Universidad Nacional de la Patagonia San Juan Bosco - Facultad de Ingeniería - Sede Trelew.

Licenciatura en Sistemas Orientación PG y CPI.

BASE DE DATOS II.

- Año 2013 –

TRABAJO PRACTICO FINAL

BASES DE DATOS ORIENTADAS A OBJETOS

Alumnos:

Morales Leonardo.

Urrutia Guillermo.

**Enunciado:**

**1)** Implementar las siguientes clases en Java:

- Juez: propiedades: Nombre, matricula profesional, trayectoria (texto).

- Juzgado: propiedades: Fuero (civil, comercial, laboral, penal), Juez (instancia), domicilio, localidad.

- Persona: nombre, apellido, dni, sexo, fecha nacimiento.

- Causa: propiedades: Juzgado, Nro. Expediente, imputados (conjunto de Persona), testigos (conjunto de Persona), sentencia (texto, nulo si no tiene todavía).

**2)** Crear una aplicación Java que genere múltiples instancias de cada una, y utilizando el entorno eclipse, les dé persistencia en el motor de bases de datos Orientado a objetos: DB4O.

**3)** Hacer las siguientes consultas con QueryByExample:

- Mostrar las causas con sentencia que tengan más de 2 imputados, se deberá mostrar los datos de la causa y los datos de sus imputados.

- Mostrar los juzgados del fuero civil que tengan al menos una causa con sentencia y una causa sin sentencia.

**4)** Resolver también las dos consultas utilizando Native Queries y SODA (una con cada método).

**Pautas:**

- Se podrá conformar un grupo de hasta 2 alumnos para realizar el trabajo.

- Se deberá entregar un informe escrito y en formato PDF con el trabajo y la presentación.

- Se deberá mostrar el trabajo en máquina a la cátedra.

- Fecha final de presentación: Lunes 1 de julio.

1) Implementar las siguientes clases en Java:

- Juez: propiedades: Nombre, matricula profesional, trayectoria (texto).

**public** **class** Juez {

**private** String nombre;

**private** Integer matriculaProfesional;

**private** String trayectoria;

//Constructor:

**public** Juez(String nombre, Integer matriculaProfesional, String trayectoria) {

// Crea un nuevo juez.

**this**.nombre = nombre;

**this**.matriculaProfesional = matriculaProfesional;

**this**.trayectoria = trayectoria;

}

//Getters and Setters:

**public** String getNombre() {

**return** nombre;

}

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

}

**public** Integer getMatriculaProfesional() {

**return** matriculaProfesional;

}

**public** **void** setMatriculaProfesional(Integer matriculaProfesional) {

**this**.matriculaProfesional = matriculaProfesional;

}

**public** String getTrayectoria() {

**return** trayectoria;

}

**public** **void** setTrayectoria(String trayectoria) {

**this**.trayectoria = trayectoria;

}

@Override

**public** String toString() {

**return** "Juez{ nombre: "+**this**.getNombre()+"\t| matricula: "+**this**.getMatriculaProfesional()+ " }";

}

}

- Juzgado: propiedades: Fuero (civil, comercial, laboral, penal), Juez (instancia), domicilio, localidad.

**public** **class** Juzgado {

**enum** TipoFuero{*civil*, *comercial*, *laboral*, *penal*};

**private** TipoFuero fuero;

**private** Juez juez;

**private** String domicilio;

**private** String localidad;

**private** ArrayList<Causa> causas;

//Constructor:

**public** Juzgado(TipoFuero fuero, Juez juez, String domicilio, String localidad) {

// Crea un juzgado.

