Módulo de Predicciones de Osmius

Sistemas Informáticos. Curso 2008/2009

Rafael García Aranda

Directores:

Luis Fernando Llana Díaz

Maria de las Mercedes García Merayo

Departamento de Sistemas Informáticos y Computación

Facultad de Informática

Universidad Complutense de Madrid

Resumen del proyecto

Resumen

En este proyecto hemos dotado con un módulo de predicciones al sistema de monitorización Osmius, desarrollado por PeopleWare bajo licencia GPL. El objetivo de dicho módulo es conseguir predecir posibles acontecimientos futuros perjudiciales, por medio del conocimiento pasado, con el fin de poder evitarlos. Éste herramienta dota a Osmius de gran potencia e incrementa su valor como software de monitorización.

En el proceso de desarrollo del proyecto se han implementado una serie de algoritmos de aprendizaje con los que realizar las predicciones y una base de datos consistente en la que poder almacenarlos. Además, todos estos algoritmos están fuertemente parametrizados para dar al usuario gran libertad de acción a la hora de realizar dichas predicciones.

Abstract

Within this project, we have provided with a prediction module to the monitoring sofware Osmius, developed by PeopleWare under GPL lincese. The goal of this module is to predict, by using knowledge of past events, future and (potentially) harmful events in order to avoid them. This tool will provide Osmius with a great power and it will increase its value as monitoring software.

In the developing of this project, we have implemented learning algorithms which allow us to make predictions, and a consistent database where the data can be stored. Moreover, these algorithms are heavily parametrized in order to give a big range of options to the user when making those predictions.

Lista de palabras clave

Monitorización, Osmius, predicción, aprendizaje, agentes, instancias, eventos, minería de datos y patrones frecuentes.

Autorización

Autorizo a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor, tanto la memoria, como el código, la documentación y/o prototipo desarrollado.

Firmado:

Rafael García Aranda

Índice general

Capítulo 1

Introducción

Este documento contiene información relativa a los conceptos, el diseño y la implementación del módulo de predicciones de Osmius.

Osmius es un sistema de supervisión de elementos heterogéneos en red basado en una estructura de Servidor Central y Agentes, multi-plataforma y de código abierto. Como tal es capaz de proporcionarnos información precisa sobre el estado del medio al que esta conectado en tiempo real; dicha información puede ser capturada de diferentes medios. Ello hace que este sistema sea considerado de gran utilidad para empresas u organismos cuya infraestructura requiere diferentes elementos de red, servidores y aplicaciones, y necesiten llevar un control de los mismos.

Es evidente que con un software de este tipo podemos ahorrar tiempo en la detección de lo que puede estar fallando en nuestro sistema, cuando a fallado o por qué, pero también ofrece la posibilidad de llegar más allá: prevenir que se produzcan dichos fallos.

Imaginemos que somos una empresa que se dedica a dar soporte a otras y que el servidor al que dichas empresas se conectan para obtener que tengan contratado falla y se tarda algún tiempo en tenerlo de nuevo operativo. ¿Sabemos cuanto dinero ha costado esta caída a nuestra empresa? ¿No sería mejor haber evitado dicha caída?

En definitiva, ese es el principal objetivo de este proyecto. Mediante la observación de los eventos que captura Osmius, conseguimos realizar un aprendizaje en forma de reglas o secuencias de eventos, que son capaces de predecir con cierta probabilidad qué es lo que ocurrirá en el medio monitorizado y a qué es debido.

1.1. Terminología básica

A continuación se introduce toda la terminología necesaria para poder comprender el resto del trabajo.

- **Instancia:** Todo elemento susceptible de ser monitorizado. La monitorización se puede realizar respecto a diferentes parámetros. Por ejemplo, en el caso de un servidor, podemos observar su tiempo de respuesta, la carga del procesador o la temperatura que alcanza.

- **Tipo de Instancia:** Toda instancia que pueda ser monitorizada ha de tener un tipo asociado por el que se pueda clasificar y con el que podamos diferenciar sus características. Esto es necesario, ya que no es lo mismo si hablamos de un servidor de Linux, una base de datos o un regulador de temperatura. Tampoco usaremos las mismas variables de control; por ejemplo, hablamos del uso de CPU de un servidor o de la capacidad disponible en disco duro, pero no de la presión o la temperatura del sistema.

- **Evento:** Mensaje obtenido de una instancia monitorizada. La criticidad de un evento es valorada como: informativa (verde), sospechosa (naranja) o critica(roja) y es representada mediante un valor numérico. Los límites que determinan estos estados están preestablecidos por el sistema.

- **Tipo de Evento:** Es cada una de las diferentes variables sobre las que podemos preguntar en un determinado tipo de instancia. Todos los eventos tienen un tipo asociado. Por ejemplo, un evento con un valor de un 80 % de CPU es de tipo "CPU usado".

- **Estado de una Instancia:** El estado de una instancia depende de la criticidad de los eventos activos asociados a esa instancia. Osmius asigna a la instancia el peor de los estados de sus eventos.

- **Disponibilidad de una Instancia:** La disponibilidad de una instancia depende de la criticidad de los eventos asociados a ésta. En Osmius, el experto que diseñe los eventos para un tipo de instancia seleccionará cuales afectaran a la disponibilidad de dicha instancia. Por ejemplo, el evento que indica la disponibilidad de una instancia de un tipo de servidor de Linux es la carga del procesador.

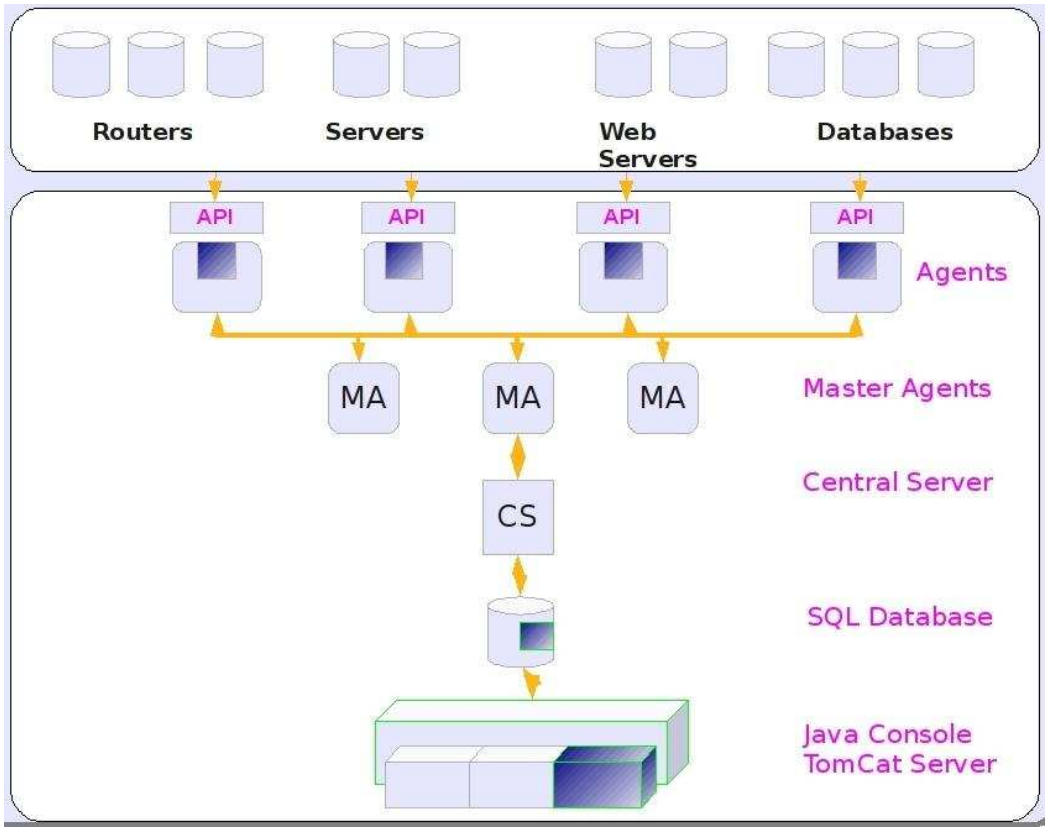
- **Eventos Activos y No Activos:** Los eventos activos son aquellos a los que les afecta el monitoreo de las instancias activas. Cuando un usuario reconoce un evento, este pasa a formar parte del histórico de eventos que podrá ser consultados posteriormente sin afectar a las instancias que se estén monitorizando en ese momento. La mayoría de eventos realizan este proceso automáticamente.

- **Servicios:** Un servicio es la organización lógica de las instancias tales que su conjunto realiza cierta funcionalidad en Osmius. Por ejemplo, el servicio de intranet de una compañía puede estar formado por el servidor de Windows, el servidor de Linux y el servidor de base de datos MySQL. Si alguno de ellos cambiase su disponibilidad, este hecho afectaría al servicio.

- **Acuerdo de Nivel de Servicio o SLA (Service Level Agreement):** En Osmius, todos los servicios están asociados a objetivos relativos a la disponibilidad y al porcentaje de tiempo que han necesitado para estar otra vez en funcionamiento. Dichos objetivos se pactan en contrato con el cliente o usuario y hace referencia al rendimiento esperado de una instancia o servicio.

- **Agente:** Un agente es un proceso software que se ejecuta en una plataforma y es capaz de obtener eventos o instancias de cierto tipo para monitorizar su estado. Es muy importante que un agente no se ejecute si no se lo pide el Agente Maestro (Master Agent), ya que puede afectar al rendimiento de todo el sistema. Existen agentes que tienen que ejecutarse en la misma máquina que la instancia que monitorizan (por ejemplo, si monitorizan un sistema operativo), pero normalmente pueden monitorizar cualquier elemento que este conectada a la misma red. Los agentes tiene distintos estados (en ejecución, parado y error) que nos informan de que está ocurriendo.

- **Agente Maestro:** Un agente maestro es un proceso software que se ejecuta en una máquina, capaz de gestionar a otros agentes y que transmite al servidor central todos los eventos que cada uno de dichos agentes obtiene. El agente maestro es el único capaz de comunicarse con dicho servidor, enviándole los eventos y recibiendo nuevas tareas.



- Servidor Central: Esta formado por una serie de procesos que se ejecutan simultáneamente en una misma máquina. En el se centralizan todas las acciones de monitorización que a continuación se describen:

  - Administrador de Eventos: Recibe todos los eventos de cada agente maestro distribuido por la plataforma y los almacena en la base de datos central.
  - Administrador de Tareas: Procesa las tareas periódicamente, enviando los comandos pertinentes a los agentes maestros. Es decir, cuando un usuario da una orden desde la consola, el tiempo que transcurre hasta que se ejecuta depende de lo que retrase la asociación de dicha tarea a un agente maestro.
  - Administrador del Estado de la Infraestructura: Es el encargado de verificar periódicamente el estado de cada agente maestro y comprobar si se está ejecutando de acuerdo a como se le indicó. En caso de que haya algún error, ordena al agente maestro que lo corrija, si fuese posible.
  - Administrador de Minería de Datos: Es el encargado de calcular periódicamente la nota global del sistema y seleccionar los datos necesarios para su explotación.

  - Administrador de Notificaciones: Procesa periódicamente los cambios en los servicios, instancias, SLA, etc, enviando estos datos a los usuarios registrados por e-mail, SMS, etc.

- Panel de Control: Es el interfaz gráfico desde el que podemos manejar todas las acciones de Osmius y, en particular, podemos monitorizar toda la infraestructura.

Capítulo 2

Fases del proyecto

El proyecto consta de varias fases que han dado lugar a diferentes productos que actúan como módulos independientes al integrarlos en Osmius.

Las fases principales son tres: instalación, aprendizaje y predicción. En la primera fase se instala la base de datos donde se almacenará toda la información que permite ejecutar correctamente las fases de aprendizaje y predicción. En la fase de aprendizaje se realiza la extracción de los datos y creación de las reglas o secuencias de datos que necesitaremos posteriormente. Por último, en la fase de predicción, se realizan las predicciones teniendo en cuenta los sucesos actuales y las reglas o secuencias obtenidas en la fase anterior.

2.1. Fase de instalación

En esta primera fase procederemos a instalar la base de datos donde se almacenará todo aquello que sea necesario para la correcta ejecución del proyecto.

Empezaremos descargando e instalando un servidor de base de datos MySQL. Aconsejamos la descarga de:

http://www.apachefriends.org/es/xampp.html

Siguiendo los pasos de instalación obtenidos en la propia página resulta muy sencillo de configurar y ejecutar dicho servidor. A continuación pasamos a crear la base de datos. Para ello disponemos de una serie de archivos SQL incluidos en el CD del proyecto que debemos importar al servidor MySQL instalado. Dichos ficheros realizan las tareas que a continuación se describen:

- osm_create8.07.sql: Encargado de crear la base de datos Osmius con los permisos necesarios.
- osm_old_data_model.sql: Encargado de crear las tablas necesarias en la base de datos de Osmius. Estas tablas son una selección de todas las que integran Osmius, pero son las mínimas necesarias para el funcionamiento del módulo de predicciones.
- osm_final_data.sql: Permite instalar los datos de prueba en las tablas creadas por el archivo anterior. Estos datos han sido obtenidos de la ejecución de Osmius.
- osm_prediction_data_model.sql: Encargado de crear las tablas propias de éste módulo que permiten consultar los resultados de las predicciones.

Una vez hemos ejecutado estos ficheros, se podrá proseguir con las siguientes fases.

2.2. Fase de aprendizaje

En esta fase se realiza la extracción de los datos y se crean de las reglas o secuencias de datos que necesitaremos posteriormente.

2.2.1. Extracción de los datos

Para realizar la extracción de los datos que utilizaremos en el aprendizaje, disponemos de un primer producto: *osmius_data_transform*. El comando para su ejecución es:

./osmius_data_transform -t <tipo>-p <periodo>

-t: tipo de aprendizaje. Los tipos disponibles son: frequent y sequential.
-p: periodo sobre el que se va a realizar el aprendizaje. Los periodos disponibles son: hours, months, days, weeks, mornings y afternoons.

Los posibles valores del parámetro tipo de aprendizaje corresponden a Frequent Pattern Mining o Sequential Pattern Mining, que se explicarán detalladamente en la sección de algoritmos.

Como ejemplo, si quisiéramos realizar un aprendizaje de tipo Frequent Pattern Mining para las horas, es decir, obtener reglas sobre los eventos con las que poder hacer predicciones de lo que sucederá en las horas sucesivas, los parámetros serían: -t frequent -p hours.

Una vez ejecutado obtendremos como resultado un fichero .csv con los datos en el formato adecuado para realizar el aprendizaje.

2.2.2. Aprendizaje

Para realizar un aprendizaje sobre los datos extraídos, disponemos de dos nuevos productos: *itemset_ test* y *sequence_ test*. Los comandos para su ejecución son:

./itemset_test -i <fich_entrada>-s <soporte>-p ><fich_salida>
./sequence_test -i <fich_entrada>-s <soporte>-p ><fich_salida>

-i: indica el fichero .csv con los datos de entrada para el aprendizaje.
-s: soporte mínimo de las reglas o secuencias.
-p: indica que los resultados se registrarán en el fichero de salida indicado.

El primer ejecutable, itemset_test, corresponde al algoritmo Frequent Pattern Mining. Con él obtenemos como resultado del aprendizaje una serie de reglas de eventos con una probabilidad de que sean ciertas.

El segundo ejecutable, sequence_test, corresponde al algoritmo Sequential Pattern Mining. Con él obtenemos como resultado del aprendizaje unas secuencias de eventos con cierta probabilidad de que sucedan.

El soporte corresponde al número de apariciones que tiene un evento en una transacción. Se define una transacción como un conjunto de eventos que suceden durante un intervalo de tiempo definido. Dichos intervalos de tiempo son: horas, días, semanas, meses, mañanas y tardes.

2.2.3. Preparación del aprendizaje

Una vez hemos realizado el aprendizaje y tenemos el fichero de resultados del mismo, ya sean reglas o secuencias, hemos de proceder a las adecuación de dichos datos para su incorporación al servidor MySQL.

Para realizar éste proceso, disponemos de un otro producto llamado: Analizador.jar. El comando para su ejecución es:

java -jar Analizador.jar -m <modo>

-m: indica el modo de transformar los datos: sequence y frequent.

Esta tarea de la fase de aprendizaje es la única que no se ha implementado en C++ sino en Java, ya que éste lenguaje dispone de herramientas que permiten dividir el texto

de los ficheros de resultados de la tarea anterior y analizarlos para adecuarlos a la salida deseada. Como resultado obtenemos un fichero .sql con las inserciones de los datos obtenidos en las tablas adecuadas. Estas tablas irán condicionadas al modo que hayamos utilizado para ejecutar esta herramienta. En caso de haberlo lanzado con -m frequent, las tablas estarán ligadas al Frequent Pattern Mining, y en caso contrario irán enlazadas al algoritmo Sequential Pattern Mining.

2.2.4. Registro del Aprendizaje

Una vez realizado todos los procesos anteriores, lo único que queda es registrar el aprendizaje obtenido en la base de datos de Osmius.

Para realizar ésta tarea tenemos dos opciones. La primera es subir el archivo .sql, obtenido como resultado en el apartado anterior desde la interfaz gráfica que contiene MySQL, y la segunda sería ejecutar el siguiente comando:

mysql -u root -e use osmius; source <predicción.sql>;

Ejecutando éste sencillo comando, se almacenarán todos los datos obtenidos en el aprendizaje en sus respectivas tablas dentro de la base de datos de Osmius.

2.3. Fase de predicción

En esta fase se realizan las predicciones teniendo en cuenta los sucesos actuales y las reglas o secuencias obtenidas en la fase anterior.

