Bases de Dades embegudes

Desfase objecte-relacional

Consorci d'Educació
de Barcelona
Generalitat de Catalunya
Ajuntament de Barcelona

El desfase Objecte - Relacional

- Com sabem el paradigma de programació POO està més que consolidat i els elements que utilitza són els objectes.
- Les bases de dades relacionals no estan dissenyades per a emmagatzemar aquests objectes per defecte. A les BBDDR desem dades primitives, relacionades en taules.
- La diferència d'esquemes entre els elements a emmagatzemar (objectes) i les característiques del repositori de dades (taules) provoca un augment en la complexitat dels programes i en els costos de desenvolupament.
- Als problemes que ocorren a causa de les diferències entre el model de dades de la base de dades i el llenguatge de programació orientat a objectes se'l denomina ***desfase objecte-relacional*** (o desajustament de la impedància).

Bases de dades embegudes

- Quan desenvolupem petites aplicacions en les quals no emmagatzemarem grans quantitats de dades no és necessari que utilitzem un sistema gestor de base de dades com Oracle o MySql.
- En el seu lloc podem utilitzar una base de dades embeguda on el motor està incrustat en l'aplicació i sigui exclusiu per a ella.
- La base de dades s'inicia quan s'executa l'aplicació, i acaba quan es tanca l'aplicació.
- Vegem alguns exemples:

SQLite

- SGBDR multiplataforma escrit en C que proporciona un motor molt lleuger.
- Les bbdd es guarden en forma de fitxers per la qual cosa és fàcil traslladar la bbdd amb l'aplicació que la usa.
- Compta amb una petita utilitat que ens permetrà executar comandes SQL en consola.
- És un projecte de domini públic i implementa la major part de SQL-92, incloent transaccions, consistència (foreign keys), triggers, consultes complexes, etc.
- SQLite es pot utilitzar des de C/C++, PHP, Visual Basic, Perl, Java, etc.

Apache Derby

- És un SGBDR de codi obert implementat completament a Java i forma part de l'Apache DB Project.
  - Avantatges: grandària reduïda, basada en Java, suporta estàndards SQL, ofereix un controlador integrat JDBC i pot incrustar-se fàcilment en qualsevol solució Java.
  - Suporta també el tradicional paradigma client-servidor utilitzant el servidor de xarxa Derby.

Activitat: Apache Derby

- Crear una nova màquina virtual amb Linux (per no fer malbé la màquina local vostra).
- Instal·lar la màquina virtual de java, per exemple des de terminal, compte amb la versió que useu per després usar una o altra versió d'Apache Derby:

1. sudo apt update
2. sudo apt install oracle-java[NUM]-installer
3. sudo apt install oracle-java[NUM]-set-default

Activitat: Apache Derby

4. wget https://dlcdn.apache.org//db/derby/db-derby-10.15.2.0/db-derby-10.15.2.0-bin.tar.gz
5. tar xzf db-derby-10.15.2.0-bin.tar.gz
6. mv db-derby-10.15.2.0 /opt
7. cd /opt
8. export DERBY_HOME=/opt/db-derby-10.15.2.0-bin
9. export CLASSPATH=$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar
10. cd $DERBY_HOME/bin
11. ./setEmbeddedCP
12. java org.apache.derby.tools.ij

- **ij** es una utilidad que nos permite crear nuestra bbdd desde consola

Activitat: Apache Derby

- Per crear una base de dades de nom “exemple” en la carpeta /DB/DERBY escriurem:

ij>CONNECT ‘jdbc:derby:/home/usuari/DB/DERBY/exemple;create=true’;

- On:
  - connect → comanda per establir connexió.
  - jdbc:derby → protocol JDBC especificat per DERBY.
  - exemple → és el nom de la bbdd que crearem (es crea una carpeta amb aquest nom i a dintre una sèrie de fitxers).
  - create=true → atribut utilitzat per crear la base de dades.
- Per sortir de **ij** escribim **exit;**
- Per obtenir ajuda podem escriure **help;**

Activitat: Apache Derby

- **A1.-** Ara comença realment el problema. Has de crear una bbdd en DERBY que contingui dues taules i que inclogui una clau foranea entre ambdues. Per exemple:

DEPARTAMENTS (dept_no, dnom, loc)
EMPLEATS (emp_no, cognom, ofici,data_alt,salari,dept_no)
ON {dept_no} REFERENCIA DEPARTAMENTS

- Intenteu a més a més realitzar alguna inserció i consulta.

HSQLDB

- Hyperthreaded Structured Query Language Database és un SGBDR escrit a Java.
- Gestiona fins a 270 mil milions de files de dades en una sola bbdd i té capacitat de còpia en calent.
- Implementa SQL:2011.
- OpenOffice ho inclou per a donar suport a l'aplicació Base.