**this**.fuero = fuero;

**this**.juez = juez;

**this**.domicilio = domicilio;

**this**.localidad = localidad;

//Conjunto de causas vacias al principio:

**this**.setCausas(**new** ArrayList<Causa>());

}

//Getters and Setters:

**public** TipoFuero getFuero() {

**return** fuero;

}

**public** Juez getJuez() {

**return** juez;

}

**public** String getDomicilio() {

**return** domicilio;

}

**public** String getLocalidad() {

**return** localidad;

}

**public** ArrayList<Causa> getCausas() {

**return** causas;

}

**public** **void** setCausas(ArrayList<Causa> causas) {

**this**.causas = causas;

}

**public** **void** setFuero(TipoFuero fuero) {

**this**.fuero = fuero;

}

**public** **void** setJuez(Juez juez) {

**this**.juez = juez;

}

**public** **void** setDomicilio(String domicilio) {

**this**.domicilio = domicilio;

}

**public** **void** setLocalidad(String localidad) {

**this**.localidad = localidad;

}

@Override

**public** String toString() {

**return** "Juzgado{ fuero: "+**this**.getFuero()+"\t| juez: "+**this**.getJuez() + " }";

}

**public** **boolean** poseeCausaConSentencia() {

**boolean** encontreCausa = **false**;

Iterator<Causa> iterador = **this**.getCausas().iterator();

**while** ((iterador.hasNext()) && (! encontreCausa)) {

**if** (iterador.next().getSentencia() != **null**)

encontreCausa = **true**; //xq la encontre.

}

**return** encontreCausa;

}

**public** **boolean** poseeCausaSinSentencia() {

**boolean** encontreCausa = **false**;

Iterator<Causa> iterador = **this**.getCausas().iterator();

**while** ((iterador.hasNext()) && (! encontreCausa)) {

**if** (iterador.next().getSentencia() == **null**)

encontreCausa = **true**; //xq la encontre.

}

**return** encontreCausa;

}

}

- Persona: nombre, apellido, dni, sexo, fecha nacimiento.

**public** **class** Persona {

**private** String nombre;

**private** String apellido;

**private** Integer dni;

**private** Date fechaNacimiento;

**private** String sexo;

//Constructor:

**public** Persona(String nombre, String apellido, Integer dni, Date fechaNacimiento, String sexo) {

// Crea una nueva persona

**this**.nombre = nombre;

**this**.apellido = apellido;

**this**.dni = dni;

**this**.fechaNacimiento = fechaNacimiento;

**this**.sexo = sexo;

}

//Getters and Setters:

**public** String getNombre() {

**return** nombre;

}

**public** String getApellido() {

**return** apellido;

}

**public** Integer getDni() {

**return** dni;

}

**public** Date getFechaNacimiento() {

**return** fechaNacimiento;

}

**public** String getSexo() {

**return** sexo;

}

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

}

**public** **void** setApellido(String apellido) {

**this**.apellido = apellido;

}

**public** **void** setDni(Integer dni) {

**this**.dni = dni;

}

**public** **void** setFechaNacimiento(Date fechaNacimiento) {

**this**.fechaNacimiento = fechaNacimiento;

}

**public** **void** setSexo(String sexo) {

**this**.sexo = sexo;

}

@Override

**public** String toString() {

**return** "Persona{ nombre y apellido: "+**this**.getNombre()+" "+**this**.getApellido()+"\t| dni: "+**this**.getDni()+"\t| fecha nac.: "+**this**.getFechaNacimiento().toString() +" }";

}

}

- Causa: propiedades: Juzgado, Nro. Expediente, imputados (conjunto de Persona), testigos (conjunto de Persona), sentencia (texto, nulo si no tiene todavia).