El comando para su ejecución es:

./osmius_prediction -p <pred>[-all -l idn_learning -t time -s idn_service -i dti_ini -m long_max]

-p: define el tipo de algoritmo: frequent y sequential.
-all: se realiza la predicción con todos los aprendizajes que existan.
-l: aprendizaje especifico con el que se quiere hacer la predicción.
-t: periodo de tiempo que establece el momento actual. Por defecto es de una hora.
-s: si queremos que las instancias sean de un servicio en concreto.
-i: momento hasta el que queremos hacer la predicción. En caso de no poner ninguno sera hasta el momento actual.
-m: longitud máxima de la cadena en caso de ser una secuencia.

Los parámetros -all y -l, permiten indicar sobre qué aprendizaje vamos a realizar la predicción. Podemos haber realizado varios aprendizajes en nuestro sistema, por ejemplo uno sobre las horas y otro sobre los días, con lo que las reglas o secuencias obtenidas por cada uno de ellos serán distintas. Si queremos realizar las predicciones sobre todos los aprendizajes de que se disponga incluiremos el parámetro -all. Por el contrario, para hacerlo sobre un aprendizaje concreto utilizaremos -l. En este último caso habremos de indicarle el identificador de dicho aprendizaje.

Con el parámetro -i, definimos hasta que momento queremos realizar la predicción. Dicho parámetro es muy útil en caso de querer comprobar si nuestro sistema se ha comportado según las predicciones durante un periodo de tiempo. Podemos ejecutarlo para un mes y comprobar que durante ese mes ha hecho lo que se esperaba. Además, con el parámetro -t, podemos indicar al sistema qué significa *momento actual* para nosotros. El momento actual es el periodo entre la fecha que establezcamos con -i y tantas horas menos como indiquemos con -t. En caso de dejar vacíos ambos parámetros, el sistema tomará el periodo entre la fecha y hora actual y una hora menos.

Utilizaremos el parámetro -s para permitir realizar predicciones sobre las instancias de un mismo servicio.

El parámetro -m se utiliza para acotar la búsqueda que realiza la predicción sobre las secuencias del algoritmo de aprendizaje Sequential Pattern Mining, ya que reduce la longitud máxima de éstas a un número entero. En caso de no asignarle valor tomará todas las secuencias del aprendizaje.

Esta herramienta almacena directamente sus resultados en la base de datos de Osmius en las tablas adecuadas para cada tipo de aprendizaje sobre el que se ha realizado la predicción.

2.4. Interfaz gráfica

Para facilitar la utilización del módulo de predicciones de Osmius, se desarrollo un interfaz gráfico que permite ejecutar todas las fases del proyecto de una manera cómoda y amigable.

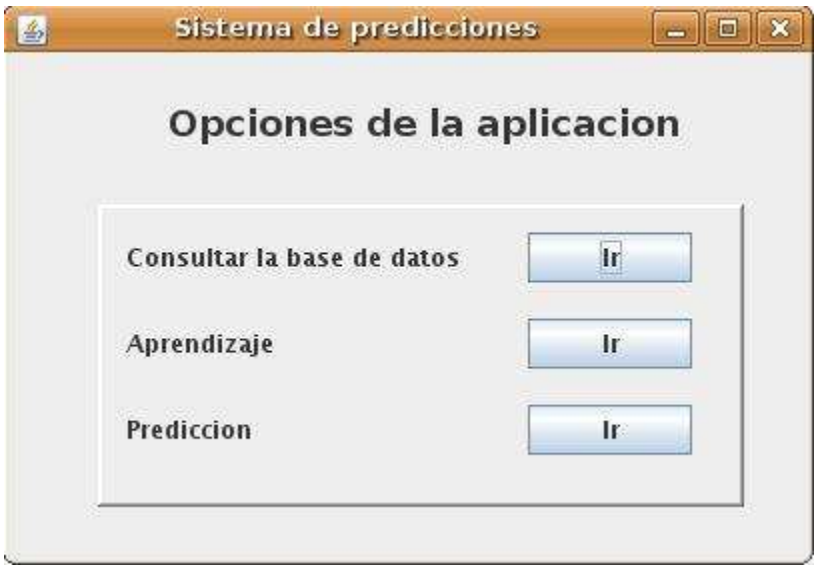
2.4.1. Instalación de la base de datos

El sistema detecta si es la primera vez que va a ser ejecutada esta tarea. En ese caso solicita permiso para llevar a cabo la instalación.

Este proceso crea la base de datos con los permisos necesarios, importa las tablas que necesitamos del modelo de datos de Osmius, crea las nuevas tablas y vistas necesarias para los procesos de aprendizaje y predicción y, por último, incluye una batería de datos para realizar las pruebas deseadas.

2.4.2. Menú principal y acceso a los datos

Desde la pantalla del menú principal podemos acceder a la fase de aprendizaje y de predicciones, ya sea para realizar dichos procesos o para ver sus respectivos resultados.



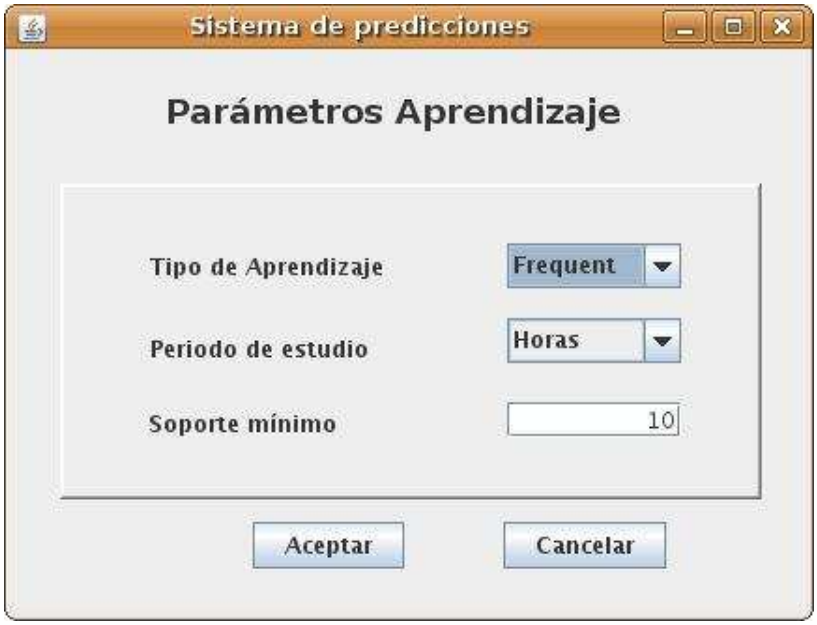
También permite consultar los datos contenidos en las tablas de Osmius, para orientarnos respecto a los parámetros más adecuados para llevar a cabo tanto el aprendizaje como las predicciones.

2.4.3. Realizar aprendizajes y ver los resultados

La opción de aprendizaje del menú principal, nos mostrará una pantalla que nos ofrece las distintas opciones disponibles.

Mediante la opción *Realizar aprendizaje* elegimos el tipo de aprendizaje (frequent o sequential), estableceremos las transacciones de agrupación (horas, días, semanas, meses, sólo mañanas o sólo tardes) y el soporte mínimo que han de tener las reglas o secuencias resultantes.

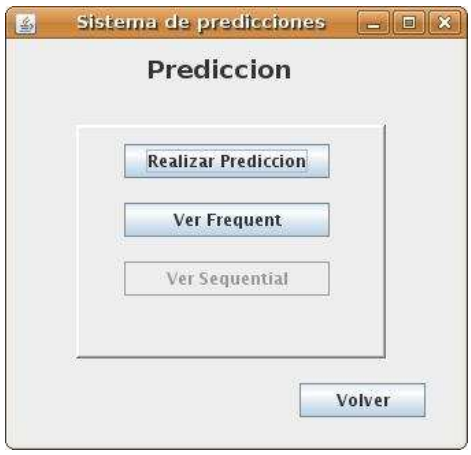
Las opciones *Ver Reglas* y *Ver Secuencias* se activarán en función del tipo de aprendizaje seleccionado, Frequent Pattern Mining o Sequential Pattern Mining respectivamente.



Una vez establecidos los parámetros se realizarán todas las tareas relacionadas con el proceso de aprendizaje. Dicho proceso puede tardar varios minutos en realizarse, pero el sistema nos avisa cuando haya finalizado. Esta fase podrá cancelarse es cualquier momento pulsando el botón correspondiente.

2.4.4. Realizar predicciones y ver los resultados

La opción de predicción del menú principal, nos ofrece las opciones disponibles.



Mediante la opción *Realizar predicción* elegimos el tipo de predicción (frequent o sequential). Las opciones *Ver Frequent* y *Ver Sequential* se activarán en función del tipo de predicción seleccionado, Frequent Pattern Mining o Sequential Pattern Mining respectivamente.

La opción *Realizar predicción* nos permite establecer el tipo de predicción, que dependerá de los tipos de aprendizaje que se hayan realizado en el sistema. Mediante el identificador del aprendizaje, se indicará al sistema que queremos realizar una predicción con todos los aprendizajes existentes o restringirnos solamente a uno de ellos.

A continuación, podremos indicar si queremos que los datos escogidos sean de un mismo servicio o que dicho servicio sea indiferente. Además, es posible fijar el momento actual del sistema a una fecha elegida. Esta opción es muy interesante para poder probar si el aprendizaje existente en el sistema es correcto, ya que se permite comprobar los resultados con los sucedidos en realidad.

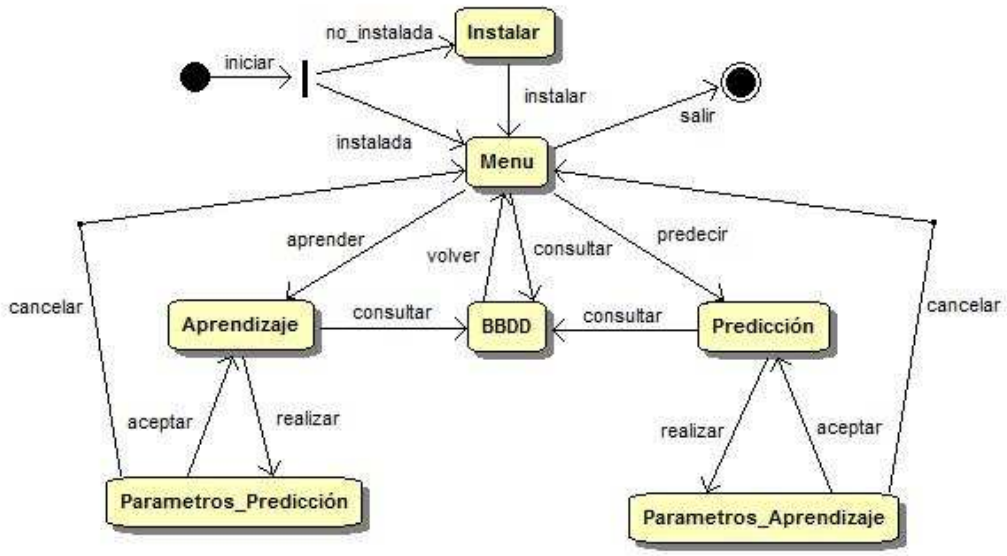
En caso de que el tipo de predicción sea sequential, podemos establecer una longitud máxima para las secuencias a utilizar.

Una vez pulsamos el botón de aceptar se realizarán todas las tareas relacionadas con el proceso de predicción. Dicho proceso puede tardar varios minutos en realizarse, pero el sistema nos avisa cuando ha finalizado. En caso de pulsar el botón de cancelar, volveremos al menú principal, ya que se entiende que no se quiere realizar ninguna predicción.

Una vez hemos realizado alguna predicción, se activarán los botones para poder ver las predicciones resultantes de dicho proceso.

2.4.5. Mapa de pantallas

Para poder tener una visión general del funcionamiento de la interfaz gráfica del sistema, se expone un mapa de pantallas, en el que se puede observar cómo se llega de una pantalla a la siguiente.



Capítulo 3

Algoritmos

La función básica de este módulo puede definirse de forma general como la predicción de los valores de criticidad y disponibilidad de instancias y servicios monitorizados que, con una probabilidad suficiente, tendrán lugar en un determinado instante futuro. Dicha predicción se divide en dos tipos:

- **Análisis de reglas de asociación de sucesos frecuentes**, conocido como Frequent Pattern Mining. Estas reglas contendrán información relativa a los valores de criticidad y disponibilidad para el conjunto de instancias monitorizadas.
- **Análisis de secuencias de asociación de sucesos temporales**, conocido como Sequential Pattern Mining. En estas secuencias se verán reflejados los valores de criticidad y disponibilidad para un conjunto de instancias monitorizadas. Si avanzamos del principio al final de la secuencia, podemos observar la relación temporal entre dichos sucesos.

Los beneficios de este análisis y sus resultados pueden resumirse en dos puntos:

- Nuestra base de conocimiento, basándonos en la cual realizaremos el diagnóstico y decidiremos la acción que se debe tomar, se compondrá, no sólo de los eventos recolectados, sino de eventos predichos en base a esas reglas o secuencias. Es decir, aumentaremos el conocimiento del estado del sistema, lo que permitirá realizar un diagnóstico más preciso.
- Adelantarse en el conocimiento de eventos futuros, obteniendo así una ventaja fundamental para predecir posibles fallos en el sistema monitorizado.

3.1. Frequent Pattern Mining

Se denominan patrones frecuentes a aquellos patrones que aparecen frecuentemente en un conjunto de datos. Encontrar estos patrones frecuentes es una de las claves para poder descubrir asociaciones, correlaciones y otras relaciones interesantes entre datos. Nuestro objetivo por lo tanto será descubrir estas relaciones frecuentes en un conjunto de datos determinado. A este proceso se le conoce con el nombre de Frequent Patter Mining [3].

Una regla de asociación es una regla que implica una cierta relación entre un conjunto de objetos que llamaremos items. Por ejemplo, que los dos ocurren simultáneamente o que la ocurrencia de uno implica al otro. El algoritmo Apriori [1] intenta descubrir este tipo de reglas que no pueden ser vistas de forma inmediata o no son fácilmente deducibles.

Una regla de asociación es una expresión de la forma: $X \rightarrow Y$ donde X e Y son conjuntos de items. La explicación intuitiva de una regla de este tipo es que si tenemos registrados los items del conjunto X, es muy posible que ocurran los items del conjunto Y.

Por ejemplo, el 40 % de las transacciones que contienen la compra de un ordenador también incluyen una impresora, o el 5 % de todas las transacciones contienen ambos productos. En este caso, tanto impresora como ordenador serían nuestros items. Al valor 40 % se le llama confianza de una regla, mientras que al 5 % se le conoce con el nombre de soporte de la regla.

Probabilísticamente, se pueden formular de la siguiente manera:

$$soporte(A \Rightarrow B) = P(AUB)$$

$$confianza(A \Rightarrow B) = P(\frac{B}{A})$$

El problema se transforma en encontrar todas las reglas de asociación que satisfacen un soporte y una confianza mínima, establecida de antemano. Se dice que estas reglas son reglas fuertes, en el sentido de frecuentes.

3.1.1. Algoritmo Apriori

Apriori fue desarrollado por el equipo de investigación de IBM Quest. En el algoritmo se utiliza el concepto de item, que son los distintos eventos ocurridos en el sistema, y los itemset, que son conjuntos de items. En particular, hablaremos de un k-itemset como un itemset que contiene k items. Por ejemplo, el itemset <ordenador, impresora>es un 2-itemset.

De forma análoga a como definimos el soporte de una regla, se define soporte de un itemset como la frecuencia de aparición de ese itemset, es decir, el número de transacciones que contienen ese itemset. En este caso nos encontramos ante un soporte absoluto. Si un itemset supera un valor umbral prefijado llamado soporte mínimo se denomina itemset frecuente. Básicamente el algoritmo se puede dividir en dos partes:

- Encontrar todas las combinaciones de items cuyo soporte supera un soporte mínimo prefijado. Es decir, descubrir todos aquellos itemsets frecuentes que aparezcan en nuestro conjunto de datos.

- Hacer uso de estos itemsets frecuentes para generar, por combinación, todas las reglas de asociación fuertes en las que intervengan estos itemsets. Para cada regla posible se debe calcular su confianza según la fórmula siguiente.

Por ejemplo, si los itemsets ABCD y AB son itemsets frecuentes, se tendrá una posible regla $AB \Rightarrow CD$, cuya confianza viene dada por:

$$confianza(AB \Rightarrow CD) = P(\frac{AB}{CD}) = \frac{soporte(ABCD)}{soporte(AB)}$$

La clave es que esta regla va a superar el soporte mínimo ya que ABCD es un itemset frecuente. Esta es la regla de búsqueda heurística que nos ayudará a no tener que explorar todo el posible árbol de estados.

Pseudocódigo

```
Procedimiento Apriori(L, minsup)
    L_1 := 1-itemset de L
    For (k:=2; L_{k-1} ≠ ∅; k++) do begin
        C_k=apriori_gen(L_{k-1}); // Nuevos candidatos
        For all transacciones t ∈ D do begin
            C_t=subset(C_k, t); //Candidatos contenidos en t
            For all candidatos c ∈ C_t do
                c.count++;
        end
        L_k:=c ∈ C_k | c.count >= minsup
    end
    Respuesta:=U_k L_k;
```

```
Funcion apriori_gen(L_{k-1})
    insert into C_k
    select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
    from L_{k-1} p, L_{k-1} k
    where p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1};
```

Donde D es la base de datos con un formato de <identificador transacción, item>, L_k es un conjunto de k-itemset, C_k es un conjunto de candidatos de k-itemset y Minsup es el soporte mínimo.

3.2. Sequential Pattern Mining

Las secuencias son un importante tipo de datos que frecuentemente se encuentran en la ciencia, la medicina, la seguridad, los negocios, etc. Por ejemplo, las secuencias de ADN codifican el mapa genético de todas las especies.

