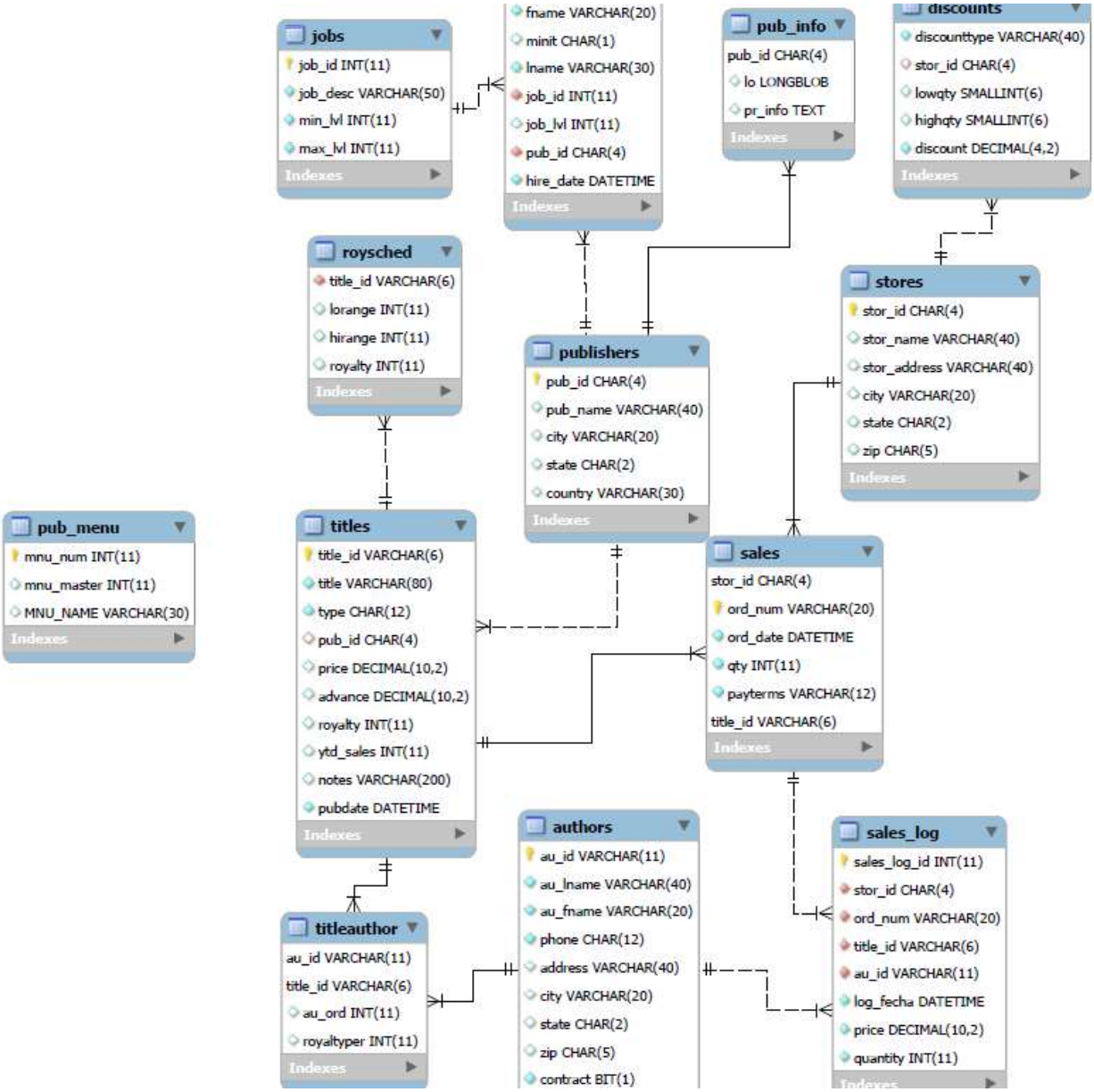
Contents

LAB008 final INF515 UASD 2023-20, Crear una aplicación que maneje múltiples tiendas y ventas de libros en Mysql, del modelo Pubs.

1) Cree las siguientes class data entity, según modelo de database (idénticas a cada tabla en mysql):
    a. Title (todos los campos)
    b. Sales
    c. Authors
    d. Employee
    e. Authortitle
    f. Job
    g. Y demas entidades (ver diagrama anexo)
    h.