H2

- H2 és un SGBDR escrit a Java. Llicència pública de Mozilla.
- Té diverses maneres d'estar embeguda: com a servidor o completament en. memòria

H2

- H2 és un SGBDR escrit a Java. Llicència pública de Mozilla.
- Té diverses maneres d'estar embeguda: com a servidor o completament en. memòria

DB4o

- Motor de base de dades orientat a objectes.
- Llicència dual: GPL/comercial: no hi ha costos per a desenvolupar programari de codi lliure, però si es desitja desenvolupar programari privatiu sí.
- No utilitza SQL. Estructures totalment diferents.
- S'instal·la afegint un únic fitxer de llibreria (JAR per a Java o DLL per a .NET).
- És un únic fitxer de base de dades amb l'extensió .YAP (grandària de 2GB a 264GB).

Exemple d'emmagatzemar

```
public class Persona { //Primer creem la classe que utilitzarem per crear els objectes a emmagatzemar:
    private String nom;
    private String ciutat;
    public Persona(String nom, String ciutat){
        this.nom = nom;
        this.ciutat = ciutat;
    }
    public String getNom(){
        return nom;
    }
    public String getCiutat(){
       return ciutat;
    }
    public void setNom (String nom){
        this.nom=nom;
    }
    public void setCiutat (String ciutat){
        this.ciutat=ciutat;
    }
}
```

Exemple d'emmagatzemar

```
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
public class Main{
   static String BDPer = "DBPersones.yap";
   public static void main (String [] args) {
        ObjectContainer db = Db4oEmbedded.openFile(Db4oEmbedded.newConfiguration(), BDPer);
        //Amb el mètode openFile() creem la base de dades "DBPersones.yap"
        //A continuació creem els objectes:
        Persona p1 = new Persona ("Juan","Guadalajara");
        Persona p2 = new Persona ("Anna","Madrid");
        Persona p3 = new Persona ("Lluis","Granada");
        Persona p4 = new Persona ("Gina","Asturias");
        //Utilitzem el mètode store() per emmagatzemar els objectes
        db.store(p1);
        db.store(p2);
        db.store(p3);
        db.store(p4);
        //Tanquem la base de dades
        db.close();
   }
}
```

Exemple consulta

```
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.db4o.ObjectSet;
public class Consulta1 {
   static String BDPer = "DBPersones.yap";
   public static void main (String[] args) {
       //Obrim la base de dades
       ObjectContainer db = Db4oEmbedded.openFile (Db4oEmbedded.newConfiguration(), BDPer);
       Persona per = new Persona (null,null); //Els paràmetres fan referència al Nom i la Ciutat
       //Per recuperar objectes utilitzem el sistema de consultas QBE (Query-By-Example).
       ObjectSet<Persona> result = db.queryByExample(per);
       if (result.size() == 0)
           System.out.println("No existeixen Registres de persones");
       else
           System.out.printf ("Número de registres : %d %n", result.size());
       while (result.hasNext()){
           Persona p = result.next();
           System.out.printf("Nom: %s, Ciutat: %s %n", p.getNom(),p.getCiutat());
       }
   }
}
```

Exemple consulta

- La següent consulta obté els objectes Persona el nom de la qual és Juan:

```
Persona per = new Persona ("Juan",null);
ObjectSet<Persona> result = db.queryByExample(per);
```

- La següent consulta obté els objectes Persona la ciutat de Guadalajara:

```
Persona per = new Persona (null,"Guadalajara");
ObjectSet<Persona> result = db.queryByExample(per);
```

Activitat

- Descarregar alguna versió de Db4o: Versió 8, Versió 7, o qualsevol altra que trobeu.
- Descomprimir el fitxer .zip descarregat.
- En la carpeta **/lib** es troben els JAR necessaris. Hem d'incloure en la variable **CLASSPATH** la ruta a aquests fitxers JAR:

```
export CLASSPATH=/home/db4o-8.0/lib/db4o-8.0.276.16149-all-java5.jar:.:
```

Activitat

- **A1.-** Crear un programa Java per modificar un objecte de la base de dades "DBPersones.yap". Per tal propòsit, primer cal localitzar l'objecte, després modificar-lo i, finalment, tornar a emmagatzemar-ho amb store(). En concret, modifica la ciutat de la persona que l'usuari del programa ens escrigui, i després ho visualitzarem.
- **A2.-** Crea un programa Java per eliminar tots els objectes el nom dels quals sigui Juan. Per a eliminar objectes utilitzem el mètode delete().
- **A3.-** Crea una nova base de dades de nom ***EMPLEDEP.yap*** i inserir objectes EMPLEATS i DEPARTAMENTS en ella. Després obtenir tots els objectes empleat d'un departament concret. Visualitza el nom d'aquest departament i el dels empleats. No cal crear una *foreing key*.