Al trabajar con el Sequential Pattern Mining [4] hay que tener en cuenta las siguientes consideraciones:

1. La relación relativa de orden entre los elementos de una secuencia es muy importante. En las secuencias, el hecho de que un elemento se encuentre a la izquierda de otro afecta al resto de elementos que se encuentren a su derecha. Además, la distancia entre dos elementos suele ser muy significativa.

2. Los patrones pueden ser subsecuencias. Algunas veces un patrón tiene que encontrarse en elementos consecutivos de una subcadena, mientras que en otras ocasiones, lo importante es que dicho patrón se encuentre en la cadena más general.

3. Es obvio que el instante de tiempo es un atributo importante en el conjunto de datos, y más aún en el proceso de minería de datos. Por ejemplo, una regla de asociación que no tenga en cuenta el tiempo podría ser "$ComprarA \rightarrow ComprarB$". Si tomásemos en cuenta el tiempo, la regla tendría un significado más preciso, como "*Comprar A implica Comprar B durante la semana*".

Un patrón de secuencias o simplemente una secuencia puede ser definida como una cadena de itemsets que ocurren frecuentemente en un orden especifico. Una secuencia si se denota como $< I_1; I_2; I_3; ::>$ donde el itemset I_1 ocurre antes que el itemset I_2, etc. Un itemset I_i en una secuencia es llamado transacción y debería tener un atributo de tiempo.

El Sequential Pattern Mining es un proceso de extracción de frecuencias de patrones de eventos ordenados. Si el número de secuencias es muy grande, los patrones de secuencias son más interesantes, aunque dicha longitud puede ser acotada por el usuario. Realmente, el algoritmo fija el soporte mínimo, de manera que el resultado sea más eficiente. A continuación se definen una serie de conceptos utilizados en este proceso:

- Una 1-secuencia es una secuencia con 1 itemset.

- Una subsecuencia a_i es una subcadena de otra secuencia s_i si $a_i \in s_i$.

- Una base de datos de secuencias SDB es un conjunto de tuplas de la forma (sid, s), donde sid es el identificador de la secuencia y s es la propia secuencia.

- Se dice que una tupla (sid, s) contiene a una secuencia t si t es subsecuencia de s. El número de tuplas de una base de datos que contienen la secuencias t se llama soporte de t, y se denota sup(t).

- Dado un soporte mínimo, una secuencia t es un patrón de secuencias si el soporte mínimo es sup(t).

El problema del Sequential Pattern Mining es encontrar un conjunto completo de los patrones de secuencias con respecto a un soporte mínimo.

3.2.1. SPADE (Sequential Pattern Discovery using Equivalent)

El algoritmo SPADE está basado en Apriori y se interpreta como una exploración de datos en formato vertical. Para conseguir dicho formato es necesario convertir los datos a un conjunto de tuplas, con la forma: <id_secuencia, id_evento>. Para un conjunto de datos dado, recorremos todos los identificadores de secuencias y los asociamos con los identificadores de eventos. Éstos últimos nos sirven como marcas de tiempo dentro de las secuencias.

La mayor ventaja de usar este formato es que podemos determinar el soporte de cualquier k-secuencia, siendo k el tamaño de la secuencia, con la simple unión entre las (k-1)-subsecuencias.

SPADE necesita un sólo escaneo de los datos para encontrar las 1-secuencias. Para obtener las 2-secuencias, uniremos todas las tuplas de las 1-secuencias, siempre que sean frecuentes, guardando el orden temporal de sus tuplas. Gracias a ésta propiedad, podemos saber que el primer evento de la secuencia ocurrió antes que el segundo y así sucesivamente. Dicho proceso lo repetiremos sucesivamente hasta que no se puedan formar más secuencias a partir de dichas uniones.

Pseudocódigo

```
Procedimiento SPADE(minsup, D)
        F_1:=1-secuencias de D
        F_2:=2-secuencias de D
        E:=Clases de equivalencia con relación θ_1
        Para todo S ∈ E hacer
                Enumerar_Secuencias_Frecuentes(S)
```

Donde minsup es el soporte mínimo, D son los datos y F_k es el conjunto de k-secuencias frecuentes.

```
Procedimiento Enumerar_Secuencias_Frecuentes(S)
    Para todo A_i ∈ S hacer
        Ti := ∅
        Para todo A_j ∈ S, con j >= i hacer
            R:= A_i U A_j
            Si (Filtro(R) = Falso) entonces
                L(R):=L(A_i)interseccionL(A_j)
                Si (Soporte(R) >= minsup) entonces
                    T_i := T_i U R; F_|R|:=F_|R| U R
        Fin_Para
        Si (Busqueda_primero_profundidad) entonces
            Enumerar_Secuencias_Frecuentes(T_i)
    Fin_Para
    Si (Busqueda_primero_anchura) entonces
        Para todo T_i ≠ ∅ hacer
            Enumerar_Secuencias_Frecuentes(T_i)
```

El siguiente procedimiento nos muestra si una secuencia pasa el filtro impuesto o no.

```
Procedimiento Filtro(R)
    Para todo (k-1)subsecuencia, A <R hacer
        Si ([A1] ha sido procesado y A ∈ F_{k-1} entonces devolver Verdad
    devolver Falso
```

3.3. Constraints Based Sequential Pattern Mining

Aunque los algoritmos propuestos son bastante eficientes, cuando tenemos un conjunto de datos muy grande, conservar dicha eficiencia es una tarea muy costosa. Si pudiéramos concentrarnos en los patrones que son realmente importantes para el usuario, el coste computacional disminuiría sustancialmente. Por estas razones de eficiencia, el uso de restricciones resulta esencial en muchas aplicaciones de minería de datos.

Éste algoritmo está basado en el Sequential Pattern Mining [2], al que se incluyen ciertas restricciones de tiempo, tamaño de las secuencias, tiempo entre los eventos de una secuencia,

etc. Con dichas restricciones se consiguen filtrar los patrones candidatos no prometedores en las primeras etapas del proceso.

Pseudocódigo

```
Procedimiento Constraint (D,C)
    Ct':=Restricción más débil que C
    F:=F1:=Items frecuentes en D que cumplen Ct'
    k:=2
    Repetir
        //Generar candidatos
        Usando Ct'y F generar Ck:= k-secuencias potencialmente frecuentes
        que cumplen Ct'
        //Filtrar candidatos
        Escanear D contando el soporte de las k-secuencias candidatas en Ck
        Fk:=Secuencias frecuentes en Ck
        F:=FUFk
        k:=k+1
    Hasta Terminar Condiciones (F,Ct')
    //Respetar la restricción inicial C
    Salen las secuencias de F que cumplen C
```

Donde D son los datos y C es el conjunto de las restricciones.

Capítulo 4

Modelo de datos

El módulo de predicción desarrollado nos permite predecir el estado futuro de las instancias sabiendo su estado actual. Dicha predicción se puede dar en periodos de tiempo de horas, días, semanas y meses. Es decir, dependiendo de las instancias no disponibles y críticas en un momento dado es capaz de avisar de las instancias que pueden dejar de estar disponibles o pasar a estado crítico dentro de un periodo de tiempo especificado.

Esta predicción se basa en reglas o secuencias de asociación de instancias caídas y críticas. Cada uno de los algoritmos de aprendizaje aplicados: Frequent Pattern Mining, Sequential Pattern Mining, Constraints Bassed Sequential Pattern Mining produce un conjunto de reglas o secuencias que guardamos en el modulo correspondiente a ese algoritmo. La predicción aplica cada uno de los algoritmos utilizando su módulos de predicción y obtiene unos resultados o predicciones que a su vez se guardan en ese mismo módulo para ser consultados por los usuarios.

A continuación se presenta el modelo de datos creado para el sistema de predicciones de Osmius. Este modelo de datos esta dividido en los distintos algoritmos de predicción utilizados, incluyendo una descripción de cada una de las tablas que intervienen en el mismo. Para que el modelo sea más sencillo de entender, en cada uno de los módulos funcionales que representan los distintos algoritmos se ha incluido una representación gráfica del modelo.

4.1. Nomenclatura

Se ha utilizado una nomenclatura estandarizada para todas las tablas y vistas usadas para facilitar la comprensión del modelo.

Tablas

- Nombre: OSM_XXXXX donde XXXXX será una pequeña descripción del contenido de esa tabla.
- Tipo de tabla: Todas las tablas son del tipo InnoDB, ya que este tipo respeta las claves ajenas y las transacciones de tipo commit/rollback.

Vistas

- Nombre: OSM_VW_XXXXX donde XXXXX será una pequeña descripción del contenido de esa vista.

Etiquetas

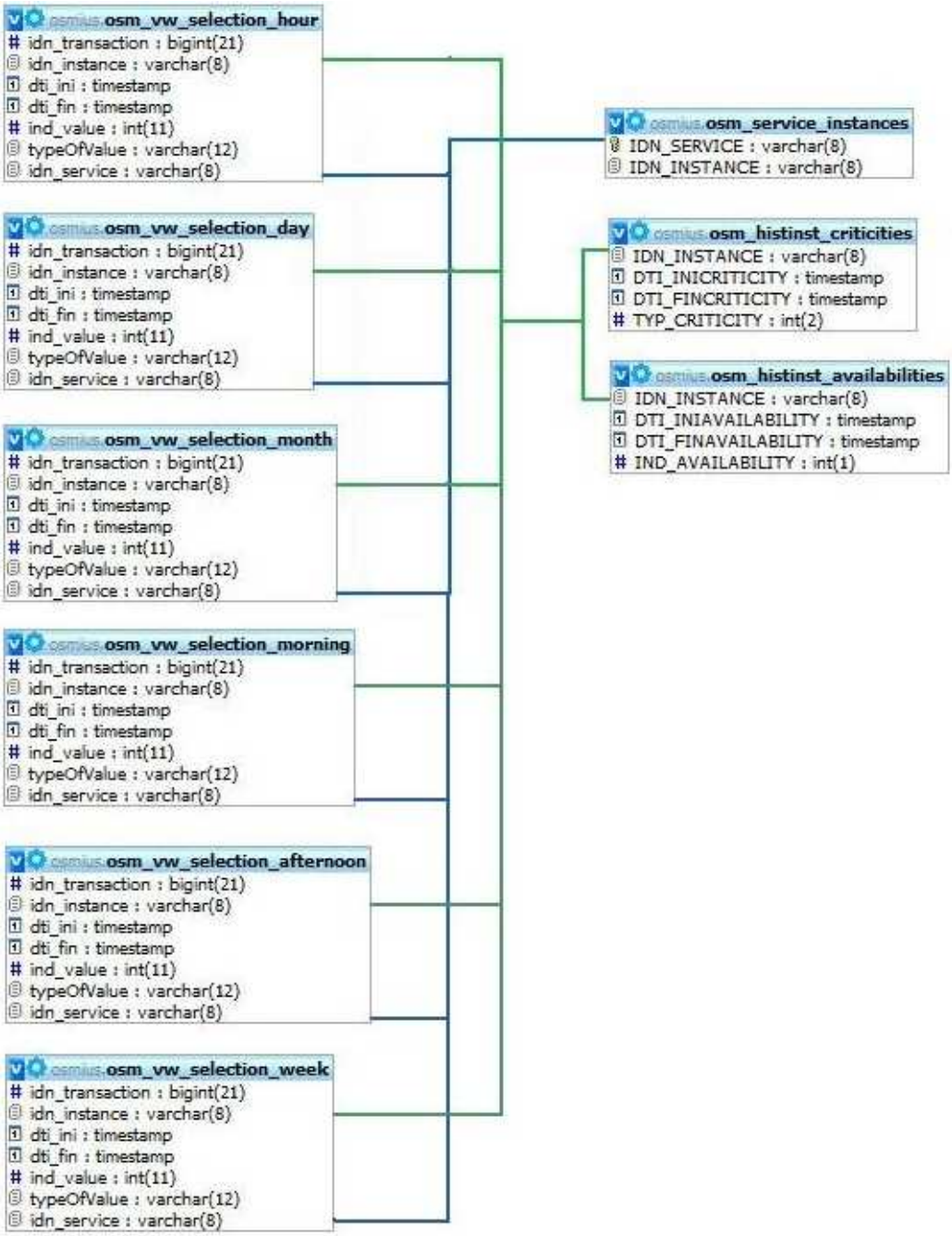
El nombre esta compuesto por tres letras que identifican el tipo de etiqueta y un nombre abreviado para el tipo de información que contiene el dato. Los posibles tipos de etiquetas son:

- IDN VARCHAR(8): Identificador, se usa en aquellas etiquetas que identifican un elemento principal en la tabla.
- TYP VARCHAR(8): Se usa para identificar las claves primarias en algunas tablas.
- DTI TIMESTAMP: Fechas.
- NUM INTEGER: Valores numéricos.
- IND INTEGER(1): Valores booleanos.
- DES VARCHAR(80): Descripciones largas.
- VAL VARCHAR(8): Valores de texto

4.2. Vistas de selección de datos

Las vistas de selección de datos componen una parte fundamental a la hora de realizar las predicciones. Estas vistas separan, de manera ordenada, los datos que vamos a utilizar en transacciones. Las transacciones son conjuntos de eventos que se asocian temporalmente. Para realizar dichas asociaciones temporales se ha creado una vista para cada periodo de asociación.

4.2.1. E/R Diagrama



4.2.2. Descripción de tablas y etiquetas

OSM_HISTINST_AVALILABILITIES					
Histórico sobre la disponibilidad de las instancias					
Field	Type	Null	Key	Default	Description
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INIAVAILABILITY	timestamp	NO			Fecha de comienzo de la disponibilidad descrita en la etiqueta ind_availability.
DTI_FINAVAILABILITY	timestamp	NO			Fecha de fin de la disponibilidad descrita en la etiqueta ind_availability.
IND_AVAILABILITY	int(1)	NO		1	Indicador de la disponibilidad: 0->No disponible 1->Disponible

OSM_HISTINST_CRITICITIES					
Histórico sobre el nivel de criticidad de las instancias					
Field	Type	Null	Key	Default	Description
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INICRITICITY	timestamp	NO			Fecha de comienzo del nivel de criticidad descrita en la etiqueta ind_criticity
DTI_FINCRITICIY	timestamp	NO			Fecha de fin del nivel de criticidad descrita en la etiqueta ind_criticity
TYP_CRITICITY	int(2)	NO		0	Nivel de criticidad del evento

OSM_SERVICES					
Servicios del sistema					
Field	Type	Null	Key	Default	Description
IDN_SERVICE	varchar(8)	NO	PRI		Identificador del servicio
DES_SERVICE	varchar(8)	NO			Descripción del servicio
TYP_CRITICITY	int(2)	NO		0	Nivel de criticidad del servicio
DTI_CRITICITY	timestamp	NO			Fecha de cambio del nivel de criticidad
IND_AVAILABILITY	int(2)	NO		1	Disponibilidad del servicio
DTI_AVAILABILITY	timestamp	NO			Fecha de cambio de la disponibilidad del servicio

OSM_INSTANCES					
Instancias del sistema susceptibles de ser monitorizadas					
Field	Type	Null	Key	Default	Description
IDN_INSTANCE	varchar(8)	NO	PRI		Identificador de la instancia
DES_INSTANCE	varchar(8)	NO			Descripción de la instancia
TYP_INSTANCE	varchar(8)	NO			Tipo de instancia
DTI_CREATED	timestamp	NO			Fecha de creación de la instancia
USR_CREATION	varchar(8)	NO			Usuario que creo la instancia
TYP_CRITICITY	int(2)	NO		0	Nivel de criticidad de la instancia
DTI_CRITICITY	timestamp	NO			Fecha de cambio del nivel de criticidad
IND_AVAILABILITY	int(1)	NO		1	Disponibilidad de la instancia
DTI_AVAILABILITY	timestamp	NO			Fecha de cambio de la disponibilidad
CONN_INFO	varchar(255)	SI			Texto de conexión usado por los agentes para conectarse con la instancia.
IDN_NODENAME	varchar(128)	SI			Nombre del nodo asociado a la instancia

OSM_SERVICE_INSTANCES					
Un servicio es un grupo de instancias					
Field	Type	Null	Key	Default	Description
IDN_SERVICE	varchar(8)	NO	PRI		Identificador de servicio
IDN_INSTANCE	varchar(8)	NO	PRI		Identificador de la Instancia

OSM_VW_SELECTION_HOUR					
Selección de los datos históricos separados en transacciones de una hora.					
Field	Type	Null	Key	Default	Description
IDN_TRANSACTION	varchar(8)	NO			Identificador de la transacción
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INI	timestamp	NO			Fecha de comienzo de la transacción.
DTI_FIN	timestamp	NO			Fecha de fin de la transacción.
IND_VALUE	int(2)	NO			Valor de la disponibilidad o criticidad de la instancia
TYP_VALUE	varchar(8)	NO			Tipo de valor: criticidad o disponibilidad
IDN_SERVICE	varchar(8)	NO			Servicio al que pertenece la instancia de la transacción

OSM_VW_SELECTION_MORNING					
Selección de los datos históricos separados en transacciones de 12 horas (solo de mañana).					
Field	**Type**	**Null**	**Key**	**Default**	**Description**
IDN_TRANSACTION	varchar(8)	NO			Identificador de la transacción
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INI	timestamp	NO			Fecha de comienzo de la transacción.
DTI_FIN	timestamp	NO			Fecha de fin de la transacción.
IND_VALUE	int(2)	NO			Valor de la disponibilidad o criticidad de la instancia
TYPE_VALUE	varchar(8)	NO			Tipo de valor: criticidad o disponibilidad
IDN_SERVICE	varchar(8)	NO			Servicio al que pertenece la instancia de la transacción