2) Model entity
    a. Arraylist para gestionar el CRUD de cada objeto
    b. Consideres hashmap, de ser necesario
3) Cree un screem consumer que instancie la clase en do{ }while.
    a. Lista de tablas demograficas.
        i. Title,
        ii. Authors/titleauthor
        iii. Job
        iv. Employee
        v. Otras
    b. Consumidor transaccional.
        i. Ventas de libros por tienda
        ii. Manejo de dropdown list en consola para elegir el libro a montar en el carrito (las ventas). IDEAS
4) Módulos de KPI/reportes/Dashboard
    a. Total de ventas por tienda, por libro, por autor (primario y secundario)
    b. Ventas todas y promedios
    c. Otras que hagan sentido con el modelo.
5) Nota importante, firmar todos los métodos con el estándar java doc.
    a. /*
    b. * Autor: matricula y nombre+apellido en el class o interface TOP
    c. * descripción del section fields, constructors, mutators/setters. Getters/accesors, functionalities methods.
    d. * y fuinctional methods objetives*/

Opción II: jueves 17-dec-2023 11:59pm. 20 puntos


jobs
job_id INT(11)
job_desc VARCHAR(50)
min_lvl INT(11)
max_lvl INT(11)
Indexes
fname VARCHAR(20)
minit CHAR(1)
lname VARCHAR(30)
job_id INT(11)
job_lvl INT(11)
pub_id CHAR(4)
hire_date DATETIME
Indexes
pub_info
pub_id CHAR(4)
lo LONGBLOB
pr_info TEXT
Indexes
discounts
discounttype VARCHAR(40)
stor_id CHAR(4)
lowqty SMALLINT(6)
highqty SMALLINT(6)
discount DECIMAL(4,2)
Indexes
roysched
title_id VARCHAR(6)
lorange INT(11)
hirange INT(11)
royalty INT(11)
Indexes
publishers
pub_id CHAR(4)
pub_name VARCHAR(40)
city VARCHAR(20)
state CHAR(2)
country VARCHAR(30)
Indexes
stores
stor_id CHAR(4)
stor_name VARCHAR(40)
stor_address VARCHAR(40)
city VARCHAR(20)
state CHAR(2)
zip CHAR(5)
Indexes
pub_menu
mnu_num INT(11)
mnu_master INT(11)
MNU_NAME VARCHAR(30)
Indexes
titles
title_id VARCHAR(6)
title VARCHAR(80)
type CHAR(12)
pub_id CHAR(4)
price DECIMAL(10,2)
advance DECIMAL(10,2)
royalty INT(11)
ytd_sales INT(11)
notes VARCHAR(200)
pubdate DATETIME
Indexes
sales
stor_id CHAR(4)
ord_num VARCHAR(20)
ord_date DATETIME
qty INT(11)
payterms VARCHAR(12)
title_id VARCHAR(6)
Indexes
authors
au_id VARCHAR(11)
au_lname VARCHAR(40)
au_fname VARCHAR(20)
phone CHAR(12)
address VARCHAR(40)
city VARCHAR(20)
state CHAR(2)
zip CHAR(5)
contract BIT(1)
sales_log
sales_log_id INT(11)
stor_id CHAR(4)
ord_num VARCHAR(20)
title_id VARCHAR(6)
au_id VARCHAR(11)
log_fecha DATETIME
price DECIMAL(10,2)
quantity INT(11)
titleauthor
au_id VARCHAR(11)
title_id VARCHAR(6)
au_ord INT(11)
royaltyper INT(11)
Indexes

Sdelorbe 2023-20 INF-515 LAB 008 DB ORM

Ejemplo entity, model y herencia:

```
package com.cine.Data;

import java.util.Date;
/**
 * UASD in Alliance with Oracle Academic JP-JavaProgramming
 * 2020-10 INF-514-515
 * Class City atributos de este objecto ORM
 *
 * @author Silverio Del Orbe
 * Uso academico exclusivamente
 * Class City hereda del padre Entity
 * bien basico. Es un simple data template  con la columnas de la tabla en el DB
 */
public class Country extends Entity {
                        public  short ID ;
                        public  String Name ;
                        public  Date UpdateDate;
                        public Country () {

                        }
                        public Country(short id, String sname, Date updt  ) {
                                this.ID  = id;
                                this.Name  = sname;
                                this.UpdateDate  = updt;
                        }


}
package com.cine.Data;

import java.time.LocalDate;
/**
 * UASD in Alliance with Oracle Academic JP-JavaProgramming
 * 2020-10 INF-514-515
 * Class Attribute abstract para agrupar todas las enetidades de datos
 * Y servir de transporte del modelo
 * @author Silverio Del Orbe
 * Uso academico exclusivamente
 * Class Entity abstract class agrupador
 * tiene un utilitario estatico para leer la fecha del actual.
