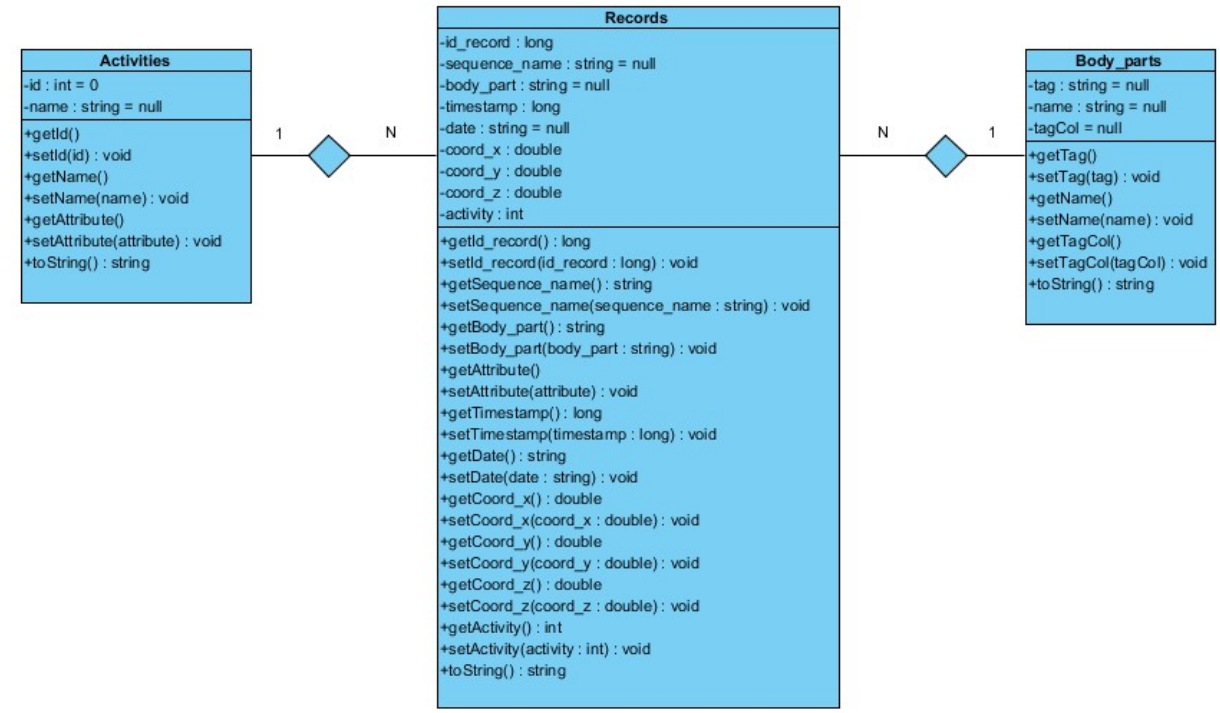
DISSENY DE BASES DE DADES

Pràctica 3: Memòria

Albert Juhé Lluveras NIA: 122149

Erik Gotera León NIA: 124834

Diagrama UML



Explicación breve de cada clase

dbd2012_p3.java

Clase ejecutable que se encarga de arrancar todo el proceso. Muestra el menú con las opciones disponibles.

Métodos:

private void dropDatabase()

Llama al método de FuncionsBD.java de vaciar la base de datos.

private void show(String categoryToDisplay)

Función para centralizar todas las llamadas a show y crear los filtros necesarios para cada acción distinta.

private void readFile():

Lee el fichero con todos los datos a añadir en la BD línea a línea y llama al método.

private void procesarFichero(String linea)

Procesa línea a línea y llama al método **addRegistro** de la clase **FuncionsBD()**.

FuncionsBD.java

Clase que se contiene la implementación de todas las funciones que realiza nuestra aplicación.

Métodos:

void addRegistro()

Método que añade cada registro a la BD

Double getConfianza()

Método que pide al usuario un valor para la **confianza** y lo devuelve

Double getSupport()

Método que pide al usuario un valor para el **support** y lo devuelve

void printInterest()

Método que imprime por pantalla la explicación del **interés**.

void showRecordsFromFilterOptimized(String filter)

Muestra las reglas obtenidas mediante SON y Apriori del conjunto de datos obtenido de aplicar el filtro *filter*.

boolean hasPair(Records r, Pair pair)

Nos dice si un registro **"r"** contiene un par (atributo-valor) **"pair"**

Double parseMin(String original)

Double parseMax(String original)