**public** **class** Causa {

**private** Juzgado juzgado;

**private** Integer nroExpediente;

**private** ArrayList<Persona> imputados;

**private** ArrayList<Persona> testigos;

**private** String sentencia;

//Constructor:

**public** Causa(Juzgado juzgado, Integer nroExpediente, ArrayList<Persona> imputados, ArrayList<Persona> testigos, String sentencia) {

**this**.juzgado = juzgado;

**this**.nroExpediente = nroExpediente;

**this**.imputados = imputados;

**this**.testigos = testigos;

**this**.sentencia = sentencia;

}

//Getters and Setters:

**public** Juzgado getJuzgado() {

**return** juzgado;

}

**public** **void** setJuzgado(Juzgado juzgado) {

**this**.juzgado = juzgado;

}

**public** Integer getNroExpediente() {

**return** nroExpediente;

}

**public** **void** setNroExpediente(Integer nroExpediente) {

**this**.nroExpediente = nroExpediente;

}

**public** ArrayList<Persona> getImputados() {

**return** imputados;

}

**public** **void** setImputados(ArrayList<Persona> imputados) {

**this**.imputados = imputados;

}

**public** ArrayList<Persona> getTestigos() {

**return** testigos;

}

**public** **void** setTestigos(ArrayList<Persona> testigos) {

**this**.testigos = testigos;

}

**public** String getSentencia() {

**return** sentencia;

}

**public** **void** setSentencia(String sentencia) {

**this**.sentencia = sentencia;

}

@Override

**public** String toString() {

**return** "Causa { expediente: "+**this**.getNroExpediente()+" juzgado: ["+**this**.getJuzgado()+"] sentencia: "+**this**.getSentencia() + " }";

}

**public** **void** agregarASuJuzgado() {

**this**.getJuzgado().getCausas().add(**this**);

}

}

2) Crear una aplicación Java que genere múltiples instancias de cada una, y utilizando el entorno eclipse, les dé persistencia en el motor de bases de datos Orientado a objetos: DB4O.

**public** **class** CreadorDeInstancias {

**public** **static** **void** main(String[] args) {

ObjectContainer db= Db4oEmbedded.*openFile*("BDJuzgado.db4o");

**try**{

//generando instancias de jueces:

Juez juez1 = **new** Juez("Delfino German", 1000, "a");

db.store(juez1);

Juez juez2 = **new** Juez("Belligoy Federico", 2000, "b");

db.store(juez2);

Juez juez3 = **new** Juez("Abal Diego", 3000, "c");

db.store(juez3);

Juez juez4 = **new** Juez("Pezzota Mauricio", 4000, "d");

db.store(juez4);

System.*out*.println("Jueces creados y almacenados: "+"\n"+juez1 +"\n"+ juez2 + "\n"+juez3+ "\n"+juez4);

//generando instancias de personas:

Persona persoGuille = **new** Persona("Guillermo", "Urrupia", 30888777, *nuevaFechaDesdeString*("1989/5/2"), "Masculino");

db.store(persoGuille);

Persona persoPablo = **new** Persona("Pablo", "Zavarro", 37555666, *nuevaFechaDesdeString*("1992/6/2"), "Masculino");

db.store(persoPablo);

Persona persoSamuel = **new** Persona("Samuel", "Almocacid", 31888777, *nuevaFechaDesdeString*("1989/7/2"), "Masculino");

db.store(persoSamuel);

Persona persoEmma = **new** Persona("Emmanuel", "Tominguez", 30888777, *nuevaFechaDesdeString*("1987/8/2"), "Masculino");

db.store(persoEmma);

System.*out*.println("\nPersonas creadas y almacenadas: "+"\n"+persoGuille +"\n"+ persoPablo+ "\n"+persoSamuel+ "\n"+persoEmma);

//generando instancias de juzgados:

Juzgado juzgado1 = **new** Juzgado(Juzgado.TipoFuero.*civil*, juez1, "San Martin 1500", "Trelew");

Juzgado juzgado2 = **new** Juzgado(Juzgado.TipoFuero.*comercial*, juez2, "Sarmiento 1500", "Trelew");

Juzgado juzgado3 = **new** Juzgado(Juzgado.TipoFuero.*penal*, juez3, "Belgrano 1500", "Trelew");

Juzgado juzgado4 = **new** Juzgado(Juzgado.TipoFuero.*civil*, juez4, "Mitre 1500", "Rawson");