 *
 */
public abstract class Entity {
        public static LocalDate getcurrentDate() {
                LocalDate today = LocalDate.now();
                return today ;
        }

}

package com.cine.Model;
```

```
import java.util.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;

import com.cine.Data.Country;
import com.cine.Data.Entity;

/**
 * UASD in Alliance with Oracle Academic JP-JavaProgramming
 * 2020-10 INF-514-515
 * Class gestion ORM List Model de paises (Country entity)
 * @author Silverio Del Orbe
 * Uso academico exclusivamente
 * Class CountryModel  conectada a  Entity<|--Country Data Atribute class
 * Invoca el super (EntityModel) recibe el ResultSet lo mapea y cierra
 *
 */
public class CountryModel extends EntityModel implements iORMObject {
	private ArrayList<Country> allData;
	/**
	 * Unico constructor
	 * Sobrecarga el constructor mas avanzado de padre EntityModel:
	 * -Objeto: Country
	 * -PK: country_id
	 * -Search Col: country
	 * -FK Col: "" -- no tiene FKs
	 * - Order by: country
	 */
	public CountryModel() {
		super("Country","country_id", "country","","country","last_update");
	}
	/**
	 * Recibe la data de search y la pasa al List<Country>
	 * Record by Records
	 * @param rSet ResultSet que recibe la data del Search
	 */
	@Override
	public void Mapping(ResultSet rSet) {
		short id;
		String sname;
		Date dt;
		allData = null; // destroy before list
		allData = new ArrayList<Country>();
		try {
			while( rSet.next()) {

				id = rSet.getShort("country_id");
				sname = rSet.getString("country");
```

```
                        dt = rSet.getDate("last_update");
                        Country objPais = new Country(id, sname, dt);
                        allData.add(objPais);
                }
                rSet.close();
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                super.ActionMessage = e.getMessage();
        }
}
/**
 * Invoca el padre Find() basico sin parametros
 * @return ArrayList<? extends Entity> de Country (alldata)
 */
@Override
public ArrayList<? extends Entity>  Get() {
        // TODO Auto-generated method stub
        Mapping(super.Find());
        return allData;
}
/**
 * Invoca el padre Find(PK) por primary key
 * @return ArrayList<? extends Entity> de Country (alldata)
 */
@Override
public ArrayList<Country>  Get(Object id) {
        // TODO Auto-generated method stub
        Mapping(super.Find(id));
        return allData;
}
/**
 * Invoca el padre Find(texto, object) por busqueda like abierta
 * @return ArrayList<? extends Entity> de Country (alldata)
 */
public ArrayList<Country>  Get(String search) {
        // TODO Auto-generated method stub
        Mapping(super.Find(search));
        return allData;
}
/**
 * Invoca el padre Find(texto, object) por busqueda like abierta y por
 * un FK preestablecido en la construccion "country_id"
 * @return ArrayList<? extends Entity> de Country (alldata)
 */
@Override
public ArrayList<Country>  Get(String search, Object fkval)  {
        // TODO Auto-generated method stub
        //throw new Exception("Esta entidad no posee search FK");
        return null;
}
/**
 * Invoca el padre Find(date, date) en un rango de fecha
 * Segun columna especificada en la construccion
```

```
 * @return ArrayList<? extends Entity> de Country (alldata)
 */
public ArrayList<Country>  Get(Date dtein, Date dteout) {
	// TODO Auto-generated method stub
	Mapping(super.Find(dtein,dteout));
	return allData;
}
@Override
public boolean Update(Entity odata) {
	// TODO Auto-generated method stub
	return super.Put(SerializerMap(odata));
}
/**
 * Recibe la entidad Country y hace la actualizacion mediante el padre
 * super.Put
 * @return boolean indicando positivo o negativo
 */
public ArrayList<Country>  Get(boolean pisfull) {
	// TODO Auto-generated method stub
	if (pisfull)
		Mapping(super.Find(true));
	else
		Mapping(super.Find());
	return allData;
}
/**
 * Recibe la entidad Country y hace un nuevo mediante el padre
 * super.Put. Primero busca el Max Country_id,
 * segun method en le padre geMaxID
 * @return boolean indicando positivo o negativo
 */
@Override
public boolean Add(Entity odata) {
	// TODO Auto-generated method stub
	// busca el ultimo PK y le suma 1;
	((Country)odata).ID = Short.parseShort(Long.toString(getMaxID()+1));
	return super.Post(SerializerMap(odata));
}
/**
 * Recibe la entidad Country y hace un delete padre
 * super.Put. NO IMPLEMENTADO AUN.
