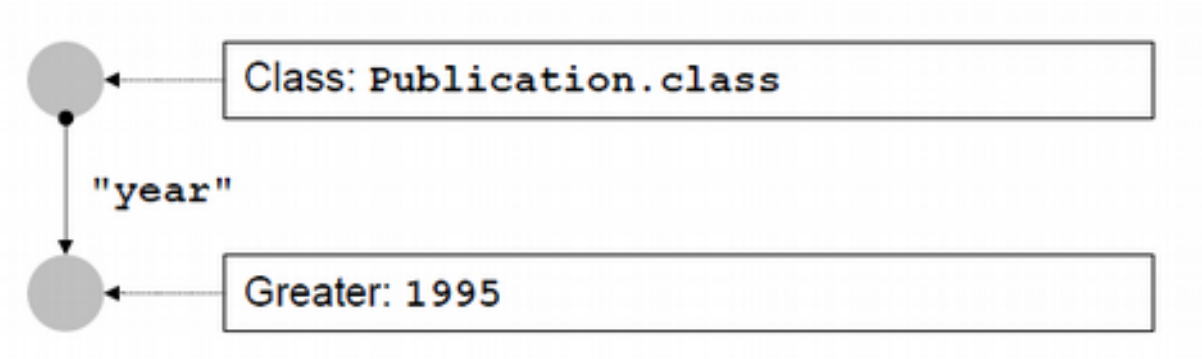
- Đối với phần lớn ứng dụng=> dùng Native query.
- Cho những ứng dụng cần sinh tự động truy vấn => dùng truy vấn SODA

SODA

- Được viết sử dụng đối tượng Query và interface Constraint.
- **Query**
  - Thêm hoặc duyệt qua một nút trong cây truy vấn với *descend()*
  - Thêm một ràng buộc (điều kiện) tới một nút dùng *constrain()*
  - Có thể sắp xếp kết quả *SortBy()* tăng dần *orderAscending()* hoặc giảm dần *orderDescending()*
  - Thực thi câu truy vấn *execute()*
- **Interface Constraint**
  - So sánh lớn hơn greater() và nhỏ hơn smaller()
  - Biểu thức bằng *equal()* và *like()*
  - Các phép toán *and, or* và *not*
  - Các hàm xử lý chuỗi với *startsWith()* và *endsWith()*
  - Kiểm tra một phần tử thuộc tập hợp với hàm *contains()*

SODA

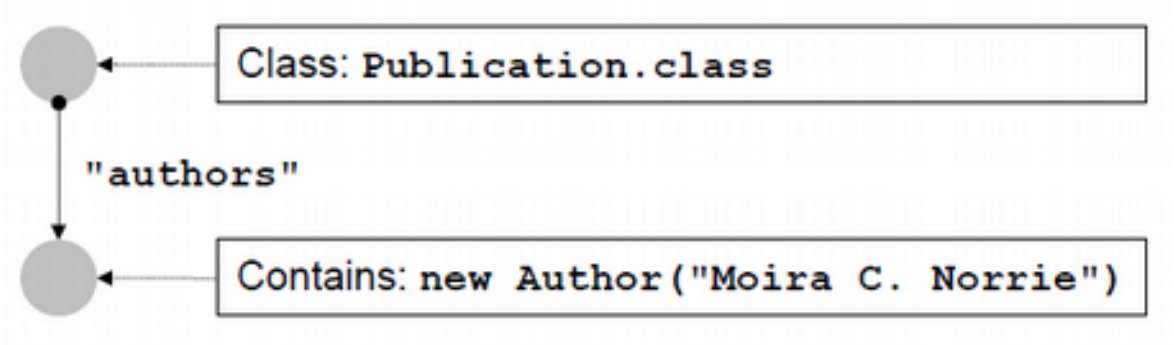
- Ví dụ




```
Query query = db.query();
query.constrain(Publication.class);
query.descend("year").constrain(Integer.valueOf(1995)).greater();
ObjectSet<Publication> publications = query.execute();
for (Publication publication : publications) {
     System.out.println(publication.getTitle()); }
```