OSM_VW_SELECTION_AFTERNOON					
Selección de los datos históricos separados en transacciones de 12 horas (solo de tarde).					
Field	**Type**	**Null**	**Key**	**Default**	**Description**
IDN_TRANSACTION	varchar(8)	NO			Identificador de la transacción
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INI	timestamp	NO			Fecha de comienzo de la transacción.
DTI_FIN	timestamp	NO			Fecha de fin de la transacción.
IND_VALUE	int(2)	NO			Valor de la disponibilidad o criticidad de la instancia
TYPE_VALUE	varchar(8)	NO			Tipo de valor: criticidad o disponibilidad
IDN_SERVICE	varchar(8)	NO			Servicio al que pertenece la instancia de la transacción

OSM_VW_SELECTION_DAY					
Selección de los datos históricos separados en transacciones diarias.					
Field	**Type**	**Null**	**Key**	**Default**	**Description**
IDN_TRANSACTION	varchar(8)	NO			Identificador de la transacción
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INI	timestamp	NO			Fecha de comienzo de la transacción.
DTI_FIN	timestamp	NO			Fecha de fin de la transacción.
IND_VALUE	int(2)	NO			Valor de la disponibilidad o criticidad de la instancia
TYPE_VALUE	varchar(8)	NO			Tipo de valor: criticidad o disponibilidad
IDN_SERVICE	varchar(8)	NO			Servicio al que pertenece la instancia de la transacción

OSM_VW_SELECTION_WEEK					
Selección de los datos históricos separados en transacciones semanales.					
Field	Type	Null	Key	Default	Description
IDN_TRANSACTION	varchar(8)	NO			Identificador de la transacción
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INI	timestamp	NO			Fecha de comienzo de la transacción.
DTI_FIN	timestamp	NO			Fecha de fin de la transacción.
IND_VALUE	int(2)	NO			Valor de la disponibilidad o criticidad de la instancia
TYPE_VALUE	varchar(8)	NO			Tipo de valor: criticidad o disponibilidad
IDN_SERVICE	varchar(8)	NO			Servicio al que pertenece la instancia de la transacción

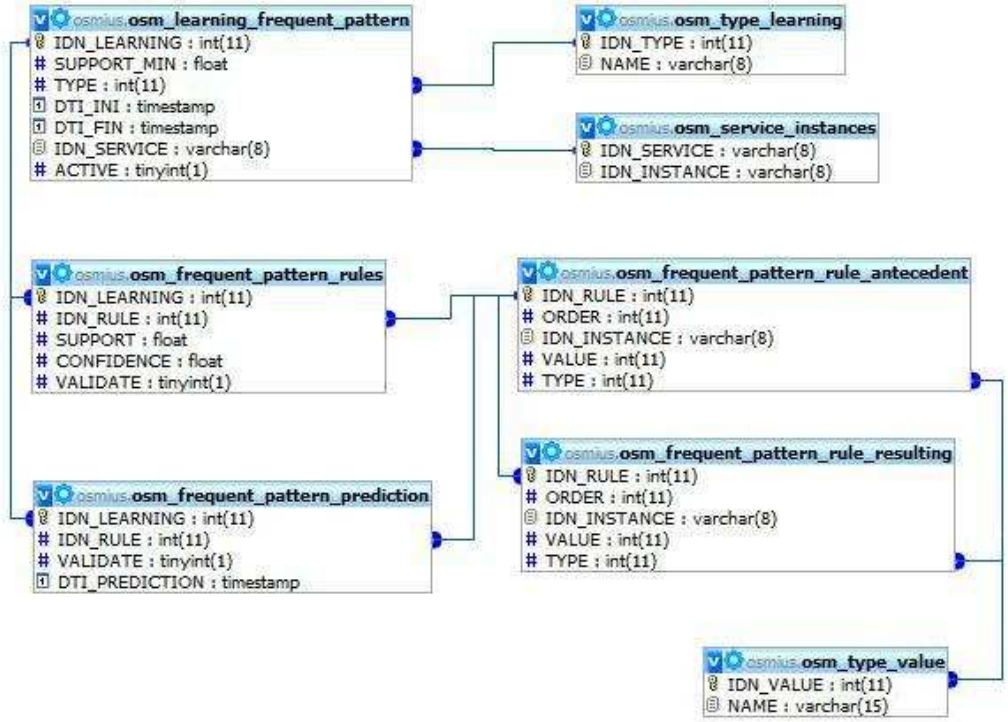
OSM_VW_SELECTION_MONTH					
Selección de los datos históricos separados en transacciones mensuales.					
Field	Type	Null	Key	Default	Description
IDN_TRANSACTION	varchar(8)	NO			Identificador de la transacción
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
DTI_INI	timestamp	NO			Fecha de comienzo de la transacción.
DTI_FIN	timestamp	NO			Fecha de fin de la transacción.
IND_VALUE	int(2)	NO			Valor de la disponibilidad o criticidad de la instancia
TYPE_VALUE	varchar(8)	NO			Tipo de valor: criticidad o disponibilidad
IDN_SERVICE	varchar(8)	NO			Servicio al que pertenece la instancia de la transacción

4.3. Tablas de frequent pattern mining

Estas tablas contienen toda la información relativa al algoritmo Frequent Pattern Mining. La tabla osm_learning_frequent_pattern ejerce de índice de los distintos aprendizajes realizados sobre los datos con éste algoritmo. Cada aprendizaje realizado tendrá asociados n reglas, almacenadas en osm_frequent_pattern_rules, y éstas a su vez serán ordenadas en antecedentes (osm_frequent_pattern_rule_antecedent) y consecuentes (osm_frequent_pattern_rule_resulting).

Por último, cada predicción realizada sobre éste conjunto de reglas se encuentra almacenada en la tabla osm_frequent_pattern_prediction.

4.3.1. E/R Diagrama



4.3.2. Descripción de tablas y etiquetas

OSM SERVICES					
Servicios del sistema					
Field	Type	Null	Key	Default	Description
IDN_SERVICE	varchar(8)	NO	PRI		Identificador del servicio
DES_SERVICE	varchar(8)	NO			Descripción del servicio
TYP_CRITICITY	int(2)	NO		0	Nivel de criticidad del servicio
DTI_CRITICITY	timestamp	NO			Fecha de cambio del nivel de criticidad
IND_AVAILABILITY	int(2)	NO		1	Disponibilidad del servicio
DTI_AVAILABILITY	timestamp	NO			Fecha de cambio de la disponibilidad del servicio

OSM TYPE LEARNING					
Lista los tipos de aprendizaje disponibles.					
Field	Type	Null	Key	Default	Description
IDN_TYPE	varchar(8)	NO	PRI		Identificador del aprendizaje
VAL_TYPE	varchar(8)	NO			Nombre del aprendizaje

OSM_LEARNING_FREQUENT_PATTERN					
Histórico de todos los aprendizajes lanzados con este algoritmo					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO		PRI	Identificador del aprendizaje
NUM_SUPPORT_MIN	int(11)	NO			Soporte mínimo del aprendizaje
IDN_TYPE	varchar(8)	NO			Tipo de aprendizaje.
DTI_INI	timestamp	NO			Fecha de inicio del aprendizaje.
DTI_FIN	timestamp	NO			Fecha de finalización del aprendizaje.
IDN_SERVICE	varchar(8)	SI		NULL	Restringimos el aprendizaje a las instancias pertenecientes a este servicio.
IND_ACTIVE	int(1)	SI		NULL	Marcamos si está o no disponible el aprendizaje.

OSM_FREQUENT_PATTERN_RULES					
Listado de reglas correspondientes a cada aprendizaje					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO	PRI		Identificador del aprendizaje
IDN_RULE	varchar(8)	NO	PRI		Identificador de la regla
NUM_SUPPORT	int(11)	NO			Soporte de la regla
NUM_CONFIDENCE	int(11)	NO			Confianza de la regla
IND_VALIDATE	int(1)	SI		NULL	Validación de si la regla es redundante o valida.

OSM_TYPE_VALUE					
Indica el tipo de valor de las reglas (si el valor es de criticidad o de disponibilidad)					
Field	Type	Null	Key	Default	Description
IDN_VALUE	varchar(8)	NO	PRI		Identificador del tipo de valor
VAL_VALUE	varchar(8)	NO			Nombre del tipo de valor

OSM_FREQUENT_PATTERN_RULE_ANTECEDENT					
Antecedente de las reglas					
Field	Type	Null	Key	Default	Description
IDN_RULE	varchar(8)	NO	PRI		Identificador de la regla
NUM_ORDER	int(11)	NO	PRI		Orden dentro del antecedente
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
NUM_VALUE	int(11)	NO			Valor
IDN_TYPE	varchar(8)	NO			Indica si el tipo de valor es criticidad o disponibilidad

OSM_FREQUENT_PATTERN_RULE_RESULTING					
Consecuente de las reglas					
Field	Type	Null	Key	Default	Description
IDN_RULE	varchar(8)	NO	PRI		Identificador de la regla
NUM_ORDER	int(11)	NO	PRI		Orden dentro del antecedente
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
NUM_VALUE	int(11)	NO			Valor
IDN_TYPE	varchar(8)	NO			Indica si el tipo de valor es criticidad o disponibilidad

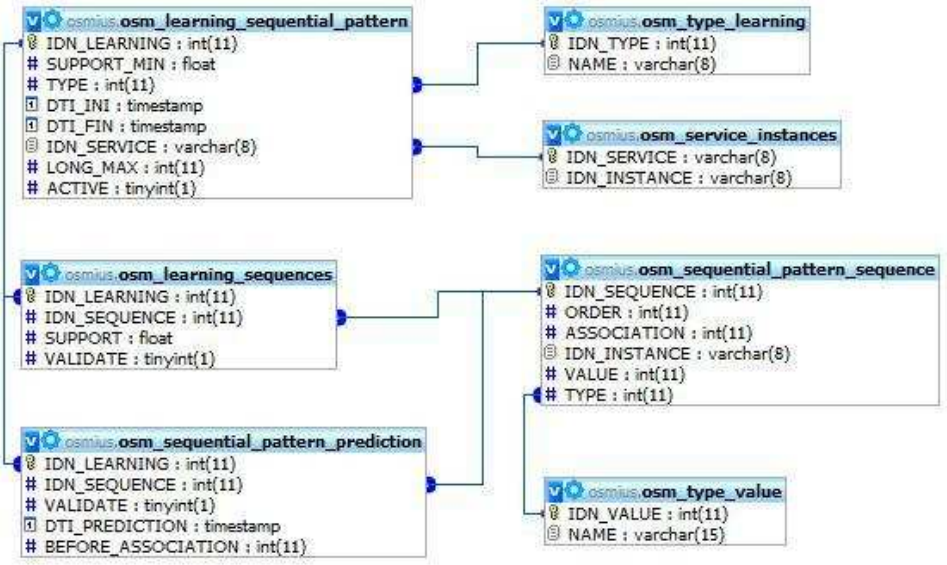
OSM FREQUENT PATTERN PREDICTION					
Resultado de las predicciones con los datos establecidos					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO	PRI		Identificador del aprendizaje
IDN_RULE	varchar(8)	NO	PRI		Identificador de la regla
IND_VALIDATE	int(1)	SI		NULL	Validación de si la predicción es valida
DTI_PREDICTION	timestamp	NO	PRI		Fecha en la que se lanza la predicción
NUM_HOURS	int(11)	NO		1	Horas antes de la predicción

4.4. Tablas de sequential pattern mining

Estas tablas contienen la información relativa al algoritmo Sequential Pattern Mining. La tabla osm_learning_sequential_pattern ejerce de índice de los distintos aprendizajes realizados sobre los datos con éste algoritmo. Cada aprendizaje realizado tienen asociadas n secuencias almacenadas en osm_learning_sequences, que a su vez están ordenadas en elementos de secuencia en la tabla osm_sequential_pattern_sequence.

Por último, esta almacenada cada predicción realizada sobre éste conjunto de secuencias en la tabla osm_sequencial_pattern_prediction.

4.4.1. E/R Diagrama



4.4.2. Descripción de tablas y etiquetas

OSM_SERVICES					
Servicios del sistema					
Field	Type	Null	Key	Default	Description
IDN_SERVICE	varchar(8)	NO	PRI		Identificador del servicio
DES_SERVICE	varchar(8)	NO			Descripción del servicio
TYP_CRITICITY	int(2)	NO		0	Nivel de criticidad del servicio
DTI_CRITICITY	timestamp	NO			Fecha de cambio del nivel de criticidad
IND_AVAILABILITY	int(2)	NO		1	Disponibilidad del servicio
DTI_AVAILABILITY	timestamp	NO			Fecha de cambio de la disponibilidad del servicio

OSM_TYPE_LEARNING					
Lista los tipos de aprendizaje disponibles.					
Field	Type	Null	Key	Default	Description
IDN_TYPE	varchar(8)	NO	PRI		Identificador del aprendizaje
VAL_TYPE	varchar(8)	NO			Nombre del aprendizaje

OSM_LEARNING_SEQUENTIAL_PATTERN					
Histórico de todos los aprendizajes lanzados con este algoritmo					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO		PRI	Identificador del aprendizaje
NUM_SUPPORT_MIN	int(11)	NO			Soporte mínimo del aprendizaje
IDN_TYPE	varchar(8)	NO			Tipo de aprendizaje.
DTI_INI	timestamp	NO			Fecha de inicio del aprendizaje.
DTI_FIN	timestamp	NO			Fecha de finalización del aprendizaje.
IDN_SERVICE	varchar(8)	SI		NULL	Restringimos el aprendizaje a las instancias pertenecientes a este servicio.
NUM_LONG_MAX	int(11)	SI		NULL	Longitud máxima de secuencia
IND_ACTIVE	int(1)	SI		NULL	Marcamos si está o no disponible el aprendizaje.

OSM_LEARNING_SEQUENCES					
Listado de secuencias correspondientes a cada aprendizaje					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO	PRI		Identificador del aprendizaje
IDN_SEQUENCE	varchar(8)	NO	PRI		Identificador de la secuencia
NUM_SUPPORT	int(11)	NO			Soporte de la secuencia
NUM_CONFIDENCE	int(11)	NO			Confianza de la secuencia
IND_VALIDATE	tinyint(1)	SI		NULL	Validación de si la secuencia es redundante o valida.

OSM_TYPE_VALUE					
Indica el tipo de valor de las secuencias (si el valor es de criticidad o de disponibilidad)					
Field	Type	Null	Key	Default	Description
IDN_VALUE	varchar(8)	NO	PRI		Identificador del tipo de valor
VAL_VALUE	varchar(8)	NO			Nombre del tipo de valor

OSM_SEQUENTIAL_PATTERN_SEQUENCE					
Listado de secuencias					
Field	Type	Null	Key	Default	Description
IDN_SEQUENCE	varchar(8)	NO	PRI		Identificador de la secuencia
NUM_ORDER	int(11)	NO	PRI		Orden dentro de la secuencia
NUM_ASSOCIATION	int(11)	NO	PRI		Asociación a la que pertenece la instancia dentro de la secuencia
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
NUM_VALUE	int(11)	NO			Valor
IDN_TYPE	varchar(8)	NO			Indica si el tipo de valor es criticidad o disponibilidad

OSM_SEQUENTIAL_PATTERN_PREDICTION					
Resultado de las predicciones con los datos establecidos					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO	PRI		Identificador del aprendizaje
IDN_SEQUENCE	varchar(8)	NO	PRI		Identificador de la secuencia
IND_VALIDATE	int(1)	SI		NULL	Validación de si la predicción es valida
DTI_PREDICTION	timestamp	NO	PRI		Fecha en la que se lanza la predicción
NUM_HOURS	int(11)	NO		1	Horas antes de la predicción
NUM_BEFORE_ASSOCIATION	int(11)	SI		NULL	Número de asociaciones que se quedan por detrás de la que representa el momento actual.

4.5. Tablas de constraint based sequential pattern mining

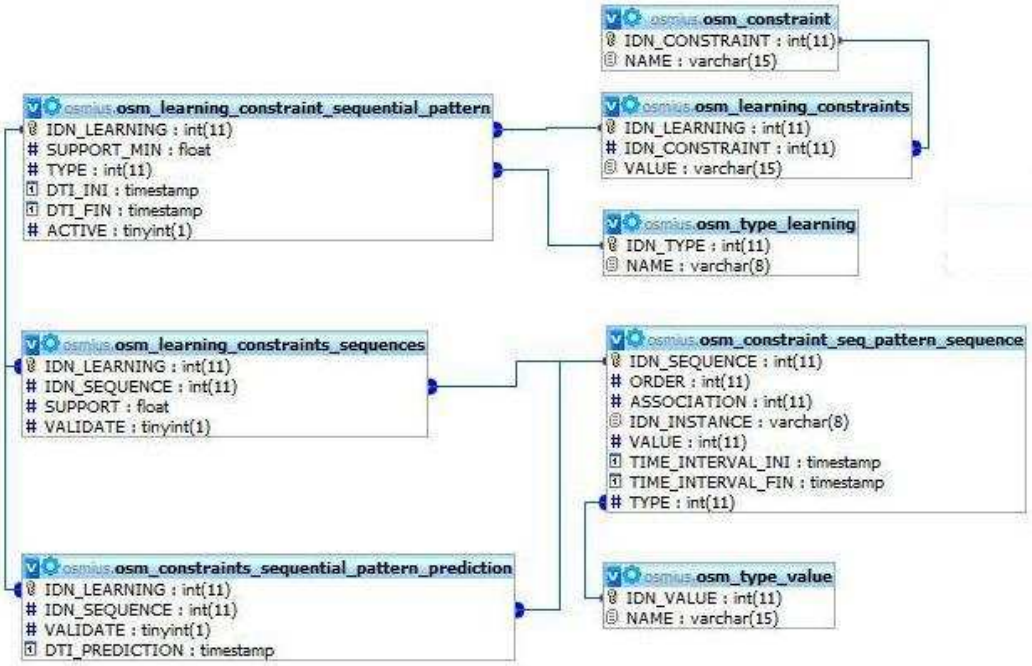
Estas tablas contienen todo la información relativa al algoritmo Constraint Based Sequential Pattern Mining.