 * @return boolean indicando positivo o negativo
 */
@Override
public boolean Delete(Entity odata)  {
	// TODO Auto-generated method stub
	return false;
}
/**
 * Pasa el Country a un HaspMap, para poderselo pasar al padre como parametro
 * Universal de interface con este para C.U.D (Create o update, DELETE)
 * super.Put. NO IMPLEMENTADO AUN.
 * @return boolean indicando positivo o negativo
```

```
	 */
	@Override
	public HashMap<String,String> SerializerMap(Entity odata){
		if (!(odata instanceof Country))
			return null;
		Country obj1 = (Country)(odata);
		HashMap<String,String> paisObj = new HashMap<String, String>();
		paisObj.put("country_id",Integer.toString(obj1.ID));
		paisObj.put("country",obj1.Name);
		paisObj.put("last_update",Entity.getcurrentDate().toString());
		return paisObj;
	}
	/**
	 * Lleva la List<Country> a Json string
	 * Basic Serializer.... cuando hay composicion hay que mejorar
	 * Esto se hace con Dynamic class loader invocation o reflexion.
	 * @return String con el formato json
	 */
	@Override
	public String Serializer() {
		// TODO Auto-generated method stub
		StringBuilder sb = new  StringBuilder("[");
		char separa = ' ';
		for (Country octy : allData) {
			sb.append(separa +"{ID:" + octy.ID);
			sb.append(",Name:\"" + octy.Name);
			sb.append(",UpdateDate:\"" + octy.UpdateDate);
			sb.append("\"}");
			separa = ',';
		}
		sb.append("]");

		return sb.toString();
	}
	/**
	 * Busqueda en Memoria sobre el List<>
	 * Landa basico forech and filter
	 * Esta es la gran tendencia, para no tener que ir tantas veces
	 * Al oltp.
	 * @return Entity buscada o null
	 */
	@Override
	public Entity inMemSearch(Object pid) {
		// TODO Auto-generated method stub
		Country oPais = null;
		  List<Country> findPais = allData.stream()		// convert list to stream
		.filter(octry ->  octry.ID == Short.parseShort(pid.toString()))	// we dont like mkyong
		.collect(Collectors.toList());
		if (findPais.size()>=1)
			oPais = findPais.get(0);
		return oPais;
	}
	/**
```

```
	 * getters/accessors con la data de la ultima busqueda list<Country>
	 *
	 * @return ArrayList<? extends Entity> buscada o null
	 */
	@Override
	public ArrayList<? extends Entity> getData() {
		// TODO Auto-generated method stub
		return allData;
	}
}


package com.cine.Model;



import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;



import com.cine.Data.Entity;



/**
 * UASD in Alliance with Oracle Academic JP-JavaProgramming
 * 2020-10 INF-514-515
 * Interface general de modelo para los hijos Entity que se comunicaran
 * Con los consumidores y modelaran los atributos de cada entidad
 * @author Silverio Del Orbe
 * Uso academico exclusivamente
 * Interface iORMObject para definir las operaciones estanndar que debe cumplicar cada objeto
 * A nivel de Data y los mapping para el modelo CRUD ORM
 *
 */
public interface iORMObject {
	public void Mapping(ResultSet rSet);
	public HashMap<String, String> SerializerMap(Entity odata);
	public ArrayList<? extends Entity> Get();
	public boolean Add(Entity odata);
	public boolean Update(Entity odata);
	public boolean Delete(Entity odata);
	public String Serializer(); // json {...}
	public Entity inMemSearch(Object pid);
	public ArrayList<? extends Entity> getData();
	public ArrayList<? extends Entity> Get(Object id);
	public ArrayList<? extends Entity> Get(String search, Object fklink);
```

```
        public ArrayList<? extends Entity> Get(String search);
        public ArrayList<? extends Entity> Get(Date dtein, Date dteout);

}

package com.cine.Model;

import java.sql.Statement;
import java.io.Closeable;
import java.sql.Connection;
import java.util.ArrayList;
import java.util.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;

import com.cine.util.PropertyFile;


/**
 * UASD in Alliance with Oracle Academic JP-JavaProgramming
 * 2020-10 INF-514-515
 * Entity model ORM padre que abstrae/encapsula toda la conectividad con la base de datos
 * Basada en el estandar ORM y CRUD: Post, Put, Delete, Read (find)