SODA

- Ví dụ




```
Query query = db.query();
query.constrain(Publication.class);
Author proto = new Author("Moira C. Norrie");
query.descend("authors").constrain(proto).contains();
ObjectSet<Publication> publications = query.execute();
for (Publication publication : publications) {
       System.out.println(publication.getTitle()); }
```

SODA

- Ví dụ

```
Query query = db.query();
query.constrain(Publication.class);
Author proto = new Author("Moira C. Norrie");
Constraint constr=
  query.descend("year").constrain(Integer.valueOf(1995)).greater();
query.descend("authors").constrain(proto).contains().and(constr);
ObjectSet<Publication> publications = query.execute();
for (Publication publication : publications) {
       System.out.println(publication.getTitle()); }
```

Thao tác trên CSDL - Cập nhật đối tượng

- Thủ tục cập nhật:
  - Tìm đối tượng từ CSDL
  - Thực hiện các sửa đổi cần thiết
  - Lưu trữ lại đối tượng vào CSDL bằng phương thức *store()*
- Phương thức thực hiện:
  - db4o sử dụng các ID để giữ các nối kết giữa các đối tượng trong bộ nhớ và các đối tượng được lưu tương ứng trong CSDL
  - Các ID được lưu tạm như tham chiếu cho đến khi CSDL đóng lại
  - Các tham chiếu *fresh* cần để cập nhật đối tượng:
    - Truy vấn đối tượng thiết lập các tham chiếu fresh
    - Tạo và lưu trữ đối tượng sinh ra các tham chiếu fresh
  - db4o sử dụng tham chiếu để tìm và cập nhật tự động các đối tượng đã được lưu trữ khi phương thức *store()* được goi.

Cập nhật đối tượng

- Ví dụ:

```
Author au = (Author) db.queryByExample(new
         Author("Michael")).next();

Calendar calendar = Calendar.getInstance();
calendar.set(1976, Calendar.JUNE, 22);
au.setBirthday(calendar.getTime());

db.store(au);
```

- Cập nhật đối tượng có cấu trúc ???

Thao tác trên CSDL – Xóa đối tượng

- Tương tự cập nhật đối tượng:
  - Cần các tham chiếu *fresh*
  - Được thiết lập bởi truy vấn đối tượng hoặc bởi tạo và lưu trữ đối tượng
- Phương thức *delete()* của ObjectContainer xóa đối tượng

```
Author au = (Author) db.queryByExample(new
         Author("Michael")).next();

db.delete(au);
// Xóa tất cả dữ liệu
ObjectSet result=db.queryByExample(new Object());
while(result.hasNext()) {
    db.delete(result.next());}
```

- Xóa đối tượng được tham chiếu ???

Đối tượng có cấu trúc

- Lưu trữ các đối tượng mới sử dụng phương thức store()
  =>CSDL cũng sẽ *lưu trữ các đối tượng có thể được duyệt từ đối tượng này trên đồ thị đối tượng trong bộ nhớ*.
- **Cập nhật các đối tượng** đã có bằng cách sử dụng phương thức store()
  - Mặc định, độ sâu cập nhật là một
  - Chỉ giá trị các kiểu nguyên thủy và chuỗi được cập nhật
  - Đồ thị đối tượng không được duyệt vì lý do tốc độ
- **Xóa các đối tượng** đã có sử dụng phương thức delete()
  - Mặc định, các thao tác xóa không cascade
  - Đối tượng được tham chiếu phải được xóa bằng tay
  - Xóa cascade có thể được cấu hình

Cập nhật đối tượng có cấu trúc

```
Author moira = db.queryByExample(
                new Author("Moira C. Norrie")).next();

// cập nhật tất cả các publications
 for (Publication publication: moira.getPublications()) {
                 publication.setYear(2007); }

// Lưu lại tác giả không có bất kỳ thay đổi nào đến các publications
db.store(moira);

// Lưu lại các publication đã cập nhật
for (Publication publication: moira.getPublications()) {
db.store(publication); }
```