La tabla osm_learning_constraint_sequential_pattern ejerce de índice de los distintos aprendizajes realizados sobre los datos con éste algoritmo.

Cada aprendizaje realizado tiene asociados n secuencias, que se encuentran almacenadas en consecutivamente en la tabla osm_learning_constraints_sequences, y éstas a su vez están ordenadas en elementos de secuencia en la tabla osm_constraint_sequential_pattern_sequence.

Por último, cada predicción realizada sobre éste conjunto de secuencias se almacenan en la tabla osm_constraints_sequencial_pattern_prediction.

4.5.1. E/R Diagrama



4.5.2. Descripción de tablas y etiquetas

OSM_TYPE_LEARNING					
Lista los tipos de aprendizaje disponibles.					
Field	Type	Null	Key	Default	Description
IDN_TYPE	varchar(8)	NO	PRI		Identificador del aprendizaje
VAL_TYPE	varchar(8)	NO			Nombre del aprendizaje

OSM_CONSTRAINT					
Lista de restricciones					
Field	Type	Null	Key	Default	Description
IDN_CONSTRAINT	varchar(8)	NO	PRI		Identificador de la restricción
VAL_CONSTRAINT	varchar(8)	NO			Nombre de la restricción

OSM_LEARNING_CONSTRAINTS					
Indica las restricciones con las que se ha realizado cada aprendizaje					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO	PRI		Identificador del aprendizaje
IDN_CONSTRAINT	varchar(8)	NO	PRI		Identificador de la restricción
NUM_VALUE	int(11)	NO			Valor de la restricción

OSM_LEARNING_CONSTRAINT_SEQUENTIAL_PATTERN					
Historico de todos los aprendizajes lanzados con este algoritmo					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO		PRI	Identificador del aprendizaje
NUM_SUPPORT_MIN	int(11)	NO			Soporte mínimo del aprendizaje
IDN_TYPE	varchar(8)	NO			Tipo de aprendizaje.
DTI_INI	timestamp	NO			Fecha de inicio del aprendizaje.
DTI_FIN	timestamp	NO			Fecha de finalización del aprendizaje.
IND_ACTIVE	int(1)	SI		NULL	Marcamos si está o no disponible el aprendizaje.

OSM_LEARNING_CONSTRAINTS_SEQUENCES					
Listado de secuencias correspondientes a cada aprendizaje					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO	PRI		Identificador del aprendizaje
IDN_SEQUENCE	varchar(8)	NO	PRI		Identificador de la secuencia
NUM_SUPPORT	int(11)	NO			Soporte de la secuencia
IND_VALIDATE	int(1)	SI		NULL	Validación de si la secuencia es redundante o valida.

OSM_TYPE_VALUE					
Indica el tipo de valor de las secuencias (si el valor es de criticidad o de disponibilidad)					
Field	Type	Null	Key	Default	Description
IDN_VALUE	varchar(8)	NO	PRI		Identificador del tipo de valor
VAL_VALUE	varchar(8)	NO			Nombre del tipo de valor

OSM_CONSTRAINT_SEQUENTIAL_PATTERN_SEQUENCE					
Listado de secuencias					
Field	Type	Null	Key	Default	Description
IDN_SEQUENCE	varchar(8)	NO	PRI		Identificador de la secuencia
NUM_ORDER	int(11)	NO	PRI		Orden dentro de la secuencia
NUM_ASSOCIATION	int(11)	NO	PRI		Asociación a la que pertenece la instancia dentro de la secuencia
IDN_INSTANCE	varchar(8)	NO			Identificador de la instancia
NUM_VALUE	int(11)	NO			Valor
TIME_INTERVAL_INI	timestamp	NO			Tiempo de comienzo de la asociación
TIME_INTERVAL_FIN	timestamp	NO			Tiempo de finalización de la asociación
IDN_TYPE	varchar(8)	NO			Indica si el tipo de valor es criticidad o disponibilidad

OSM_CONSTRAINTS_SEQUENTIAL_PATTERN_PREDICTION					
Resultado de las predicciones con los datos establecidos					
Field	Type	Null	Key	Default	Description
IDN_LEARNING	varchar(8)	NO	PRI		Identificador del aprendizaje
IDN_SEQUENCE	varchar(8)	NO	PRI		Identificador de la secuencia
VALIDATE	int(1)	SI		NULL	Validación de si la predicción es valida
DTI_PREDICTION	timestamp	NO	PRI		Fecha en la que se lanza la predicción
NUM_HOURS	int(11)	SI		1	Horas antes de la predicción

Capítulo 5

Estudio realizado

En éste capítulo se explica cómo se ha realizado el montaje y distribución del laboratorio para poder realizar las pruebas correspondientes al módulo de predicciones de Osmius.

A continuación se explican el tipo de pruebas realizadas y los resultados obtenidos por el sistema de predicciones. En ésta sección se compara la eficiencia de los algoritmos tomando como referencia el tiempo utilizado y el volumen de datos que pueden manejar sin dificultad.

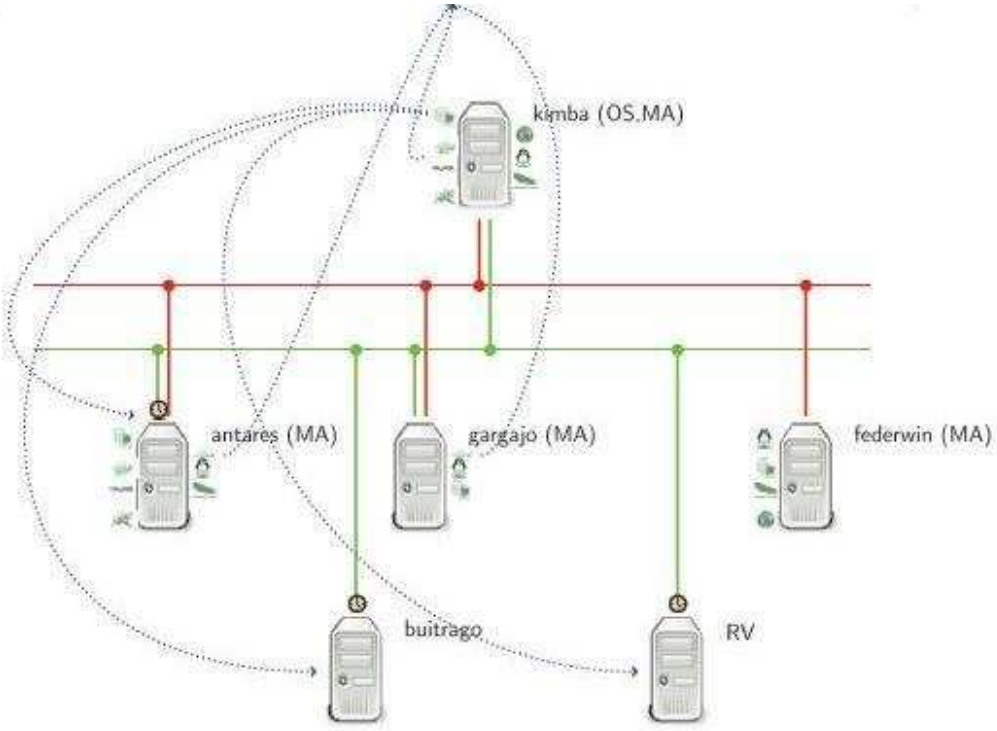
5.1. Distribución del laboratorio

El servidor central de Osmius fue instalado en una máquina llamada kimba en la que se ejecutaron las instancias usuales de Osmius y otras que se describirán más tarde.

También desplegamos tres agentes maestros en otras máquinas conectadas a la misma red local: antares, gargajo y federwin. Además de la red local, kimba es el servidor de una red OpenVPN (http://openvpn.net) que hemos usado para monitorizar máquinas remotas a partir de sus direcciones IP: RV, buitrago, gargajo y antares. El objetivo de utilizar esta red virtual fue poder simular errores en la red para observar la reacción de Osmius y el nuevo módulo.

5.1.1. Instancias Osmius

Además de las instancias propias de Osmius, en kimba se incluyeron otras instancias que dotan de mayor realismo al medio, y algunas que provocan errores controlados y correlacionados que son usados para evaluar el sistema de predicción.



- Instancias http: Proveen de mayor realismo al medio, dotándole de un alto número de eventos. Esta tarea consistió en monitorizar una serie de páginas web con un índice alto de visitas: la Universidad Complutense de Madrid (http://www.ucm.es), el periódico El País (http://www.elpais.com), el periódico El Mundo (http://www.elmundo.es) y la red social Facebook (http://www.facebook.com).

- Instancias IP: Una forma sencilla de generar errores controlados es la utilización de una red OpenVPN. En nuestro caso se desplegaron tres instancias IP en el agente maestro de kimba para monitorizar las direcciones virtuales de las máquinas RV, antares, gargajo y buitrago. Utilizando el cronómetro de Linux, se provoco la caída de las distintas máquinas en tiempos establecidos, para posteriormente poder analizar si los distintos algoritmos de predicción eran capaces de captar dichos patrones.

- Instancias de LOG: Simultáneamente se desplegaron otro tipo agentes para generar otros errores. Para ello se utilizaron agentes de tipo LOG, que son fácilmente programables. Dichos agentes provocaban errores controlados a intervalos de 1 hora, 12 horas, 24 horas, 1 semana y 1 mes. Se desarrollaron tres agentes LOG para cada intervalo de errores que se quiso generar. Dichos agentes leían constantemente un archivo y en caso de aparecer en éste su nombre (el nombre del agente), provocaba un fallo del tipo

indicado en dicho archivo.

- Intervalo de 1 hora: crit0101hNR, crit0201hNR y crit0301hNR. Corresponden a eventos de criticidad que se producen en un intervalo de 1 hora. También se crearon otras instancias similares a éstas, pero con las que se provocan fallos de disponibilidad y se expresan mediante avail0101hNR, avail0201hNR y avail0301hNR. Por último se desplegaron nuevas instancias en las que el intervalo entre ellas era aleatorio. Estas instancias se diferencian porque el sufijo es YR (yes random) en vez de NR (no random).
- Intervalos de 12 horas: Estas instancias son similares a las primeras, pero cambiando el periodo de 01h a 12h. Por tanto, el periodo de retraso entre ellas es más largo.
- Intervalos de 24 horas, 1 semana y 1 mes: Dichas instancias son como las previas, pero cambiando el identificador del periodo a 24h, 01w y 01m respectivamente.

Puesto que existen muchos más eventos para los periodos cortos que para los largos (tenemos 4 tipos para cada hora y solo 4 para un mes), podemos incluir nuevas instancias en caso de que los resultados obtenidos no sean relevantes.

5.1.2. Resultados obtenidos

En cuanto al aprendizaje realizado con Frequent Pattern Mining, se ha podido observar que las reglas obtenidas no son concluyentes. En el proceso de obtención de datos se han recolectado gran cantidad de eventos y resulta haber muchas reglas que son repetitivas o que no son completas con el conjunto de datos sobre el que se realiza el aprendizaje. Por tanto, usando éste algoritmo, se ve la necesidad de contar con un experto que analice las reglas conseguidas y elimine aquellas que sean irrelevantes.

En cuanto al tiempo que tarda en obtener las reglas, el algoritmo Apriori es significativamente más lento que el resto de los que componen éste módulo. Además, se ha observado que cuanto mayor es el volumen de datos en el que se buscan los patrones, el tiempo crece exponencialmente, consumiendo una gran cantidad de recursos hardware.

En favor de éste algoritmo, hay que indicar que si el volumen de datos sobre el que se quiere realizar la predicción es pequeño, éste algoritmo es el más rápido y los resultados obtenidos, al ser reglas con su antecedente y consecuente, son fácilmente interpretables.

El problema de tener gran cantidad de datos que analizar queda resuelto con el algoritmo, Sequential Pattern Mining. En éste caso, el tiempo en el que obtenemos las secuencias resul-

tantes del análisis de los datos es sustancialmente menor. Sin embargo, persiste el problema de que existen muchas secuencias que han de ser eliminadas en una fase posterior.

Según se ha explicado en el capítulo de algoritmos, lo primero que se hace es recoger todos los items y generar secuencias de un sólo elemento. A partir de éstas, iremos construyendo el resto de las secuencias. Esto quiere decir que en los resultados obtenidos se encontraran secuencias de un sólo elemento que no aporten información útil, y secuencias de más elementos, que sólo servirán para la construcción de secuencias mayores, por lo que tampoco nos valdrán como patrones de predicción relevantes.

Esta última desventaja queda parcialmente superada con el uso del algoritmo Constraint Based Sequential Pattern Mining. Con éste algoritmo podemos indicar una serie de restricciones previas a su ejecución, con lo que el filtrado posterior lo podemos prácticamente omitir. No obstante su uso requiere tener cierto conocimiento del conjunto de datos y del proceso de aprendizaje para saber que tipo de restricciones son las más adecuadas para obtener las secuencias más precisas posibles.

Desde el punto de vista de consumo de tiempo en relación con el volumen de datos, éste es el algoritmo más eficiente de los mostrados. Esto es debido a que se restringe el conjunto de datos sobre el que se realiza el aprendizaje antes de empezar con el propio proceso. De ésta manera, aunque tengamos un gran volumen de datos, se seleccionarán solamente los que a nosotros nos interesan para realizar la búsqueda de patrones.

Aunque teóricamente éste algoritmo parece el más eficiente, todas las implementaciones probadas son muy antiguas, con lo que sería necesario reestructurar dicho algoritmo para el hardware actual. Además, el formato de datos necesario para la ejecución del mismo es demasiado restrictivo para las ventajas que se obtienen, puesto que las restricciones que se pueden imponer, son fácilmente simuladas por una simple consulta previa de los datos en lenguaje SQL.

Conclusiones

En este proyecto se ha dotado con un módulo de predicciones al sistema de monitorización Osmius, desarrollado por PeopleWare bajo licencia GPL.

El objetivo de dicho módulo ha sido poder predecir, por medio del conocimiento pasado, los posibles acontecimientos futuros perjudiciales con el fin de poder evitarlos. Éste herramienta ha dotado a Osmius de gran potencia e incrementa su valor como software de monitorización.

En el proceso de desarrollo del proyecto, se han implementado una serie de algoritmos de aprendizaje con los que se pueden realizar las predicciones y una base de datos consistente en la que se han podido almacenar dichos datos. Además, todos estos algoritmos están fuertemente parametrizados para dar al usuario gran libertad de acción a la hora de realizar dichas predicciones.

En cuanto a los algoritmos estudiados para realizar dicho análisis, se observo que era necesario el conocimiento de un experto que realizara un filtrado de las reglas y secuencias resultantes de la ejecución de los mismos.

Además, podemos concluir que a mayor número de datos, el algoritmo Frequent Pattern Mining funciona sustancialmente peor que su homólogo, el algoritmo Sequential Pattern Mining.

También fue relevante la comprobación de que según se modificaban los datos se conseguían mejores resultados, por eso se pensó en utilizar el algoritmo Constraint Based Sequential Pattern Mining. La razón por la que nos inclinamos a usarlo fue que se podían definir y utilizar una serie de restricciones que limitaban los datos a utilizar.

Aunque teóricamente éste algoritmo parecía el más eficiente, todas las implementaciones probadas eran muy antiguas, con lo que sería necesario reestructurar dicho algoritmo para el hardware actual. Además, el formato de datos necesario para la ejecución del mismo es demasiado restrictivo en comparación con los beneficios que se obtienen, puesto que las restricciones que se pueden imponer son fácilmente simuladas por una simple consulta previa

de los datos en lenguaje SQL. Por tanto, se decidió no usar y optar por dicha alternativa.

Éste proyecto admite una serie de ampliaciones que sería conveniente llevar a cabo. Lo primero sería adaptarlo de manera que se pudiese realizar una integración total con la consola de Osmius. Dicha tarea se ha de realizar por medio de tecnologías J2EE, ya que es una consola Web multi-plataforma con la que se puede acceder virtualmente a cualquier funcionalidad del sistema.

Otra posible ampliación de gran interés para el sistema y el usuario sería la predicción de los SLA, mediante la nota global del sistema, ya que permitirían evitar muchos posibles problemas de acuerdos.

Para ésta última tarea sería muy beneficiosa la utilización de redes neuronales, puesto que son capaces de crear rectas de regresión con las que realizar las predicciones y, además, son capaces de aprender de ellas de forma automatizada.

Bibliografía

[1] R. Agrawal, T. Imielinski, and AN. Swami. Mining association rules between sets of items in large databases. *SIGMOD*, 22(2):207–16, 1993.

[2] M. Garofalakis, N. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 223–224, 1999.

[3] J. Han, J. Pei, Y. Yin, and M. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. pages 318–362, 1986.

Apéndice A

Creación de la base de datos

En este anexo se verán todos los pasos necesarios para la creación de la base de datos. Se ha subdividido en tres apartados distintos, uno por cada archivo que será necesario importar a nuestro gestor de bases de datos. Dichos archivos responden a la creación de la base de datos, de las tablas necesarias existentes en Osmius y las nuevas tablas que ha sido necesario crear para llevar a cabo el proceso de aprendizaje y predicción.