 * @author Silverio Del Orbe
 * Uso academico exclusivamente
 * Class EntityModel base de principal del modelo y la conectividad con DB
 * Puede subir a un nivel mas alto con el DBContext o Algo Asi (ETF), pero para este prototipo
 * es suficiente. Object o connection Pool, se puede ver EE (Java 2)
 * Los metodos son protected  o private. Solo los hijos pueden invocar
 */
public class EntityModel implements Closeable  {
        //Entity fields
        private String _objectName;
        private String _PKColumn;
        private String _SearchExpresion;
        private String _FKColumn;
        private String _ORDColumns;
        private String _DateColumns;
        private short _RECTop;

        // sql objects. PreparedStatement JDBC
        private Connection oConn = null;
        private ResultSetMetaData ObjectMeta = null;
        private HashMap<String,Integer> ColIndexs = null;
        private PreparedStatement prepstmPK  = null;
        private PreparedStatement prepstmSearch = null;
        private PreparedStatement prepstmDateRange = null;
        private PreparedStatement prepstmSearchFK = null;
```

```
	private PreparedStatement prepstmDef = null;
	private PreparedStatement prepstmMAX = null;
	private PreparedStatement prepstmFull = null;
	private Statement rawstm = null;
	private ArrayList<Statement> AllStms = null;
	// other vars
//	private String _vCatalog;
	protected String ActionMessage;
	private final String  GetsqlPattern = " SELECT * \n FROM {<obj>} \n WHERE {<filter>} ORDER BY
{<order>} LIMIT 0,{<reccount>} ";

	/**
	 * Main constructor: Con el nombre de la Tabla, la columna PK y la o las columna de busquedas
	 * @param objName string con el nombre del objeto o tabla
	 * @param pkcol string con el nombre de la columna PK de la tabla
	 * @param shexp string con el nombre de la o las columna(s) de busqueda.
	 *  Ej: city, lastname+firstname, mysql concat(lastname,firstname), etc.
	 */
	public EntityModel(String objName, String pkcol,String  shexp) {
		this._objectName = objName;
		this._PKColumn = pkcol;
		this._SearchExpresion =  shexp;
		this._FKColumn = "";
		this._DateColumns = "";
		this._RECTop = 10;
		this._ORDColumns = "2,1";
		initEntity();
	}
	/**
	 * Main constructor: Con el Tabla, PK , col busquedas, FK col y Order By Col
	 * @param objName string con el nombre del objeto o tabla
	 * @param pkcol string con la columna PK de la tabla
	 * @param shexp string con la o las columna(s) de busqueda.
	 * @param fkcol string con la col FK o enlace de la tabla.
	 * ----------Puede ser mas de , pero para eso hay que hacer otro constructor con una lista
	 * ----------Y mejorar los metodo que trabajan con esta. Esto es con fines de busqueda solamente
	 * @param ordcol string con la o las col de ordemamiento. El default es 2,1
	 *  Ej: city, lastname+firstname, mysql concat(lastname,firstname), etc.
	 */
	public EntityModel(String objName, String pkcol,String  shexp, String fkcol, String ordcol) {
		this._objectName = objName;
		this._PKColumn = pkcol;
		this._SearchExpresion =  shexp;
		this._FKColumn = fkcol;
		this._RECTop = 10;
		this._ORDColumns = "2,1";
		if (ordcol != "")
			this._ORDColumns = ordcol;

		initEntity();
	}
	/**
	 * Main constructor: Con el Tabla, PK , col busquedas, FK col y Order By Col
```

```
	 * @param objName string con el nombre del objeto o tabla
	 * @param pkcol string con la columna PK de la tabla
	 * @param shexp string con la o las columna(s) de busqueda.
	 * @param fkcol string con la col FK o enlace de la tabla.
	 * ----------Puede ser mas de , pero para eso hay que hacer otro constructor con una lista
	 * ----------Y mejorar los metodo que trabajan con esta. Esto es con fines de busqueda solamente
	 * @param ordcol string con la o las col de ordemamiento. El default es 2,1
	 * @param dtecol string con la de fecha para preparar un between search de esta.
	 *  Ej: city, lastname+firstname, mysql concat(lastname,firstname), etc.