Cập nhật đối tượng có cấu trúc

- Cấu hình cập nhật cascade sử dụng phương thức cascadeOnUpdate() của ObjectClass
- Độ sâu cập nhật được cấu hình :
  - Phương thức *store(object, depth)* của *ExtObjectContainer* cập nhật các trường được tham chiếu đến độ sâu đã cho.
  - Phương thức *updateDepth(depth)* của *ObjectClass* định nghĩa một độ sâu cập nhật cho một lớp các đối tượng.
  - Phương thức *updateDepth(depth)* của *Configuration* chỉ ra độ sâu cập nhật toàn cục cho tất cả các đối tượng

Cập nhật đối tượng có cấu trúc

```
EmbeddedConfiguration config = Db4oEmbedded.newConfiguration();
config.common().objectClass(Author.class).cascadeOnUpdate(true);
ObjectContainer db = Db4oEmbedded.openFile(config, "
"D:/db4o/db4oExample.db4o");

Author moira = db.queryByExample(
                new Author("Moira C. Norrie")).next();

for (Publication publications: moira.getPublications()) {
                 publication.setYear(2007); }

db.store(moira);
```

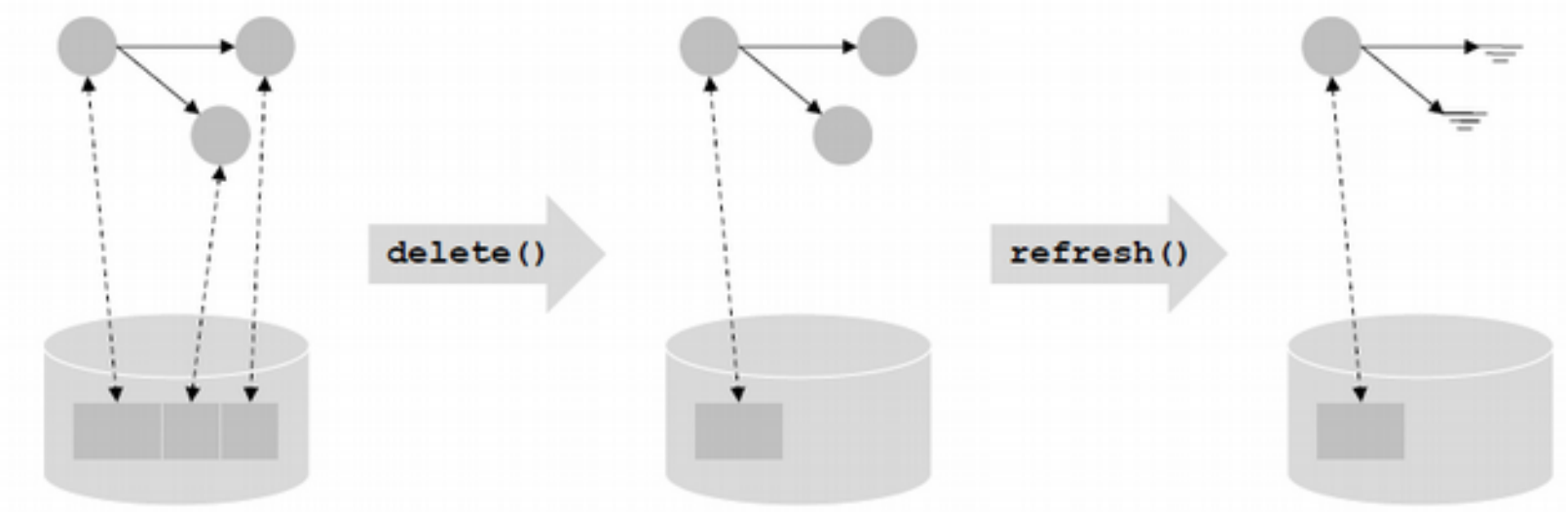
Xóa đối tượng có cấu trúc

- Xóa Cascade tương tự cập nhật cascade:
  - Cấu hình cho mỗi lớp đối tượng
  - Phương thức objectClass() của CommonConfiguration
  - Phương thức cascadeOnDelete() của ObjectClass

```
EmbeddedConfiguration config = Db4oEmbedded.newConfiguration();
config.common().objectClass(Author.class).cascadeOnDelete(true);
ObjectContainer db =
          Db4oEmbedded.openFile(config,"D:/db4o/db4oExample.db4o");
Author moira = db.queryByExample(
                  new Author("Moira C. Norrie")).next();
db.delete(moira);
```