Archivo osm_create8.07.sql

El archivo que se muestra a continuación crea la base de datos con las restricciones y usuarios necesarios.

```
CREATE DATABASE osmius;
CREATE USER 'osmius'@'localhost';
GRANT ALL ON osmius.* to 'osmius'@'localhost';
GRANT ALL ON mysql.* to 'osmius'@'localhost';
GRANT CREATE VIEW, SHOW VIEW ON osmius.* to 'osmius'@'localhost';
GRANT SUPER ON *.* TO 'osmius'@'localhost';
set password for 'osmius'@'localhost' = PASSWORD('osmius');
CREATE USER 'osmius';
GRANT ALL ON osmius.* to 'osmius';
GRANT ALL ON mysql.* to 'osmius';
GRANT CREATE VIEW, SHOW VIEW ON osmius.* to 'osmius';
GRANT SUPER ON *.* TO osmius;
set password for 'osmius' = PASSWORD('osmius');
```

Archivo osm_old_data_model.sql

Este archivo contiene las tablas necesarias de pre-existentes en Osmius que necesitaremos para recabar los datos del proceso de monitorización.

Servicios

Contiene los servicios existentes en la base de datos.

```
CREATE TABLE ‘OSM_SERVICES‘ (
‘IDN_SERVICE‘ varchar(8) NOT NULL COMMENT ’Service identifier’,
‘DES_SERVICE‘ varchar(80) NOT NULL COMMENT ’Description of the service’,
‘TYP_CRITICITY‘ int(2) NOT NULL default ’0’
COMMENT ’Critical level of the service’,
‘DTI_CRITICITY‘ timestamp NOT NULL default ’0000-00-00 00:00:00’
COMMENT ’Date of change of the critical level’,
‘IND_AVAILABILITY‘ int(1) NOT NULL default ’1’
COMMENT ’Availability of the service’,
‘IND_AVAILABILITY‘ timestamp NOT NULL default ’0000-00-00 00:00:00’
COMMENT ’Date of change of availability of the service’,
PRIMARY KEY (‘IDN_SERVICE‘),
KEY ‘TYP_CRITICITY‘ (‘TYP_CRITICITY‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Services of the system’;
```

Instancias

Contiene la información sobre las instancias que tiene la base de datos.

```
CREATE TABLE ‘OSM_INSTANCES‘ (
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Identifier of the instance’,
‘DES_INSTANCE‘ varchar(80) NOT NULL COMMENT ’Description of the instance’,
‘TYP_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Instance type’,
‘DTI_CREATED‘ timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP COMMENT ’Data of creation of the instance’,
‘USR_CREATION‘ varchar(8) NOT NULL COMMENT ’User who created the instance’,
```

```
‘TYP_CRITICITY‘ int(2) NOT NULL default ’0’ COMMENT ’Critical level of
the instance’,
‘DTI_CRITICITY‘ timestamp NOT NULL default ’0000-00-00 00:00:00’
COMMENT ’Date of change of the critical leve of the instance’,
‘IND_AVAILABILITY‘ int(1) NOT NULL default ’1’ COMMENT ’Indicator of
the availability of the instance’,
‘DTI_AVAILABILITY‘ timestamp NOT NULL default ’0000-00-00 00:00:00’
COMMENT ’Date of change of the availability of the instance’,
‘CONN_INFO‘ varchar(255) default NULL COMMENT ’Connection string used
by the agent to connect to the instance’,
‘IDN_NODENAME‘ varchar(128) default NULL COMMENT ’Nodename associate to
the instance’,
PRIMARY KEY (‘IDN_INSTANCE‘),
KEY ‘TYP_INSTANCE‘ (‘TYP_INSTANCE‘),
KEY ‘TYP_CRITICITY‘ (‘TYP_CRITICITY‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Instances in the system
susceptible to be monitoring’;
```

Servicios con instancias

En esta tabla vemos que instancias pertenecen a cada servicio.

```
CREATE TABLE ‘OSM_SERVICE_INSTANCES‘ (
‘IDN_SERVICE‘ varchar(8) NOT NULL COMMENT ’Service identifier’,
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Instance identifier’,
PRIMARY KEY (‘IDN_SERVICE‘,‘IDN_INSTANCE‘),
KEY ‘osm_services_instances_ibfk_2‘ (‘IDN_INSTANCE‘),
KEY ‘osm_services_instances_ibfk_1‘ (‘IDN_SERVICE‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’A service is a
group of instances’;
```

Histórico de disponibilidad

En la siguiente tabla podremos consultar todos los eventos relacionados con la disponibilidad de las instancias desde que se ejecuta Osmius en nuestro sistema.

```
CREATE TABLE ‘OSM_HISTINST_AVAILABILITIES‘ (
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Identifier of
the instance’,
‘DTI_INIAVAILABILITY‘ timestamp NOT NULL
default ’0000-00-00 00:00:00’
COMMENT ’Date of start of availability described in ind_availability field’,
‘DTI_FINAVAILABILITY‘ timestamp NOT NULL
default ’0000-00-00 00:00:00’
COMMENT ’Date of end of availability described in ind_availability field’,
‘IND_AVAILABILITY‘ int(1) NOT NULL
default ’1’ COMMENT ’Indicator of
avalabilitiy: 0->Not available 1->Available’,
KEY ‘IDN_INSTANCE‘ (‘IDN_INSTANCE‘,‘DTI_INIAVAILABILITY‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 PACK_KEYS=1
ROW_FORMAT=FIXED
COMMENT=’Historic of the availability of the instances’;
```

Histórico de criticidad

En la siguiente tabla podremos consultar todos los eventos relacionados con la criticidad de las instancias desde que se ejecuta Osmius en nuestro sistema.

```
CREATE TABLE ‘OSM_HISTINST_CRITICITIES‘ (
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Identifier of the instance’,
‘DTI_INICRITICITY‘ timestamp NOT NULL
default ’0000-00-00 00:00:00’
COMMENT ’Date of start of critical level described in ind_criticity field’,
‘DTI_FINCRITICITY‘ timestamp NOT NULL
default ’0000-00-00 00:00:00’
COMMENT ’Day of end of critical level described in idn_criticiy field’,
‘TYP_CRITICITY‘ int(2) NOT NULL default ’0’ COMMENT ’Critical level of
the event’,
KEY ‘TYP_CRITICITY‘ (‘TYP_CRITICITY‘),
KEY ‘IDN_INSTANCE‘ (‘IDN_INSTANCE‘,‘DTI_INICRITICITY‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 PACK_KEYS=1
ROW_FORMAT=FIXED
COMMENT=’Historic of the critical level of the instances’;
```

Archivo osm_prediction_data_model.sql

En este archivo podemos encontrar todas aquellas tablas y vistas que ha sido necesario crear para el funcionamiento de módulo de predicciones.

Vistas de selección

Las siguientes vistas seleccionan sólo los datos que son de nuestro interés entre todos los existentes en la base de datos. Además, dichas vistas agrupan en transacciones los datos en las diferentes franjas horarias marcadas.

HORAS

```
DROP VIEW IF EXISTS `OSM_VW_SELECTION_HOUR`;
CREATE OR REPLACE VIEW `OSM_VW_SELECTION_HOUR` AS
SELECT TIMESTAMPDIFF(HOUR,
`OSM_HISTINST_AVAILABILITIES`.`DTI_INIAVAILABILITY`
,CURRENT_TIMESTAMP) AS `idn_transaction`,
`OSM_HISTINST_AVAILABILITIES`.`IDN_INSTANCE` AS `idn_instance`,
`OSM_HISTINST_AVAILABILITIES`.`DTI_INIAVAILABILITY` AS `dti_ini`,
`OSM_HISTINST_AVAILABILITIES`.`DTI_FINAVAILABILITY` AS `dti_fin`,
`OSM_HISTINST_AVAILABILITIES`.`IND_AVAILABILITY` AS `ind_value`,
'AVAILABILITY' AS `typ_value`,
`OSM_SERVICE_INSTANCES`.`IDN_SERVICE` AS `idn_service`
FROM `OSM_HISTINST_AVAILABILITIES` JOIN `OSM_SERVICE_INSTANCES`
WHERE (`OSM_HISTINST_AVAILABILITIES`.`IDN_INSTANCE` =
`OSM_SERVICE_INSTANCES`.`IDN_INSTANCE`) AND
(`OSM_HISTINST_AVAILABILITIES`.`IND_AVAILABILITY` = 0)
UNION ALL
SELECT TIMESTAMPDIFF(HOUR,
`OSM_HISTINST_CRITICITIES`.`DTI_INICRITICITY`,
CURRENT_TIMESTAMP) AS `idn_transaction`,
`OSM_HISTINST_CRITICITIES`.`IDN_INSTANCE` AS `idn_instance`,
`OSM_HISTINST_CRITICITIES`.`DTI_INICRITICITY` AS `dti_ini`,
`OSM_HISTINST_CRITICITIES`.`DTI_FINCRITICITY` AS `dti_fin`,
`OSM_HISTINST_CRITICITIES`.`TYP_CRITICITY` AS `ind_value`,
'CRITICITY' AS `typ_value`,
`OSM_SERVICE_INSTANCES`.`IDN_SERVICE` AS `idn_service`
```

```
FROM ‘OSM_HISTINST_CRITICITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ = 2)
ORDER BY ‘idn_transaction‘;
```

MAÑANAS

```
DROP VIEW IF EXISTS ‘OSM_VW_SELECTION_MORNING‘;
CREATE OR REPLACE VIEW ‘OSM_VW_SELECTION_MORNING‘ AS
SELECT TIMESTAMPDIFF(DAY,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘,
CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_FINAVAILABILITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ AS ‘ind_value‘,
’AVAILABILITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_AVAILABILITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ = 0)
AND (HOUR(TIME
(‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘)) <’12’)
UNION ALL
SELECT TIMESTAMPDIFF(DAY,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘,
CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_FINCRITICITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ AS ‘ind_value‘,
’CRITICITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_CRITICITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
```

```
WHERE (‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ = 2)
AND (HOUR(TIME(‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘)) <’12’)
ORDER BY ‘idn_transaction‘;
```

TARDES

```
DROP VIEW IF EXISTS ‘OSM_VW_SELECTION_AFTERNOON‘;
CREATE OR REPLACE VIEW ‘OSM_VW_SELECTION_AFTERNOON‘ AS
SELECT TIMESTAMPDIFF(DAY,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘,
CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_FINAVAILABILITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ AS ‘ind_value‘,
’AVAILABILITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_AVAILABILITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ = 0)
AND (HOUR(TIME
(‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘)) >= ’12’)
UNION ALL
SELECT TIMESTAMPDIFF(DAY,‘OSM_HISTINST_CRITICITIES‘.
‘DTI_INICRITICITY‘,CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_FINCRITICITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ AS ‘ind_value‘,
’CRITICITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_CRITICITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ = 2)
AND (HOUR(TIME(‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘)) >= ’12’)
ORDER BY ‘idn_transaction‘;
```

SEMANAS

```
DROP VIEW IF EXISTS ‘OSM_VW_SELECTION_WEEK‘;
CREATE OR REPLACE VIEW ‘OSM_VW_SELECTION_WEEK‘ AS
SELECT TIMESTAMPDIFF(WEEK,‘OSM_HISTINST_AVAILABILITIES‘.
‘DTI_INIAVAILABILITY‘, CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_FINAVAILABILITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ AS ‘ind_value‘,
’AVAILABILITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_AVAILABILITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ = 0)
UNION ALL
SELECT TIMESTAMPDIFF(WEEK,‘OSM_HISTINST_CRITICITIES‘.
‘DTI_INICRITICITY‘, CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_FINCRITICITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ AS ‘ind_value‘,
’CRITICITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_CRITICITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ = 2)
ORDER BY ‘idn_transaction‘;
```

DIAS

```
DROP VIEW IF EXISTS ‘OSM_VW_SELECTION_DAY‘;
CREATE OR REPLACE VIEW ‘OSM_VW_SELECTION_DAY‘ AS
SELECT TIMESTAMPDIFF(DAY,‘OSM_HISTINST_AVAILABILITIES‘.
‘DTI_INIAVAILABILITY‘, CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_FINAVAILABILITY‘ AS ‘dti_fin‘,
```

```
‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ AS ‘ind_value‘,
’AVAILABILITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_AVAILABILITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ = 0)
UNION ALL
SELECT TIMESTAMPDIFF(DAY,‘OSM_HISTINST_CRITICITIES‘.
‘DTI_INICRITICITY‘, CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_FINCRITICITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ AS ‘ind_value‘,
’CRITICITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_CRITICITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ = 2)
ORDER BY ‘idn_transaction‘;
```

MESES

```
DROP VIEW IF EXISTS ‘OSM_VW_SELECTION_MONTH‘;
CREATE OR REPLACE VIEW ‘OSM_VW_SELECTION_MONTH‘ AS
SELECT TIMESTAMPDIFF(MONTH,‘OSM_HISTINST_AVAILABILITIES‘
.‘DTI_INIAVAILABILITY‘, CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_INIAVAILABILITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘DTI_FINAVAILABILITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ AS ‘ind_value‘,
’AVAILABILITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_AVAILABILITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_AVAILABILITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_AVAILABILITIES‘.‘IND_AVAILABILITY‘ = 0)
```

```
UNION ALL
SELECT TIMESTAMPDIFF(MONTH,‘OSM_HISTINST_CRITICITIES‘.
‘DTI_INICRITICITY‘,CURRENT_TIMESTAMP) AS ‘idn_transaction‘,
‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ AS ‘idn_instance‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_INICRITICITY‘ AS ‘dti_ini‘,
‘OSM_HISTINST_CRITICITIES‘.‘DTI_FINCRITICITY‘ AS ‘dti_fin‘,
‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ AS ‘ind_value‘,
’CRITICITY’ AS ‘typ_value‘,
‘OSM_SERVICE_INSTANCES‘.‘IDN_SERVICE‘ AS ‘idn_service‘
FROM ‘OSM_HISTINST_CRITICITIES‘ JOIN ‘OSM_SERVICE_INSTANCES‘
WHERE (‘OSM_HISTINST_CRITICITIES‘.‘IDN_INSTANCE‘ =
‘OSM_SERVICE_INSTANCES‘.‘IDN_INSTANCE‘)
AND (‘OSM_HISTINST_CRITICITIES‘.‘TYP_CRITICITY‘ = 2)
ORDER BY ‘idn_transaction‘;
```

Frequent Pattern Mining

En las siguientes tablas encontraremos todo lo necesario para realizar la persistencia de datos al utilizar el algoritmo frequent pattern mining.

Tipo de aprendizaje

Se almacenarán los distintos tipos de aprendizaje que estén disponibles en el sistema.

```
DROP TABLE IF EXISTS ‘OSM_TYPE_LEARNING‘;
CREATE TABLE IF NOT EXISTS ‘OSM_TYPE_LEARNING‘ (
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Identifier of the type’,
‘VAL_TYPE‘ varchar(8) NOT NULL COMMENT ’Name of the type’,
PRIMARY KEY (‘IDN_TYPE‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Types of learning’;
```

Aprendizajes

En esta tabla encontraremos todos los aprendizajes realizados para poder acceder a cualquiera de ellos o, incluso, hacer predicciones sobre alguno de ellos en específico.

```
DROP TABLE IF EXISTS ‘OSM_LEARNING_FREQUENT_PATTERN‘;
CREATE TABLE IF NOT EXISTS ‘OSM_LEARNING_FREQUENT_PATTERN‘ (
‘IDN_LEARNING‘ varchar(8) NOT NULL COMMENT ’Learning identifier’,
‘NUM_SUPPORT_MIN‘ int(11) NOT NULL COMMENT ’Minimun support for
the learning’,
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Type of learning’,
‘DTI_INI‘ timestamp NOT NULL default CURRENT_TIMESTAMP
COMMENT ’Inicial time of the learning’,
‘DTI_FIN‘ timestamp NOT NULL default ’0000-00-00 00:00:00’
COMMENT ’final time of the learning’,
‘IDN_SERVICE‘ varchar(8) collate latin1_bin default NULL
COMMENT ’Which service do you want to watch’,
‘IND_ACTIVE‘ int(1) default NULL COMMENT ’The user marks if the learning
is actived’,
PRIMARY KEY (‘IDN_LEARNING‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT=’Learning manager for the frequent pattern mining’;
ALTER TABLE ‘OSM_LEARNING_FREQUENT_PATTERN‘
ADD CONSTRAINT ‘osm_learning_frecuent_pattern_ibfk_1‘
FOREIGN KEY (‘IDN_TYPE‘)
REFERENCES ‘OSM_TYPE_LEARNING‘ (‘IDN_TYPE‘),
ADD CONSTRAINT ‘osm_learning_frecuent_pattern_ibfk_2‘
FOREIGN KEY (‘IDN_SERVICE‘)
REFERENCES ‘OSM_SERVICE‘ (‘IDN_SERVICE‘);
```

Tipos de valores

Serán los tipos de valores que podemos encontrar. Inicialmente estarán la criticidad y la disponibilidad.

```
DROP TABLE IF EXISTS ‘OSM_TYPE_VALUE‘;
CREATE TABLE IF NOT EXISTS ‘OSM_TYPE_VALUE‘ (
‘IDN_VALUE‘ varchar(8) NOT NULL COMMENT ’Rule identifier’,
‘VAL_VALUE‘ varchar(8) NOT NULL COMMENT ’Name of the type of rule’,
PRIMARY KEY (‘IDN_VALUE‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Types of rules’ ;
```