	 */
	public EntityModel(String objName, String pkcol,String  shexp, String fkcol, String ordcol, String
dtecol) {
		this._objectName = objName;
		this._PKColumn = pkcol;
		this._SearchExpresion =  shexp;
		this._FKColumn = fkcol;
		this._RECTop = 10;
		this._ORDColumns = "2,1";
		if (ordcol != "")
			this._ORDColumns = ordcol;
		this._DateColumns = dtecol;
		initEntity();
	}
	/**
	 * Find principal del modelo. Default 10 records Top
	 * @return ResultSet con la data buscada
	 */
	protected ResultSet Find()  {
		ResultSet rSet = null;
		try {
			rSet = prepstmDef.executeQuery();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage =  e.getMessage();
		}
		return rSet;
	}
	/**
	 * Find 2do del modelo. Extiende a 5000 record max
	 * @return ResultSet con la data buscada
	 */
	protected ResultSet Find(boolean isFull)  {
			ResultSet rSet = null;
			try {
				rSet = prepstmFull.executeQuery();
			} catch (SQLException e) {
				// TODO Auto-generated catch block
				ActionMessage =  e.getMessage();
			}
			return rSet;
	}
	/**
	 * Find 3ro del modelo. busqueda por PK col
```

```
	 * El mas ejemonico
	 * @param pkval Object de cualquier tipo como valor para el PK
	 * @return ResultSet con la data buscada
	 */
	protected   ResultSet Find(Object  pkval) {
		ResultSet rSet = null;
		try {
			prepstmPK.setObject(1, pkval);
			rSet = prepstmPK.executeQuery();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage =  e.getMessage();
		}
		return rSet;


	}
	/**
	 * Find 4to del modelo. busca en un rango de fecha
	 * si este campo fue especificado en el constructor
	 * @param dtein Date donde inicia el intervalo
	 * @param dteout Date donde cierra intervalo
	 * @return ResultSet con la data buscada
	 */
	protected   ResultSet Find(Date  dtein, Date dteout )  {
		ResultSet rSet = null;
		try {
			/*java.sql.Date sqldt1 = java.sql.Date.valueOf(dtein.toString());
			java.sql.Date sqldt2 = java.sql.Date.valueOf(dteout.toString());

			*/
			java.sql.Date sqldt1 = new java.sql.Date(dtein.getTime());
			java.sql.Date sqldt2 = new java.sql.Date(dteout.getTime());
			prepstmDateRange.setDate(1, sqldt1);
			prepstmDateRange.setDate(2, sqldt2);
			rSet = prepstmDateRange.executeQuery();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage =  e.getMessage();
		}
		return rSet;
	}
	/**
	 * Find 5to del modelo. busca en base a un criterio de busqueda
	 * Texto searchers.
	 * @param search string con el criterio para LIKE
	 * @return ResultSet con la data buscada
	 */
	protected   ResultSet Find(String  search)  {
		ResultSet rSet= null;
		try {
			prepstmSearch.setString(1, search);
			rSet = prepstmSearch.executeQuery();
```

```
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage =  e.getMessage();
		}
		return rSet;

	}
	/**
	 * Find 6to del modelo. busca en un rango de fecha
	 * si este campo fue especificado en el constructor
	 * @param search string con el criterio
	 * @param dteout Object con el valor para el FK (City.country_id)
	 * @return ResultSet con la data buscada
	 */
	protected   ResultSet Find(String  search, Object fkval) {
		ResultSet rSet= null;
		try {
			prepstmSearchFK.setString(1, search);
			prepstmSearchFK.setObject(2, fkval);
			rSet = prepstmSearchFK.executeQuery();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage =  e.getMessage();
		}

		return rSet;
	}
	/**
	 * Find 6to del modelo. busca en un rango de fecha
	 * si este campo fue especificado en el constructor
	 * @param strsql string con SQL Completo o Rawsql que se desea ejecutar
	 * @param boolean indicador de que se quiere hacer un rawsql sin prepare=true
	 * @return ResultSet con la data buscada
	 */
	protected   ResultSet Find(String  strsql, boolean rawsql)  {
		if (!rawsql)
			return null;
		ResultSet rSet= null;
		try {
			rSet = rawstm.executeQuery(strsql);
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage =  e.getMessage();
		}
		return rSet;
	}
	/**
	 * Update o Actualizacion de data del modelo/entity
	 * Recibe un HashMap con el campo y su valor <string,string>
	 * Y prepara la sentencia SQL con estos
	 * Dicho map se crea en los hijos
	 * @param putdatos HashMap<String,String>  con el par ordenado col-->valor
	 * @return boolean  true si la actualizacion fue exitosa o false de lo contrario
```

```
	 */
	protected boolean Put(HashMap<String,String> putdatos) {
		boolean blnresult = false;
		String ssql = prepareUpdate(putdatos);
		try {
			rawstm.execute(ssql);
			blnresult = true;
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage = e.getMessage();
			//e.printStackTrace();

		}
		return blnresult ;
	}
	/**
	 * INSERT o agregar un nuevo record.