Xóa đối tượng có cấu trúc

- Mâu thuẫn giữa bộ nhớ và các đối tượng được lưu trữ
  - Bộ nhớ cache và đĩa có thể trở nên không nhất quán khi xóa các đối tượng
  - Phương thức refresh() của ExtObjectContainer đồng bộ các đối tượng
  - Hoàn trả lại các đối tượng bộ nhớ để commit các giá trị trên đĩa



Thay đổi cấu trúc CSDL

- Db4o hỗ trợ các thay đổi :
  - Xóa một thuộc tính/trường của một đối tượng
  - Thêm một thuộc tính/trường cho một đối tượng
  - Thay đổi kiểu của một thuộc tính
  - Đặt lại tên cho thuộc tính
  - Đặt lại tên cho lớp

Object Manager Enterprise

- Là ứng dụng Java và là công cụ quản trị db4o.
- Cho phép hiển thị nội dung của một CSDL db4o đang mở.
- Có thể xem được các thuộc tính của CSDL đang mở hoặc các thuộc tính của một lớp CSDL được chọn.
- Cho phép tạo các câu truy vấn sử dụng chức năng kéo thả và hiển thị kết quả truy vấn.

Sự phân cấp đối tượng

- db4o xử lý tự động các đối tượng cấu trúc phức
  - Đối tượng phân cấp (vd Customer – Adresse – Zipcode)
    - Lưu đối tượng mức cao nhất
    - Các đối tượng cấp thấp hơn tự động được lưu
  - Kết hợp đảo (vd Authors – Publications)
    - Lưu đối tượng này đối tượng kết hợp tự động lưu
  - Thừa kế
    - Lưu đối tượng lớp con như lưu đối tượng thông thường
    - Truy vấn đối tượng lớp cha, hiển thị tất cả thể hiện cha+con
    - Có thể truy vấn đối tượng lớp cha là Abstract hoặc Interface
  - Đối tượng với thuộc tính Mảng
    - Lưu thông thường
    - Sử dụng truy vấn Native
    - Cập nhật thông thường (không dùng cascadeUpdate)
  - Đối tượng với thuộc tính Tập hợp (đối tượng)
    - Lưu đối tượng chính đồng thời lưu tập hợp đối tượng kết hợp
    - Truy vấn đối tượng đồng thời trả về đối tượng kết hợp trong tập hợp
    - Có thể truy vấn đối tượng trong tập hợp
    - Cập nhật và xóa như trường hợp các đối tượng có cấu trúc

Kích hoạt (Activation)

- Các đối tượng có trong bộ nhớ và sẵn dùng
  =>đối tượng được kích hoạt
- Độ sâu kích hoạt biểu thị chiều dài của chuỗi tham chiếu từ một đối tượng tới đối tượng khác
- Độ sâu activation mặc định là 5
- Activation xuất hiện trong các trường hợp sau:
  - Phương thức next() của ObjectSet được gọi để truy xuất kết quả của một query
  - Phương thức activate() của ObjectContainer
  - Truy xuất đến một phần tử của tập hợp db4o
  - Các phần tử của một tập hợp Java được kích hoạt khi tập hợp này được kích hoạt
- Hủy bỏ kích hoạt dùng deactivate()

Index

- Db4o sử dụng cấu trúc B-tree
- Tăng kích thước CSDL, tăng tốc độ tìm kiếm
- Tạo index : trước khi ObjectContainer mở
- Xóa index : không thể xóa index cho đến khi một ObjectContainer được mở

```
//Tạo
Db4o.configure().objectClass(Author.class).
                        objectField("name").indexed(true);

//Xóa
Db4o.configure().objectClass(Author.class).
                       objectField("name").indexed(false);
```