Antecedentes

Son los antecedentes de las reglas creadas en el aprendizaje.

```
DROP TABLE IF EXISTS ‘OSM_FREQUENT_PATTERN_RULE_ANTECEDENT‘;
CREATE TABLE IF NOT EXISTS
‘OSM_FREQUENT_PATTERN_RULE_ANTECEDENT‘ (
‘IDN_RULE‘ varchar(8) NOT NULL COMMENT ’Rule identifier’,
‘NUM_ORDER‘ int(11) NOT NULL COMMENT ’Order of the instance in the rule’,
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Instance identifier’,
‘IND_VALUE‘ int(11) NOT NULL COMMENT ’Value of the instance’,
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Type of rule’,
PRIMARY KEY (‘IDN_RULE‘,‘NUM_ORDER‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Rule antecedent’;
ALTER TABLE ‘OSM_FREQUENT_PATTERN_RULE_ANTECEDENT‘
ADD CONSTRAINT ‘osm_frequent_pattern_rule_antecedent_ibfk_1‘
FOREIGN KEY (‘IDN_TYPE‘)
REFERENCES ‘OSM_TYPE_VALUE‘ (‘IDN_VALUE‘);
```

Consecuentes

Serán los consecuentes de las reglas creadas en el aprendizaje y que han de corresponder con los antecedentes mencionados.

```
DROP TABLE IF EXISTS
‘OSM_FREQUENT_PATTERN_RULE_RESULTING‘;
CREATE TABLE IF NOT EXISTS
‘OSM_FREQUENT_PATTERN_RULE_RESULTING‘ (
‘IDN_RULE‘ varchar(8) NOT NULL COMMENT ’Rule identifier’,
‘NUM_ORDER‘ int(11) NOT NULL COMMENT ’Order of the instance in the rule’,
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Instance identifier’,
‘IDN_VALUE‘ varchar(8) NOT NULL COMMENT ’Value of the instance’,
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Type of rule’,
PRIMARY KEY (‘IDN_RULE‘,‘NUM_ORDER‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Rule Resulting’;
ALTER TABLE ‘OSM_FREQUENT_PATTERN_RULE_RESULTING‘
ADD CONSTRAINT ‘osm_frequent_pattern_rule_resulting_ibfk_1‘
FOREIGN KEY (‘IDN_TYPE‘)
REFERENCES ‘OSM_TYPE_VALUE‘ (‘IDN_VALUE‘),
```

```
ADD CONSTRAINT `osm_frequent_pattern_rule_resulting_ibfk_2`
FOREIGN KEY (`IDN_RULE`)
REFERENCES `OSM_FREQUENT_PATTERN_RULE_ANTECEDENT`
(`IDN_RULE`);
```

Reglas

Hará las funciones de índice de las reglas obtenidas en el aprendizaje.

```
DROP TABLE IF EXISTS `OSM_FREQUENT_PATTERN_RULES`;
CREATE TABLE IF NOT EXISTS `OSM_FREQUENT_PATTERN_RULES` (
`IDN_LEARNING` varchar(8) NOT NULL COMMENT 'Learning identifier',
`IDN_RULE` varchar(8) NOT NULL COMMENT 'Rule identifier',
`NUM_SUPPORT` int(11) NOT NULL COMMENT 'Support of the rule',
`NUM_CONFIDENCE` int(11) NOT NULL COMMENT 'Confidence of the rule',
`IND_VALIDATE` int(1) default NULL COMMENT 'The user marks if the rule is valid',
PRIMARY KEY (`IDN_LEARNING`,`IDN_RULE`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT='Join the rules with their learning';
ALTER TABLE `OSM_FREQUENT_PATTERN_RULES`
ADD CONSTRAINT `osm_frecuent_pattern_rules_ibfk_1`
FOREIGN KEY (`IDN_RULE`)
REFERENCES `OSM_FREQUENT_PATTERN_RULE_ANTECEDENT` (`IDN_RULE`),
ADD CONSTRAINT `osm_frecuent_pattern_rules_ibfk_2`
FOREIGN KEY (`IDN_LEARNING`)
REFERENCES `OSM_LEARNING_FREQUENT_PATTERN` (`IDN_LEARNING`);
```

Predicciones

En esta tabla se encontrarán las predicciones realizadas en el módulo de predicciones.

```
DROP TABLE IF EXISTS `OSM_FREQUENT_PATTERN_PREDICTION`;
CREATE TABLE IF NOT EXISTS `OSM_FREQUENT_PATTERN_PREDICTION` (
`IDN_LEARNING` varchar(8) NOT NULL COMMENT 'Learning identifier',
```

```
‘IDN_RULE‘ varchar(8) NOT NULL COMMENT ’Rule identifier’,
‘IND_VALIDATE‘ int(1) default NULL COMMENT ’The user marks if the prediction is valid’,
‘DTI_PREDICTION‘ timestamp NOT NULL default
CURRENT_TIMESTAMP COMMENT ’Inicial time of the prediction’,
‘NUM_HOURS‘ int(11) NOT NULL default 1 COMMENT ’Hours before the prediction’,
PRIMARY KEY (‘IDN_LEARNING‘,‘IDN_RULE‘,‘DTI_PREDICTION‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT=’Join the prediction with the learning’;
ALTER TABLE ‘OSM_FREQUENT_PATTERN_PREDICTION‘
ADD CONSTRAINT ‘osm_frecuent_pattern_prediction_ibfk_1‘
FOREIGN KEY (‘IDN_RULE‘)
REFERENCES ‘OSM_FREQUENT_PATTERN_RULE_ANTECEDENT‘ (‘IDN_RULE‘),
ADD CONSTRAINT ‘osm_frecuent_pattern_prediction_ibfk_2‘
FOREIGN KEY (‘IDN_LEARNING‘)
REFERENCES ‘OSM_LEARNING_FREQUENT_PATTERN‘ (‘IDN_LEARNING‘);
```

Sequential Pattern Mining

En las siguientes tablas encontraremos todo lo necesario para realizar la persistencia de datos al utilizar el algoritmo sequential pattern mining.

Aprendizajes

En esta tabla encontraremos todos los aprendizajes realizados para poder acceder a cualquiera de ellos o, incluso, hacer predicciones sobre alguno de ellos en específico.

```
DROP TABLE IF EXISTS ‘OSM_LEARNING_SEQUENTIAL_PATTERN‘;
CREATE TABLE IF NOT EXISTS ‘OSM_LEARNING_SEQUENTIAL_PATTERN‘ (
‘IDN_LEARNING‘ varchar(8) NOT NULL COMMENT ’Learning identifier’,
‘NUM_SUPPORT_MIN‘ int(11) NOT NULL
COMMENT ’Minimun support for the learning’,
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Type of learning’,
‘DTI_INI‘ timestamp NOT NULL default
CURRENT_TIMESTAMP COMMENT ’Inicial time of the learning’,
‘DTI_FIN‘ timestamp NOT NULL default
’0000-00-00 00:00:00’ COMMENT ’final time of the learning’,
‘IDN_SERVICE‘ varchar(8) default NULL
COMMENT ’Which service do you want to watch’,
‘NUM_LONG_MAX‘ int(11) default NULL
COMMENT ’Maximun longer of the sequence’,
```

```
‘IND_ACTIVE‘ int(1) default NULL
COMMENT ’The user marks if the learning is actived’,
PRIMARY KEY (‘IDN_LEARNING‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT=’Learning manager for the sequential pattern mining’;
ALTER TABLE ‘OSM_LEARNING_SEQUENTIAL_PATTERN‘
ADD CONSTRAINT ‘osm_learning_sequential_pattern_ibfk_1‘
FOREIGN KEY (‘IDN_TYPE‘)
REFERENCES ‘OSM_TYPE_LEARNING‘ (‘IDN_TYPE‘),
ADD CONSTRAINT ‘osm_learning_sequential_pattern_ibfk_2‘
FOREIGN KEY (‘IDN_SERVICE‘)
REFERENCES ‘OSM_SERVICE_INSTANCES‘ (‘IDN_SERVICE‘);
```

Secuencias

En esta tabla se encuentran ordenadas todas las secuencias de los aprendizajes.

```
DROP TABLE IF EXISTS ‘OSM_SEQUENTIAL_PATTERN_SEQUENCE‘;
CREATE TABLE IF NOT EXISTS ‘OSM_SEQUENTIAL_PATTERN_SEQUENCE‘ (
‘IDN_SEQUENCE‘ varchar(8) NOT NULL COMMENT ’Sequence identifier’,
‘NUM_ORDER‘ int(11) NOT NULL COMMENT ’Order of the instance in the sequence’,
‘NUM_ASSOCIATION‘ int(11) NOT NULL
COMMENT ’Association of the instance in the sequence’,
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Instance identifier’,
‘IND_VALUE‘ int(11) NOT NULL COMMENT ’Value of the instance’,
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Type of sequence’,
PRIMARY KEY (‘IDN_SEQUENCE‘,‘NUM_ORDER‘,‘NUM_ASSOCIATION‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Sequences’;

ALTER TABLE ‘OSM_SEQUENTIAL_PATTERN_SEQUENCE‘
ADD CONSTRAINT ‘osm_sequential_pattern_sequence_ibfk_1‘
FOREIGN KEY (‘IDN_TYPE‘)
REFERENCES ‘OSM_TYPE_VALUE‘ (‘IDN_VALUE‘);
```

Aprendizaje con secuencias

Esta tabla podemos ver las secuencias que pertenecen a cada aprendizaje.

```
DROP TABLE IF EXISTS ‘OSM_LEARNING_SEQUENCES‘;
CREATE TABLE IF NOT EXISTS ‘OSM_LEARNING_SEQUENCES‘ (
‘IDN_LEARNING‘ varchar(8) NOT NULL COMMENT ’Learning identifier’,
‘IDN_SEQUENCE‘ varchar(8) NOT NULL COMMENT ’Sequence identifier’,
‘NUM_SUPPORT‘ int(11) NOT NULL COMMENT ’Support of the sequence’,
‘NUM_CONFIDENCE‘ int(11) NOT NULL COMMENT ’Confidence of the sequence’,
‘IND_VALIDATE‘ int(1) default NULL
COMMENT ’The user marks if the rule is valid’,
PRIMARY KEY (‘IDN_LEARNING‘,‘IDN_SEQUENCE‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT=’Join the sequences with their learning’;

ALTER TABLE ‘OSM_LEARNING_SEQUENCES‘
ADD CONSTRAINT ‘osm_learning_sequences_ibfk_1‘
FOREIGN KEY (‘IDN_SEQUENCE‘)
REFERENCES
‘OSM_SEQUENTIAL_PATTERN_SEQUENCE‘ (‘IDN_SEQUENCE‘),
ADD CONSTRAINT ‘osm_learning_sequences_ibfk_2‘
FOREIGN KEY (‘IDN_LEARNING‘)
REFERENCES ‘OSM_LEARNING_SEQUENTIAL_PATTERN‘ (‘IDN_LEARNING‘);
```

Predicciones

En esta tabla se encontrarán las predicciones realizadas en el módulo de predicciones.

```
DROP TABLE IF EXISTS ‘OSM_SEQUENTIAL_PATTERN_PREDICTION‘;
CREATE TABLE IF NOT EXISTS
‘OSM_SEQUENTIAL_PATTERN_PREDICTION‘ (
‘IDN_LEARNING‘ varchar(8) NOT NULL COMMENT ’Learning identifier’,
‘IDN_SEQUENCE‘ varchar(8) NOT NULL COMMENT ’Sequence identifier’,
‘IND_VALIDATE‘ int(1) default NULL
COMMENT ’The user marks if the prediction is valid’,
‘DTI_PREDICTION‘ timestamp NOT NULL default
CURRENT_TIMESTAMP COMMENT ’Inicial time of the prediction’,
‘NUM_HOURS‘ int(11) NOT NULL default 1
COMMENT ’Hours before the prediction’,
```

```
‘NUM_BEFORE_ASSOCIATION‘ int(11)
default NULL COMMENT ’Instances before the current time’,
PRIMARY KEY (‘IDN_LEARNING‘,‘IDN_SEQUENCE‘,‘DTI_PREDICTION‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT=’Join the prediction with the learning’;

ALTER TABLE ‘OSM_SEQUENTIAL_PATTERN_PREDICTION‘
ADD CONSTRAINT ‘osm_sequential_pattern_prediction_ibfk_1‘
FOREIGN KEY (‘IDN_SEQUENCE‘)
REFERENCES ‘OSM_SEQUENTIAL_PATTERN_SEQUENCE‘ (‘IDN_SEQUENCE‘),
ADD CONSTRAINT ‘osm_sequential_pattern_prediction_ibfk_2‘
FOREIGN KEY (‘IDN_LEARNING‘)
REFERENCES ‘OSM_LEARNING_SEQUENTIAL_PATTERN‘ (‘IDN_LEARNING‘);
```

Constraints based Sequential Pattern Mining

En las siguientes tablas encontraremos todo lo necesario para realizar la persistencia de datos al utilizar el algoritmo constraints bassed sequential pattern mining.

Aprendizajes

En esta tabla encontraremos todos los aprendizajes realizados para poder acceder a cualquiera de ellos o, incluso, hacer predicciones sobre alguno de ellos en específico.

```
DROP TABLE IF EXISTS
‘OSM_LEARNING_CONSTRAINT_SEQUENTIAL_PATTERN‘;
CREATE TABLE IF NOT EXISTS
‘OSM_LEARNING_CONSTRAINT_SEQUENTIAL_PATTERN‘ (
‘IDN_LEARNING‘ varchar(8) NOT NULL COMMENT ’Learning identifier’,
‘NUM_SUPPORT_MIN‘ int(11) NOT NULL
COMMENT ’Minimun support for the learning’,
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Type of learning’,
‘DTI_INI‘ timestamp NOT NULL
default CURRENT_TIMESTAMP COMMENT ’Inicial time of the learning’,
‘DTI_FIN‘ timestamp NOT NULL
default ’0000-00-00 00:00:00’ COMMENT ’final time of the learning’,
‘IND_ACTIVE‘ int(1) default NULL
COMMENT ’The user marks if the learning is actived’,
PRIMARY KEY (‘IDN_LEARNING‘)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT='Learning manager for the constraint bassed sequential pattern mining';

ALTER TABLE `OSM_LEARNING_CONSTRAINT_SEQUENTIAL_PATTERN`
ADD CONSTRAINT `osm_learning_sequential_pattern_ibfk_1`
FOREIGN KEY (`IDN_TYPE`)
REFERENCES `OSM_TYPE_LEARNING` (`IDN_TYPE`);
```

Restricciones

Serán las restricciones que podemos poner a la hora de realizar nuestros aprendizajes.

```
DROP TABLE IF EXISTS `OSM_CONSTRAINT`;
CREATE TABLE IF NOT EXISTS `OSM_CONSTRAINT` (
`IDN_CONSTRAINT` varchar(8) NOT NULL COMMENT 'Constraint identifier',
`VAL_CONSTRAINT` varchar(8) NOT NULL COMMENT 'Name of the constraint',
PRIMARY KEY (`IDN_CONSTRAINT`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT='Constraint of the learning' ;
```

Aprendizaje con restricciones

Indica las restricciones que tiene activadas cada aprendizaje.

```
DROP TABLE IF EXISTS `OSM_LEARNING_CONSTRAINTS`;
CREATE TABLE IF NOT EXISTS `OSM_LEARNING_CONSTRAINTS` (
`IDN_LEARNING` varchar(8) NOT NULL
COMMENT 'Learning identifier',
`IDN_CONSTRAINT` varchar(8) NOT NULL
COMMENT 'Constraint identifier',
`IND_VALUE` int(11) NOT NULL COMMENT 'Value of the constraint',
PRIMARY KEY (`IDN_LEARNING`,`IDN_CONSTRAINT`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT='Learning with the constraint';
ALTER TABLE `OSM_LEARNING_CONSTRAINTS`
ADD CONSTRAINT `osm_learning_constraint_ibfk_1`
FOREIGN KEY (`IDN_LEARNING`)
```

```
REFERENCES
‘OSM_LEARNING_CONSTRAINT_SEQUENTIAL_PATTERN‘ (‘IDN_LEARNING‘),
ADD CONSTRAINT ‘osm_learning_constraint_ibfk_2‘
FOREIGN KEY (‘IDN_CONSTRAINT‘)
REFERENCES ‘OSM_CONSTRAINT‘ (‘IDN_CONSTRAINT‘);
```

Secuencias

En esta tabla se encuentran ordenadas todas las secuencias de los aprendizajes.

```
DROP TABLE IF EXISTS
‘OSM_CONSTRAINT_SEQUENTIAL_PATTERN_SEQUENCE‘;
CREATE TABLE IF NOT EXISTS
‘OSM_CONSTRAINT_SEQUENTIAL_PATTERN_SEQUENCE‘ (
‘IDN_SEQUENCE‘ varchar(8) NOT NULL COMMENT ’Sequence identifier’,
‘NUM_ORDER‘ int(11) NOT NULL
COMMENT ’Order of the instance in the sequence’,
‘NUM_ASSOCIATION‘ int(11) NOT NULL
COMMENT ’Association of the instance in the sequence’,
‘IDN_INSTANCE‘ varchar(8) NOT NULL COMMENT ’Instance identifier’,
‘NUM_VALUE‘ varchar(8) NOT NULL COMMENT ’Value of the instance’,
‘DTI_TIME_INTERVAL_INI‘ timestamp NOT NULL
default ’0000-00-00 00:00:00’
COMMENT ’Value of the inicial interval time between two instances’,
‘DTI_TIME_INTERVAL_INI‘ timestamp NOT NULL
default ’0000-00-00 00:00:00’
COMMENT ’Value of the inicial interval time between two instances’,
‘DTI_TIME_INTERVAL_FIN‘ timestamp NOT NULL
COMMENT ’Value of the final interval time between two instances’,
‘IDN_TYPE‘ varchar(8) NOT NULL COMMENT ’Type of sequence’,
PRIMARY KEY (‘IDN_SEQUENCE‘,‘NUM_ORDER‘,‘NUM_ASSOCIATION‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT=’Sequences’;

ALTER TABLE ‘OSM_CONSTRAINT_SEQUENTIAL_PATTERN_SEQUENCE‘
ADD CONSTRAINT ‘osm_constraint_sequential_pattern_sequence_ibfk_1‘
FOREIGN KEY (‘IDN_TYPE‘)
REFERENCES ‘OSM_TYPE_VALUE‘ (‘IDN_VALUE‘);
```