Estos dos métodos parsean Strings cómo estos:

- '(6.3379022611088E17-6.3379022625044365E17]'
- '(-inf-6.3379022645358221E17]'
- '(6.3379022656994368E17-inf)'

Y devuelven el número Double mínimo y el máximo, por ejemplo, en los casos anteriores devolverían (parseMin / parseMax):

- 633790226110880000 / 633790226250443650
- -inf / 633790226453582210
- 633790226569943680 / inf

void showRecordsFromFilter(String filter)

Muestra los registros y las reglas obtenidas mediante Apriori a partir de un filtro dado

void showRecordsFromFilter(String filter)

Limpia de registros las tablas de toda la Base de datos.

public Apriori applyApriori(String dataSetName, Double conf, Double sup)

Método que pasándoles la ruta del archivo discretizado, el valor de la confianza y el del support nos aplica Apriori para el **dataSetName**.

protected static Instances load(String filename)

protected static void save(Instances data, String filename)

protected static void csvToArff(String csv, String arff)

Método que pasa de csv a arff.

private void discretizar(String arff, String arffDisc)

Método que discretiza un arff y lo guarda en otro arff.

private BufferedWriter camposCsv(BufferedWriter bw)

Método que pone el encabezado en los csv que se crean.

Rule.java

Clase que define una regla, eso es, un conjunto de pares (atributo-valor) dependiendo de si son causa (están en la izquierda) o consecuencia (están en la derecha).

Pair

Es una clase que guarda el par atributo-valor.

toString

Imprimi la regla, su confianza y los contadores de cuantas veces se ha encontrado.

RuleComparator.java

Se utiliza para ordenar la lista de reglas, de forma que las que tienen mayor confianza aparezcan primero.

Activities.java

Clase que define los atributos de la tabla Activities en la BD

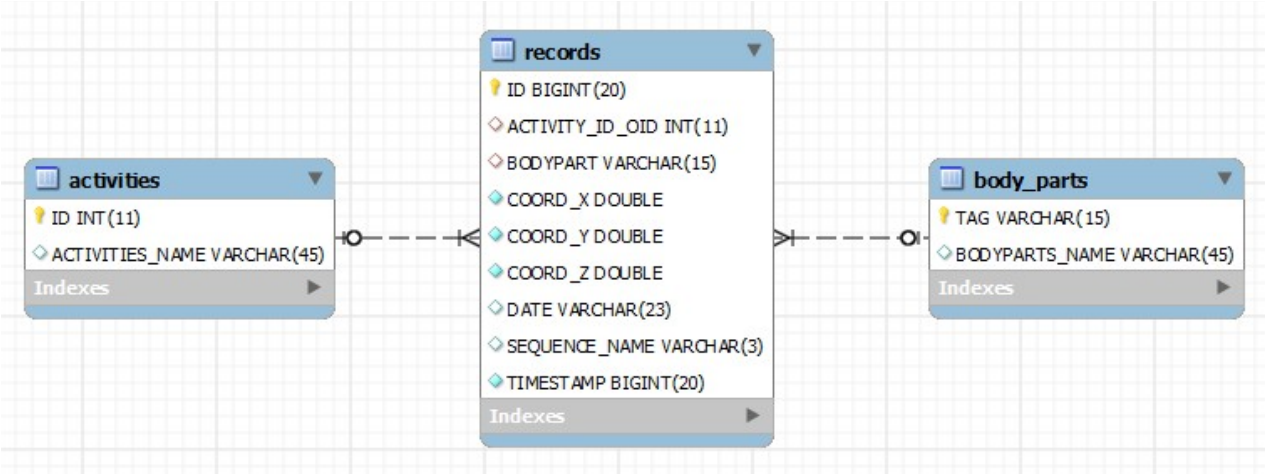
Body_parts.java

Clase que define los atributos de la tabla Body_parts en la BD.

Records.java

Clase que define los atributos de la tabla Records en la BD.

EER Diagram


records
ID BIGINT(20)
ACTIVITY_ID_OID INT(11)
BODYPART VARCHAR(15)
COORD_X DOUBLE
COORD_Y DOUBLE
COORD_Z DOUBLE
DATE VARCHAR(23)
SEQUENCE_NAME VARCHAR(3)
TIMESTAMP BIGINT(20)
Indexes
activities
ID INT(11)
ACTIVITIES_NAME VARCHAR(45)
Indexes
body_parts
TAG VARCHAR(15)
BODYPARTS_NAME VARCHAR(45)
Indexes