Giao dịch

- Hỗ trợ mô hình giao dịch ACID (Atomicity, Consistency, Isolation, Durability)
- Nhật ký giao dịch dữ liệu
  - không mất dữ liệu trong trường hợp lỗi hệ thống
  - phục hồi dữ liệu tự động sau khi lỗi hệ thống
- Tất cả các thao tác trong ObjectContainer là giao dịch
  - Giao dịch bắt đầu khi ObjectContainer mở
  - Giao dịch commit khi ObjectContainer đóng hoặc dùng phương thức commit() của ObjectContainer.
  - rollback() sẽ hủy bỏ các thay đổi

Giao dịch

- Commit CSDL

```
Author norrie = (Author) db.queryByExample(new
                Author("Moira C. Norrie")).next();

Author stefania = new Author("Stefania Leone");

Publication ar = new Publication("Web 2.0 Survey");
article.addAuthor(stefania);
article.addAuthor(norrie);

db.store(ar);
// committing database
db.commit();
```

Giao dịch

- Rollback CSDL
  - Sửa đổi được ghi vào vùng nhớ tạm
  - Commit tiềm ẩn hay tường minh sẽ ghi lại những sửa đổi vào đĩa.
  - rollback CSDL đặt lại trạng thái cơ sở dữ liệu ở lần commit gần nhất.

```
Publication article = db.queryByExample(
        new Publication("Web 2.0 Survey")).next();
//cập nhật publication
Author michael = new Author("Michael Grossniklaus");
article.addAuthor(michael);
db.store(article);
// hủy bỏ giao dịch
db.rollback();
```

Giao dịch

- Rollback CSDL
  - Không đồng bộ giữa bộ nhớ và đĩa
  - Phương thức rollback hủy bỏ những thay đổi trên đĩa
  - Trạng thái của các đối tượng trong bộ nhớ không được cập nhật
  - Các đối tượng đang hoạt động cần phải được refresh một cách tường minh

```
Publication article = db.queryByExample(
        new Publication("Web 2.0 Survey")).next();
//cập nhật publication
Author michael = new Author("Michael Grossniklaus");
article.addAuthor(michael);
db.store(article);
// hủy bỏ giao dịch
db.rollback();
//refresh article để xóa author khỏi vùng nhớ
db.ext().refresh(article, Integer.MAX_VALUE);
```

Giao dịch cạnh tranh

- Db4o sử dụng mức cô lập (Isolation) *read committed* :
  - Chỉ các giá trị đã được cập nhật và được commit bởi giao dịch khác có thể được đọc.
- Sự không nhất quán và đụng độ có thể xảy ra
- Phát hiện đụng độ:
  - Kiểm tra nếu đối tượng được thay đổi trong suốt giao dịch trước khi commit giao dịch.
  - Lưu trữ giá trị của đối tượng vào biến cục bộ khi giao dịch bắt đầu
  - Sử dụng phương thức peekPersisted() của ExtObjectContainer để tìm các thay đổi của đối tượng.
  - So sánh giá trị ban đầu với giá trị lưu hiên hành
  - rollback giao dịch hiện thời nếu giá trị thay đổi
- Phương thức peekPersisted() trả về một đối tượng độc lập không liên quan CSDL
  - Có thể chỉ ra độ sâu độ như tham số của phương thức
  - Có thể được sử dụng để đọc một trong hai giá trị commit hoặc được lưu trữ

Kiểm tra đụng độ

```
ObjectContainer db = Db4oClientServer.openClient("localhost", 3927,
"user", "pass");
Author moira = db.queryByExample(new
                    Author("Moira C. Norrie")).next();
Date birthday = moira.getBirthday();
...
Author persisted = db.ext().peekPersisted(moira, 9, true);
if (persisted.getBirthday() != birthday) {
   db.rollback();
} else {
    db.commit();
}
```