Aprendizaje con secuencias

Esta tabla podemos ver las secuencias que pertenecen a cada aprendizaje.

```
DROP TABLE IF EXISTS ‘OSM_LEARNING_CONSTRAINTS_SEQUENCES‘;
CREATE TABLE IF NOT EXISTS
‘OSM_LEARNING_CONSTRAINTS_SEQUENCES‘ (
‘IDN_LEARNING‘ varchar(8) NOT NULL COMMENT ’Learning identifier’,
‘IDN_SEQUENCE‘ varchar(8) NOT NULL COMMENT ’Sequence identifier’,
‘NUM_SUPPORT‘ int(11) NOT NULL COMMENT ’Support of the sequence’,
‘IND_VALIDATE‘ int(1) default NULL
COMMENT ’The user marks if the rule is valid’,
PRIMARY KEY (‘IDN_LEARNING‘,‘IDN_SEQUENCE‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT=’Join the sequences with their learning’;
ALTER TABLE ‘OSM_LEARNING_CONSTRAINTS_SEQUENCES‘
ADD CONSTRAINT ‘osm_learning_constraints_sequences_ibfk_1‘
FOREIGN KEY (‘IDN_SEQUENCE‘)
REFERENCES ‘OSM_CONSTRAINT_SEQUENTIAL_PATTERN_SEQUENCE‘
(‘IDN_SEQUENCE‘),
ADD CONSTRAINT ‘osm_learning_constraints_sequences_ibfk_2‘
FOREIGN KEY (‘IDN_LEARNING‘)
REFERENCES ‘OSM_LEARNING_CONSTRAINT_SEQUENTIAL_PATTERN‘
(‘IDN_LEARNING‘);
```

Predicciones

En esta tabla se encontrarán las predicciones realizadas en el módulo de predicciones.

```
DROP TABLE IF EXISTS
‘OSM_CONSTRAINTS_SEQUENTIAL_PATTERN_PREDICTION‘;
CREATE TABLE IF NOT EXISTS
‘OSM_CONSTRAINTS_SEQUENTIAL_PATTERN_PREDICTION‘ (
‘IDN_LEARNING‘ varchar(8) NOT NULL COMMENT ’Learning identifier’,
‘IDN_SEQUENCE‘ varchar(8) NOT NULL COMMENT ’Sequence identifier’,
‘IND_VALIDATE‘ int(1) default NULL
COMMENT ’The user marks if the prediction is valid’,
‘DTI_PREDICTION‘ timestamp NOT NULL
default CURRENT_TIMESTAMP COMMENT ’Inicial time of the prediction’,
```

```
‘NUM_HOURS‘ int(11) NOT NULL default 1 COMMENT ’Hours before the prediction’,
PRIMARY KEY (‘IDN_LEARNING‘,‘IDN_SEQUENCE‘,‘DTI_PREDICTION‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
COMMENT=’Join the prediction with the learning’;
ALTER TABLE ‘OSM_CONSTRAINTS_SEQUENTIAL_PATTERN_PREDICTION‘
ADD CONSTRAINT ‘osm_constraints_sequential_pattern_prediction_ibfk_1‘
FOREIGN KEY (‘IDN_SEQUENCE‘)
REFERENCES ‘OSM_CONSTRAINT_SEQUENTIAL_PATTERN_SEQUENCE‘
(‘IDN_SEQUENCE‘),
ADD CONSTRAINT ‘osm_constraints_sequential_pattern_prediction_ibfk_2‘
FOREIGN KEY (‘IDN_LEARNING‘)
REFERENCES ‘OSM_LEARNING_CONSTRAINT_SEQUENTIAL_PATTERN‘
(‘IDN_LEARNING‘);
```

Apéndice B

Algoritmos de predicción

En esta sección vamos a ver el código más relevante dentro de los algoritmos de predicción. Empezaremos por las predicciones relacionadas con el algoritmo de aprendizaje frequent pattern mining y continuaremos con el correspondiente al sequential pattern mining.

Predicción frequent pattern mining

Se empezará estudiando el algoritmo de predicción de éste tipo de aprendizaje en la versión que tiene menos restricciones.

Primero recogeremos los datos que queremos analizar, que corresponderán al momento actual (todos los eventos ocurridos entre el momento de inicio dti_ini y el tiempo que se le indique time). Los vamos a almacenar en una vista auxiliar de la base de datos, puesto que manejarlos al completo desde la aplicación ralentizaría el algoritmo.

```
if (dti_ini == NULL ) {
sprintf(query, CREATE OR REPLACE VIEW
OSM_VW_AUX_PREDICTION_FREQUENT AS
SELECT
older.IDN_INSTANCE AS instance,
older.IND_AVAILABILITY AS valor,
AVAILABILITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_AVAILABILITIES AS older
JOIN OSM_SERVICE_INSTANCES
```

```
WHERE older.IDN_INSTANCE =
OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.IND_AVAILABILITY = 0
AND older.DTI_FINAVAILABILITY = 2038-01-01
GROUP BY older.IDN_INSTANCE
if (dti_ini == NULL ) {
sprintf(query, CREATE OR REPLACE VIEW
OSM_VW_AUX_PREDICTION_FREQUENT AS
SELECT
older.IDN_INSTANCE AS instance,
older.IND_AVAILABILITY AS valor,
AVAILABILITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_AVAILABILITIES AS older
JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE =
OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.IND_AVAILABILITY = 0
AND older.DTI_FINAVAILABILITY = 2038-01-01
GROUP BY older.IDN_INSTANCE
UNION ALL
SELECT
older.IDN_INSTANCE AS instance,
older.TYP_CRITICITY AS valor,
CRITICITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_CRITICITIES AS older
JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE =
OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.TYP_CRITICITY = 2
AND older.DTI_FINCRITICITY = 2038-01-01
GROUP BY older.IDN_INSTANCE );
}
```

```
else {
sprintf(query, CREATE OR REPLACE VIEW
OSM_VW_AUX_PREDICTION_FREQUENT AS
SELECT
older.IDN_INSTANCE AS instance,
older.IND_AVAILABILITY AS valor,
AVAILABILITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_AVAILABILITIES AS older
JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE = OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.IND_AVAILABILITY = 0
AND older.DTI_INIAVAILABILITY <= %s
AND older.DTI_FINAVAILABILITY >=
DATE_SUB( %s, INTERVAL %s HOUR_MINUTE)
GROUP BY older.IDN_INSTANCE
UNION ALL
SELECT
older.IDN_INSTANCE AS instance,
older.TYP_CRITICITY AS valor,
CRITICITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_CRITICITIES AS older
JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE =
OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.TYP_CRITICITY = 2
AND older.DTI_INICRITICITY <= %s
AND older.DTI_FINCRITICITY >=
DATE_SUB( %s, INTERVAL %s HOUR_MINUTE)
GROUP BY older.IDN_INSTANCE
ORDER BY instance , dti_ini, dti_ini, time,
dti_ini, dti_ini, time);
}
```

A continuación, se van a buscar todos aquellos datos que se encuentren en la vista creada, que coincidan con alguno de los antecedentes de las reglas construidas por el algoritmo de aprendizaje Frequent Pattern Mining.

Para realizar esta tarea hemos de recordar que las reglas están divididas en antecedente

y consecuente ($A \Rightarrow C$). Por tanto, vamos a establecer la restricción de que el antecedente y el consecuente no sean el mismo evento. Si no se hiciera dicha restricción, cualquier evento ocurriría siempre.

Además, es posible definir en dicha selección que todas las instancias que provocaron esos eventos sean de un mismo servicio o pertenezcan al un mismo aprendizaje, en caso de que existan varios.

```
if (idn_service == NULL) {
sprintf(query, SELECT
b. IDN_RULE
FROM OSM_VW_AUX_PREDICTION_FREQUENT a,
OSM_FREQUENT_PATTERN_RULE_ANTECEDENT b,
OSM_FREQUENT_PATTERN_RULES c
WHERE a. instance like b. IDN_INSTANCE
AND c. IDN_RULE = b. IDN_RULE
and c. IDN_LEARNING LIKE %s
and b. IDN_RULE NOT IN (SELECT
c. IDN_RULE
FROM OSM_FREQUENT_PATTERN_RULE_RESULTING c
WHERE b. IDN_INSTANCE LIKE c. IDN_INSTANCE )
GROUP BY b. IDN_RULE , aux);
}
else {
sprintf(query, SELECT
b. IDN_RULE
FROM OSM_VW_AUX_PREDICTION_FREQUENT a,
OSM_FREQUENT_PATTERN_RULE_ANTECEDENT b,
OSM_FREQUENT_PATTERN_RULES c
WHERE a. instance like b. IDN_INSTANCE
AND c. IDN_RULE = b. IDN_RULE
and c. IDN_LEARNING LIKE %s
and a. service like %s
and b. IDN_RULE NOT IN (SELECT
c. IDN_RULE
FROM OSM_FREQUENT_PATTERN_RULE_RESULTING c
}
```

```
WHERE b. IDN_INSTANCE LIKE c. IDN_INSTANCE )
GROUP BY b. IDN_RULE ,
aux, idn_service);
}
```

Después, vamos a proceder a preguntar la hora al sistema, ya que es un dato importante para importar los resultados a la base de datos.

```
sprintf(query, SELECT NOW() );
if (-1==this->db_connector_.execute_query(query,res_set_aux)){
mysql_free_result (res_set_aux);
}
```

Una vez hemos recabado toda la información necesaria, procederemos a insertar los datos con el formato correcto a la base de datos para que se almacenen los resultados.

```
for (int i=0; i<num_rows;i++){
row = mysql_fetch_row (res_set);
if (dti_ini == NULL){
sprintf(query, INSERT INTO OSM_FREQUENT_PATTERN_PREDICTION
( IDN_LEARNING, IDN_RULE, VALIDATE, DTI_PREDICTION, NUM_HOURS )
VALUES
( %s, %s, 1, %s, %s),
aux, row[0], row_time[0], time);
}
else {
sprintf(query, INSERT INTO OSM_FREQUENT_PATTERN_PREDICTION
( IDN_LEARNING, IDN_RULE, VALIDATE, DTI_PREDICTION, NUM_HOURS )
VALUES
( %s, %s, 1, %s, %s),
aux, row[0], dti_ini, time);
}
}
```

Para finalizar borraremos la vista auxiliar que hemos utilizado para almacenar los eventos ocurridos entre las fechas que se marcaron.

```
sprintf(query, DROP VIEW IF EXISTS
OSM_VW_AUX_PREDICTION_FREQUENT );
```

Predicción sequential pattern mining

Se empezará estudiando el algoritmo de predicción de este tipo de aprendizaje en la versión que tiene menos restricciones.

Primero recogeremos los datos que queremos analizar, que corresponderán al momento actual (todos los eventos ocurridos entre el momento de inicio dti_ini y el tiempo que se le indique time). Los vamos a almacenar en una vista auxiliar de la base de datos, puesto que manejarlos al completo desde la aplicación ralentizaría el algoritmo.

```
if (dti_ini == NULL ){ sprintf(query, CREATE OR REPLACE VIEW
OSM_VW_AUX_PREDICTION_SEQUENCE AS
SELECT older.IDN_INSTANCE AS instance,
older.IND_AVAILABILITY AS valor,
AVAILABILITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_AVAILABILITIES AS older
JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE =
OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.IND_AVAILABILITY = 0
AND older.DTI_FINAVAILABILITY = 2038-01-01
GROUP BY older.IDN_INSTANCE
UNION ALL
SELECT older.IDN_INSTANCE AS instance,
older.TYP_CRITICITY AS valor,
CRITICITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_CRITICITIES AS
older JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE =
OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.TYP_CRITICITY = 2
AND older.DTI_FINCRITICITY = 2038-01-01
GROUP BY older.IDN_INSTANCE );
}
```

```
else{ sprintf(query, CREATE OR REPLACE VIEW
OSM_VW_AUX_PREDICTION_SEQUENCE AS
SELECT older.IDN_INSTANCE AS instance,
older.IND_AVAILABILITY AS valor,
AVAILABILITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_AVAILABILITIES AS older
JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE = OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.IND_AVAILABILITY = 0
AND older.DTI_INIAVAILABILITY <= %s
AND older.DTI_FINAVAILABILITY >=
DATE_SUB( %s, INTERVAL %s HOUR_MINUTE)
GROUP BY older.IDN_INSTANCE
UNION ALL
SELECT older.IDN_INSTANCE AS instance,
older.TYP_CRITICITY AS valor,
CRITICITY AS typeOfValue,
OSM_SERVICE_INSTANCES.IDN_SERVICE AS service
FROM OSM_HISTINST_CRITICITIES AS older
JOIN OSM_SERVICE_INSTANCES
WHERE older.IDN_INSTANCE =
OSM_SERVICE_INSTANCES.IDN_INSTANCE
AND older.TYP_CRITICITY = 2
AND older.DTI_INICRITICITY <= %s
AND older.DTI_FINCRITICITY >=
DATE_SUB( %s, INTERVAL %s HOUR_MINUTE)
GROUP BY older.IDN_INSTANCE
ORDER BY instance , dti_ini, dti_ini,
time, dti_ini, dti_ini, time);
}
```

A continuación, se van a buscar todos aquellos datos que se encuentren en la vista creada, que coincidan con alguna de las secuencias construidas por el algoritmo de aprendizaje Sequential Pattern Mining.

Además, es posible definir en dicha selección que todas las instancias que provocaron esos eventos sean de un mismo servicio o pertenezcan al un mismo aprendizaje, en caso de que existan varios.

También se puede limitar la longitud máxima de las secuencias mediante el atributo long_max, ya que puede que no nos interesen las cadenas de eventos demasiado largas.

```
if (idn_service == NULL){ if (long_max == 0{
sprintf(query, SELECT b. IDN_SEQUENCE
FROM OSM_VW_AUX_PREDICTION_SEQUENCE a,
OSM_SEQUENTIAL_PATTERN_SEQUENCE b,
OSM_LEARNING_SEQUENCE c
WHERE a. instance LIKE b. IDN_INSTANCE
AND c. IDN_SEQUENCE LIKE b. IDN_SEQUENCE
AND c. IDN_LEARNING LIKE %s
GROUP BY b. IDN_SEQUENCE , aux);
else{ sprintf(query, SELECT b. IDN_SEQUENCE
FROM OSM_VW_AUX_PREDICTION_SEQUENCE a,
OSM_SEQUENTIAL_PATTERN_SEQUENCE b,
OSM_LEARNING_SEQUENCE c
WHERE a. instance LIKE b. IDN_INSTANCE
AND c. IDN_SEQUENCE LIKE b. IDN_SEQUENCE
AND c. IDN_LEARNING LIKE %s
AND b. ASSOCIATION <= %s
GROUP BY b. IDN_SEQUENCE ,aux, aux_long);
}
}
else{ if (long_max == 0{ sprintf(query, SELECT b. IDN_SEQUENCE
FROM OSM_VW_AUX_PREDICTION_SEQUENCE a,
OSM_SEQUENTIAL_PATTERN_SEQUENCE b,
OSM_LEARNING_SEQUENCE c
WHERE a. instance LIKE b. IDN_INSTANCE
AND c. IDN_SEQUENCE LIKE b. IDN_SEQUENCE
AND c. IDN_LEARNING LIKE %s
AND a. service LIKE %s
GROUP BY b. IDN_SEQUENCE ,aux, idn_service);
}
else{ sprintf(query, SELECT b. IDN_SEQUENCE
FROM OSM_VW_AUX_PREDICTION_SEQUENCE a,
OSM_SEQUENTIAL_PATTERN_SEQUENCE b,
OSM_LEARNING_SEQUENCE c
WHERE a. instance LIKE b. IDN_INSTANCE
```

```
AND c. IDN_SEQUENCE LIKE b. IDN_SEQUENCE
AND c. IDN_LEARNING LIKE %s
AND b. ASSOCIATION <= %s
AND a. service LIKE %s
GROUP BY b. IDN_SEQUENCE ,aux, aux_long, idn_service);
}
}
```

Después, vamos a proceder a preguntar la hora al sistema, ya que es un dato importante para importar los resultados a la base de datos.

```
sprintf(query, SELECT NOW() );
if (-1==this->db_connector_.execute_query(query,res_set_aux){
mysql_free_result (res_set_aux); res_set_aux=NULL;
}
```

Una vez hemos recabada toda la información necesaria, procederemos a insertar los datos con el formato correcto a la base de datos para que se almacenen los resultados.

```
for (int i=0; i<num_rows;i++){
row = mysql_fetch_row (res_set);
if (dti_ini == NULL{ sprintf(query, INSERT INTO
OSM_SEQUENTIAL_PATTERN_PREDICTION
( idn_lEARNING, IDN_SEQUENCE, VALIDATE,
DTI_PREDICTION, NUM_HOURS, BEFORE_ASSOCIATION )
VALUES
( %s, %s, 1, %s, %s, 0), aux, row[0], row_time[0], time);
}
else{ sprintf(query, INSERT INTO
OSM_SEQUENTIAL_PATTERN_PREDICTION
( IDN_LEARNING, IDN_SEQUENCE,
VALIDATE, DTI_PREDICTION, NUM_HOURS, BEFORE_ASSOCIATION )
VALUES
( %s, %s, 1, %s, %s, 0), aux, row[0],
dti_ini, time);
}
}
```

Para finalizar borraremos la vista auxiliar que hemos utilizado para almacenar los eventos

ocurridos entre las fechas que se marcaron.

```
sprintf(query, DROP VIEW IF EXISTS
OSM_VW_AUX_PREDICTION_SEQUENCE );
```