//Almacenamos los juzgados.

db.store(juzgado1);

db.store(juzgado2);

db.store(juzgado3);

db.store(juzgado4);

System.*out*.println("\nJuzgados creados y almacenados: "+"\n"+juzgado1 +"\n"+ juzgado2 + "\n"+juzgado3+ "\n"+juzgado4);

//generando instancias de causas:

//Causa1

ArrayList<Persona> imputados = **new** ArrayList<Persona>();

imputados.add(persoPablo);

imputados.add(persoEmma);

ArrayList<Persona> testigos = **new** ArrayList<Persona>();

testigos.add(persoGuille);

Causa causa1 = **new** Causa(juzgado1, 0001, imputados, testigos, "Culpables");

db.store(causa1);

//Causa2:

ArrayList<Persona> imputados1 = **new** ArrayList<Persona>();

imputados1.add(persoSamuel); //Solo un imputado!

Causa causa2 = **new** Causa(juzgado2, 0002, imputados1, testigos, "");

db.store(causa2);

//Causa3:

Causa causa3 = **new** Causa(juzgado3, 0003, imputados, testigos, "Inocentes");

db.store(causa3);

//Dos causas (4 y 5), una con sentencia y otra sin sentencia:

//Causa4

ArrayList<Persona> imputadosC4 = **new** ArrayList<Persona>();

imputadosC4.add(persoPablo);

imputadosC4.add(persoSamuel);

ArrayList<Persona> testigosC4 = **new** ArrayList<Persona>();

testigosC4.add(persoEmma);

Causa causa4 = **new** Causa(juzgado4, 0004, imputadosC4, testigosC4, "Culpables. Pablo y Samuel mataron a Guille.Pablo autor intelectual, Samuel autor material");

db.store(causa4);

//Causa5

ArrayList<Persona> imputadosC5 = **new** ArrayList<Persona>();

imputadosC5.add(persoPablo);

imputadosC5.add(persoGuille);

ArrayList<Persona> testigosC5 = **new** ArrayList<Persona>();

testigosC5.add(persoEmma);

Causa causa5 = **new** Causa(juzgado4, 0005, imputadosC5, testigosC5, **null**);

db.store(causa5);

System.*out*.println("\nCausas creadas y almacenadas: "+"\n"+causa1 +"\n"+ causa2+ "\n"+causa3+ "\n"+causa4+ "\n"+causa5);

db.commit();

}

**finally**{

db.close();

}

}

3) Hacer las siguientes consultas con QueryByExample:

- Mostrar las causas con sentencia que tengan más de 2 imputados , se deberá mostrar los datos de la causa y los datos de sus imputados.

- Mostrar los juzgados del fuero civil que tengan al menos una causa con sentencia y una causa sin sentencia.

**public** **class** Main {

**static** String *barraDivisoria* = "\n========================================";

**public** **static** **void** main(String[] args) {

ObjectContainer db= Db4oEmbedded.*openFile*("BDJuzgado.db4o");

ArrayList<Causa> causasMasDeDosImputados;

**try**{

//generarInstancias(db);

System.*out*.println(*barraDivisoria*+"\n\nCONSULTA: Mostrar las causas con sentencia que tengan mas de 2 imputados , se deberá mostrar los datos de la causa y los datos de sus imputados:\n\n"+*barraDivisoria*);

causasMasDeDosImputados = *recuperarTodasLasCausas*(db);

*mostrarCausas*(causasMasDeDosImputados);

System.*out*.println(*barraDivisoria*+"\n\nCONSULTA: Mostrar los juzgados del fuero civil que tengan al menos una causa con sentencia y una causa sin sentencia:\n\n"+*barraDivisoria*);

*consultaJuzgadosCivil*(db);

}

**finally**{

db.close();

}

}

**private** **static** **void** consultaJuzgadosCivil(ObjectContainer db) {

//prototipo: todos los Juzgados del fuero civil:

Juzgado protoJuzgado = **new** Juzgado(Juzgado.TipoFuero.*civil*, **null**, **null**, **null**);

ObjectSet<Object> resultJuzgado = db.queryByExample(protoJuzgado);

Causa protoCausa = **new** Causa(protoJuzgado, **null**, **null**, **null**, **null**);

ObjectSet<Object> resultCausas = db.queryByExample(protoCausa);

*listResult*(resultJuzgado, resultCausas);

}

**private** **static** **boolean** tieneCausasValidas(Juzgado unJuzgado) {

**return** unJuzgado.poseeCausaConSentencia() && unJuzgado.poseeCausaSinSentencia();

}

**public** **static** **void** listResult(ObjectSet<Object> resultJuzgado, ObjectSet<Object> resultCausas)

{

Causa unaCausa;

**while**(resultCausas.hasNext()) {

unaCausa = (Causa) resultCausas.next();

unaCausa.agregarASuJuzgado();

}

System.*out*.println("\nCantidad total de juzgados con fuero civil encontrados:"+resultJuzgado.size()+"\nJuzgados que cumplen la condicion:");

Juzgado unJuzgado;

**while**(resultJuzgado.hasNext()) {

unJuzgado = (Juzgado) resultJuzgado.next();

**if** (*tieneCausasValidas*(unJuzgado)) {

System.*out*.println("\n"+unJuzgado.toString());

System.*out*.println("\tCausas del juzgado: "+unJuzgado.getCausas().size());

**for** (**int** i = 0; i < unJuzgado.getCausas().size(); i++)

System.*out*.println("\t"+unJuzgado.getCausas().get(i));

}

}

}

**public** **static** ArrayList<Causa> recuperarTodasLasCausas(ObjectContainer db) {

Causa protoCausa = **new** Causa(**null**, **null**, **null**, **null**, **null**);

ObjectSet<Object> result = db.queryByExample(protoCausa);

//filtro...

ArrayList<Causa> causas = *causasMasDeDosImputados*(result);

**return**(causas);

}

**public** **static** ArrayList<Causa> causasMasDeDosImputados(ObjectSet<Object> causas) {

Causa unaCausa;

ArrayList<Causa> causasRet = **new** ArrayList<Causa>();

**while** (causas.hasNext()) {

unaCausa = (Causa) causas.next();

**if** (unaCausa.getImputados().size() > 1) {

causasRet.add(unaCausa);

}

}

**return** causasRet;

}

**public** **static** **void** mostrarCausas(ArrayList<Causa> causas){

System.*out*.println("\nCausas con mas de dos imputados:"+causas.size());

Iterator<Causa> iterador = causas.iterator();

Causa causa;

**while**(iterador.hasNext()) {

causa = iterador.next();

System.*out*.println("\n"+causa.toString());

System.*out*.println("\tDatos de los Imputados:");

**for** (**int** i = 0; i < causa.getImputados().size(); i++) {

System.*out*.println("\t"+causa.getImputados().get(i).toString());

}

}

}

/\* Mostrar los Juzgados que tengan al menos una causa con sentencia \*/

**private static void juzgadosUnaCausasConSentenciasSODA(ObjectContainer db) {**

**/\* SODA Querie example \*/**

**System.*out*.println(*barraDivisoria*+"\n\nCONSULTA BY SODA: Mostrar los Juzgados que tengan al menos una causa con sentencia\n\n"+*barraDivisoria*);**

**//prototipo: todos los Juzgados del fuero civil:**

**Query query = db.query();**

**query.constrain(Juzgado.class);**

**//Consulta por el tipo de Juzgado**

**query.descend("fuero").constrain(Juzgado.TipoFuero.*civil*);**

**//Subconsulta sobre la coleccion de causas**

**Query causasQuery = query.descend("causas");**

**causasQuery.constrain(Causa.class);**

**causasQuery.descend("sentencia").constrain(null).not();**

**ObjectSet<Juzgado> juzgados = query.execute();**

**System.*out*.println("Cantidad de Juzgados: " + juzgados.size());**

***imprimirJuzgados*(juzgados);**

**}**

/\* Mostrar los juzgados del fuero civil que tengan al menos una causa sin sentencia \*/