	 * @param putdatos HashMap<String,String> con el par ordenado col-->valor
	 * @return boolean  true si la actualizacion fue exitosa o false de lo contrario
	 */
	protected   boolean Post(HashMap<String,String> putdatos) {
		boolean blnresult = false;
		String ssql = prepareInsert(putdatos);
		try {
			rawstm.execute(ssql);
			blnresult = true;
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			ActionMessage = e.getMessage();
			//e.printStackTrace();

		}
		return blnresult;
	}
	/**
	 * Interno init Class o Arranque del ambiente
	 * Construye los Statament, Connection, DriverManager
	 * Le el config.propeties mediante la clase Utilitaria
	 * com.cine.util.PropertyFile
	 */
	@SuppressWarnings("deprecation")
	private void initEntity() {
		try {
			if (_objectName.trim().length()==0  ||
					_PKColumn.trim().length()==0||
					_SearchExpresion.trim().length()==0)
					throw new Exception("Campos requeridos no fueron
suministrados (obj, PK, search)");

				PropertyFile objsetting = new PropertyFile();
				String sdriver =  objsetting.getPropValue("dbdriver");
				String dburl = objsetting.getPropValue("dburl");
			//	_vCatalog = objsetting.getPropValue("dbcatalog");
```

```
                                        dburl += "?user=" +objsetting.getPropValue("dbuser");
                                        dburl += "&password=" +objsetting.getPropValue("dbpassword");
                                        // begin conexion opening
                                        Class.forName(sdriver).newInstance();
                                        // ejemplo
"jdbc:mysql://localhost:3306/sakila?user=getrootuser&password=Java*mYsqldB"
                                        oConn = DriverManager.getConnection(dburl);
                                        rawstm = oConn.createStatement() ;
                                        prepareStms();
                                        Object obj = -3.1416+2.7118;
                                        ResultSet rset = Find(obj);
                                        ObjectMeta = rset.getMetaData();
                                        prepareMetaIndex();
                                        rset.close();

                                }catch(InstantiationException| IllegalAccessException| ClassNotFoundException E)
{
                                        ActionMessage = E.getMessage();
                                } catch (SQLException e) {
                                        // TODO Auto-generated catch block
                                        ActionMessage = e.getMessage();
                                } catch (Exception e) {
                                        // TODO Auto-generated catch block
                                        ActionMessage = e.getMessage();
                                }
        }
        /**
         * Interno PreparedStatement prepara todos los modelos de queries
         * que admite el modelo, segun los parametros de construccion
         */
        private void prepareStms() throws SQLException {
                // simgle PK only. No multiple
                String ssql = GetsqlPattern.replace("{<obj>}", _objectName);
                ssql = ssql.replace("{<order>}", _ORDColumns);
                ssql = ssql.replace("{<reccount>}", Integer.toString(_RECTop));
                prepstmDef= oConn.prepareStatement(ssql.replace("{<filter>}"," 1=1 "));
                prepstmPK = oConn.prepareStatement(ssql.replace("{<filter>}", _PKColumn + " =  ? " ));
                prepstmSearch = oConn.prepareStatement(ssql.replace("{<filter>}", _SearchExpresion + "
LIKE concat('%' ,  ? , '%')"));
                if (_DateColumns.trim().length()>0)
                        prepstmDateRange = oConn.prepareStatement(ssql.replace("{<filter>}",
_DateColumns + " BETWEEN ?  AND ? ") );
                if (_FKColumn.trim().length()>0)
                        prepstmSearchFK = oConn.prepareStatement(ssql.replace("{<filter>}",
_SearchExpresion + " LIKE concat('%' ,  ? , '%') \n AND " + _FKColumn  + " = ?"));
                String maxsql = "SELECT MAX("+ _PKColumn + ") FROM " + _objectName;
                prepstmMAX = oConn.prepareStatement(maxsql);
                maxsql = "SELECT * FROM " + _objectName + " LIMIT 0,5000"; // MAX 5000 FOR fkS o full
data open
                prepstmFull = oConn.prepareStatement(maxsql);

                ArrayList<Statement> AllStms = new ArrayList<Statement>();
                AllStms.add(prepstmDef);// 1) defaul 10 records
```

```
                AllStms.add(prepstmPK);// 2) PK closed search
                AllStms.add(prepstmSearch);// 3) Text search central
                AllStms.add(prepstmDateRange);// 4) Date Range Searchs
                AllStms.add(prepstmSearchFK);// 5) FK + text search
                AllStms.add(prepstmMAX);// 6) Select Max(PK) col for insert propositos
                AllStms.add(prepstmFull);// 7) Full Data search 5000