**private** **static** **void** juzgadosUnaCausaSinSentenciaSODA(ObjectContainer db) {

/\* SODA Querie example \*/

System.*out*.println(*barraDivisoria*+"\n\nCONSULTA BY SODA: Mostrar los juzgados del fuero civil que tengan al menos una causa sin sentencia\n\n"+*barraDivisoria*);

//prototipo: todos los Juzgados del fuero civil:

Query query = db.query();

query.constrain(Juzgado.**class**);

query.descend("fuero").constrain(Juzgado.TipoFuero.*civil*);

//Subconsulta sobre la coleccion de causas

Query causasQuery = query.descend("causas");

causasQuery.constrain(Causa.**class**);

causasQuery.descend("sentencia").constrain(**null**);

ObjectSet<Juzgado> juzgados = query.execute();

System.*out*.println("Cantidad de Juzgados: " + juzgados.size());

*imprimirJuzgados*(juzgados);

}

**/\*** NATIVE QUERIE: Mostrar los juzgados del fuero civil que tengan al menos una causa sin sentencia. **\*/**

**private static void juzgadosSinMasDeUnaCausasConSentenciasByNQ(ObjectContainer db) {**

**System.*out*.println(*barraDivisoria*+"\n\nCONSULTA BY NATIVE QUERIE: Mostrar los juzgados del fuero civil que tengan al menos una causa sin sentencia\n\n"+*barraDivisoria*);**

**List<Juzgado> juzgados = db.query(new Predicate<Juzgado>() {**

**@Override**

**public boolean match(Juzgado j) {**

**// TODO Auto-generated method stub**

**boolean tieneCausaSinSentencia = false;**

**for (Causa c : j.getCausas()) {**

**if(c.getSentencia()==null)**

**{**

**tieneCausaSinSentencia = true;**

**break;**

**}**

**}**

**return (j.getFuero() == juzgado.Juzgado.TipoFuero.*civil*) && tieneCausaSinSentencia;**

**}**

**});**

***imprimirJuzgados*(juzgados);**

**}**

/\* NATIVE QUERIE: Mostrar los juzgados del fuero civil que tengan al menos una causa con sentencia. \*/

**private** **static** **void** juzgadosConMasUnaCausasConSentenciasByNQ(ObjectContainer db) {

System.*out*.println(*barraDivisoria*+"\n\nCONSULTA BY NATIVE QUERIE: Mostrar los juzgados del fuero civil que tengan al menos una causa con sentencia\n\n"+*barraDivisoria*);

List<Juzgado> juzgados = db.query(**new** Predicate<Juzgado>() {

@Override

**public** **boolean** match(Juzgado j) {

// **TODO** Auto-generated method stub

**boolean** tieneCausaConSentencia = **false**;

**for** (Causa c : j.getCausas()) {

**if**(c.getSentencia()!=**null**)

{

tieneCausaConSentencia = **true**;

**break**;

}

}

**return** (j.getFuero() == juzgado.Juzgado.TipoFuero.*civil*) && tieneCausaConSentencia;

}

});

*imprimirJuzgados*(juzgados);

}

**private** **static** **void** imprimirJuzgados(List<Juzgado> juzgados) {

**for** (Juzgado j : juzgados) {

System.*out*.println(j.toString()+"cantidad de causas: "+j.getCausas().size());

**for** (Causa c : j.getCausas()) {

System.*out*.println("\t" + c);

}

}

}

}

4) Resolver también las dos consultas utilizando Native Queries y SODA (una con cada método).