                AllStms.add(rawstm);// 8) Free SQL port

        }
        /**
         * Prepara el string SQL para Update
         * Utilitario interno
         */
        private String prepareUpdate(HashMap<String,String> putdatos) {
                StringBuilder sb = new StringBuilder("UPDATE " + _objectName + "\n SET ");
                char separe = ' ';
                for (Map.Entry<String,String> one :  putdatos.entrySet())
                        if (one.getKey().compareToIgnoreCase(_PKColumn) != 0) {
                                sb.append("\n" + separe +  one.getKey() + "=" +
getSQLValue(one.getKey() , one.getValue()));
                                separe = ',';
                        }
                sb.append("\n WHERE " + _PKColumn + " = " + getSQLValue(_PKColumn ,
putdatos.get(_PKColumn) ) );
                return sb.toString();
        }
        /**
         * Prepara el string SQL para Insert
         * Utilitario interno
         */
        private String prepareInsert(HashMap<String,String> adddatos) {
                StringBuilder sb = new StringBuilder("INSERT INTO " + _objectName + " (");
                StringBuilder svalues = new StringBuilder(" VALUES(");
                char separe = ' ';
                for (Map.Entry<String,String> one :  adddatos.entrySet()) {
                                sb.append("\n" + separe +  one.getKey() );
                                svalues.append("\n" + separe + getSQLValue(one.getKey() ,
one.getValue()));
                                separe = ',';

                        }
                sb.append(") \n " + svalues + ")" );
                return sb.toString();
        }
        /**
         * Retorna el SQL valor entre '' si es naturaleza no numerica
         * de lo contrario directo
         * Utilitario interno
         */
        private String getSQLValue(String k, String v) {
                if (isColNumeric(ColIndexs.get(k)))
                        return v;
                else
```

```
                return "'"+ v + "'";
        }
        /**
         * Toma el tipo de dato de la Meta Data y lo fija en un HashMap
         * Para poderlo leer por nombre de columna.
         * Utilitario interno
         */
        private void prepareMetaIndex() throws SQLException {
                ColIndexs = new HashMap<String, Integer>();
                for (int k = 1; k <= ObjectMeta.getColumnCount();k++)
                        ColIndexs.put(ObjectMeta.getColumnName(k), ObjectMeta.getColumnType(k));
        }
        /**
         * Verifica la naturaleza del tipo de dato
         * @return boolean indicando que es numerico.
         * Trabaja basado en un enumerado
         * Utilitario interno java.sql.Types
         */
        private boolean isColNumeric(int ctype) {
                boolean yrst = false;
                switch(ctype){
                        case java.sql.Types.DECIMAL:
                        case java.sql.Types.INTEGER:
                        case java.sql.Types.BIGINT:
                        case java.sql.Types.FLOAT:
                        case java.sql.Types.SMALLINT:
                        case java.sql.Types.NUMERIC:
                        case java.sql.Types.BIT:
                        case java.sql.Types.BOOLEAN:
                                yrst = true;
                                break;
                }
                return yrst;
        }
        /**
         * Ejecuta la sentencia PreparedStatement para buscar el maximo
         * De la entidad. El objetivo es apoyar los ADD o create cuando el PK
         * Es numerico.
         * Este punto se puede combinar con secuencia, segun la maneje cada motor
         * @return
         */
        protected long getMaxID() {
                long ymax=-1;
                try {
                        ResultSet rmax = prepstmMAX.executeQuery();
                        rmax.next();
                        ymax = rmax.getInt(1);
                        rmax.close();
                }catch (SQLException e) {
                        // TODO: handle exception
                        ActionMessage =  e.getMessage();
                }
```

```
		return ymax;
	}
	/**
	 * Cierra todos los elementos abieros
	 * Statement, Connection, Etc.
	 * ------ Los resultSet, lo cierran los hijos una vez hacen el mapping
	 * punto de destruccion/destroy
	 */
	@Override
	public void close() {
		try {
			// close all Statements open
			for (Statement stm : AllStms) {
				if (stm != null && !stm.isClosed())
					stm.close();
				stm = null;
			}

			oConn.close();
			oConn = null;
			ObjectMeta = null;


		} catch (SQLException  e) {
			// TODO Auto-generated catch block
			ActionMessage = e.getMessage();
		}catch(Exception e) {
			ActionMessage = e.getMessage();
		}
	}
	/**
	 * Retorna el mensaje de posible error
	 * Durante las operaciones
	 * @return string con el ActionMessage en los try catch
	 */
	public String getMessage() {
		return ActionMessage;
	}


}
```