Check for updates
DOI: https://doi.org/10.34069/AI/2025.86.02.13
How to Cite:
Ramírez-Romero, T.A., Jiménez-Ruíz, R.B., Patiño-Ortiz, J., & Manzanilla-Granados, H.M. (2025). Sistema de inteligencia artificial basado en manejador de reglas dinámico, operado sobre base de datos. *Amazonia Investiga*, *14*(86), 164-179. https://doi.org/10.34069/AI/2025.86.02.13

Sistema de inteligencia artificial basado en manejador de reglas dinámico, operado sobre base de datos

Artificial intelligence system based on a dynamic rule-based engine operated on a database

Received: May 17, 2025 Accepted: June 30, 2025

Written by:
Tonáhtiu Arturo Ramírez-Romero[1]
https://orcid.org/0009-0001-5686-8189
René Baltazar Jiménez-Ruíz[2]
https://orcid.org/0009-0008-3022-9232
Julián Patiño-Ortiz[3]
https://orcid.org/0000-0001-8106-9293
Héctor Manuel Manzanilla-Granados[4]
https://orcid.org/0000-0002-0276-1853

Resumen

Se presenta una propuesta desarrollo de un manejador de reglas, aplicable a sistemas de filtrado, de diagnóstico o para toma de decisiones, esta propuesta permite el manejo de reglas sencillas o compuestas, tanto que incluyan: Operaciones de consultas a bases de datos, operaciones aritméticas y operadores relacionales, acepta reglas conjuntivas y disyuntivas, se crea un pequeño lenguaje para representar reglas, mismas que son almacenadas en base de datos y por tanto pueden crecer dinámicamente. El manejador se utiliza como herramienta para otros desarrollos que requieran esta funcionalidad, el producto final es una librería en lenguaje PHP.

Palabras clave: Manejador de reglas, inteligencia artificial, bases de datos.

Abstract

We present a proposal for the development of a rule handler, applicable to filtering, diagnostic, or decision-making systems. This proposal allows the management of simple or complex rules, including: database query operations, arithmetic operations, and relational operators. It accepts conjunctive and disjunctive rules. A small language is created to represent rules, which are stored in a database and can therefore grow dynamically. The handler is used as a tool for other developments that require this functionality, and the final product is a library in PHP.

Keywords: rule handler, artificial intelligence, database.

Introducción

Dado el incremento del uso de la inteligencia artificial en la actualidad, la representación y manejo del conocimiento ha tomado más importancia, una técnica para manejo de conocimiento son los sistemas basados en reglas con sus variantes (Hou et al., 2025), así como la inteligencia artificial explicable (Tchuente et al., 2024), que permiten que el proceso y salida de la inteligencia artificial sea más comprensible para los humanos.

[1] Dr. Escuela Superior de Cómputo, Instituto Politécnico Nacional México, México. Email: tonahtiu@yahoo.com
[2] M.D. Escuela Superior de Cómputo, Instituto Politécnico Nacional México, México. Email: rbjimenez@ipn.mx
[3] Dr. Escuela Superior de Ingeniería Mecánica y Eléctrica, Instituto Politécnico Nacional México, México. Email: jpatinoo@ipn.mx
[4] Dr. Escuela Superior de Cómputo, Instituto Politécnico Nacional México, México. Email: hmanzanilla@ipn.mx

This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.
Creative Commons Attribution 4.0 International (CC BY 4.0)

Los sistemas basados en árboles de decisión operados a través de manejadores de reglas pueden manejar diferentes tareas como: Clasificación (Pawuś e al., 2025), (Zolfagharnasab et al., 2025), filtraje de datos o información, hacer diagnósticos (Aguilera-Venegas et al., 2023), así como otras tareas con base en un conjunto de reglas fijas definidas por el usuario y dependiendo de la capacidad del sistema computacional inteligente puede o no actualizar dichas reglas para poder resolver nuevas situaciones. De igual forma se observa que el almacenamiento de las reglas en las bases de datos es una buena opción (Martinez Llario et al., 2017).

El propósito de este artículo es mostrar el diseño de un sistema manejador de reglas, este diseño tiene características funcionales, por ejemplo, permite separar las reglas de conocimiento respecto del código fuente, esta separación permite la operación del conocimiento de forma dinámica lo que permite: Agregar, modificar, borrar y explorar reglas con el sistema en operación. Las reglas de conocimiento son almacenadas en una tabla en una base de datos. Otra cualidad de este manejador de reglas es la posibilidad de incluir reglas con acceso a base de datos, con los permisos necesarios. Lo que amplia el uso en la toma de decisiones sobre bases de datos ya operativas y que son alimentadas por otros sistemas.

Este documento viene organizado y se comienza al presentar trabajos similares, posteriormente el marco teórico que proporciona elementos importantes para la mejor comprensión del artículo, posterior a ello se presenta nuestra propuesta, luego los resultados obtenidos, finalmente las conclusiones y las referencias.

Marco teórico y Revisión de literatura

Los sistemas basados en reglas constan de un intérprete y reglas, esto también se puede interpretar de la siguiente forma:

- Motor de inferencia.
- Base de conocimientos.

Motor de inferencia

El papel de interpretar, seleccionar y aplicar reglas es atendido por el motor de inferencia. Mediante inferencia es posible seleccionar el conjunto de reglas que coinciden en el contexto, en este proceso se debe hacer coincidir reglas y resolver resolución de conflictos.

Los motores de inferencia pueden ser de tres tipos:

- Basados en reglas.
- Basados en lógica difusa.
- Basados en reglas Bayesianas.

Base de conocimientos

Se puede decir que una base de conocimientos contiene reglas y hechos, las reglas pueden ser complejas y los hechos pueden incluir: Secuencias, entidades estructuradas, atributos tales como entidades y relaciones entre ellos. Los hechos son fijos y las reglas incluyen variables. Esta información es declarativa, además puede ser interpretada como la unión de dos componentes fundamentales (Santander - Open Academy, 2023)

- Conjunto constituido por hechos o conceptos pertenecientes a un área específica del conocimiento.
- Conjunto de relaciones entre los hechos y conceptos.

Las reglas modelan instancias, casos o hechos, estas instancias pueden ser almacenadas, junto con sus soluciones asociadas como tuplas en una base de datos relacional. Justo con esta propuesta se apoya la idea de almacenar las reglas en una base de datos relacional.

Las reglas se constituyen como una representación del conocimiento en forma de instrucción condicional. Cada una de las reglas se construye de variables de la base de hechos, así como instrucciones de control (if then else), de igual forma pueden incluir operadores AND y OR (Font Fernández, 2008).

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Metareglas

Son reglas que no tienen conexión directa con conocimiento de la aplicación, pero sí de cómo debe aplicarse el conocimiento. Estas son reglas de reglas o reglas del conocimiento. Están directamente relacionadas con los conflictos múltiples y objetivos no acordes, son útiles en resolución de conflictos o bien para dictar el funcionamiento por parte del motor de inferencia para cierto árbol de conocimientos. En este diseño las metareglas pueden indicar que secciones del conocimiento pueden o no se borradas o podadas, también puede definirse una o más metareglas para indicar que debe considerar diferentes líneas del tiempo para ciertas reglas o que no considere el tiempo.

Resolución de conflictos

Ocasionalmente pueden existir más de una regla que se puede aplicar, dado que cumple los requisitos para ser ejecutadas y es entonces cuando surge un conflicto el cual debe de resolverse de una u otra forma. Para resolver dichos conflictos existen diversos métodos como el uso de: metareglas, prioridades o preferencias del usuario.

Desarrollos similares

Algunos desarrollos que usan sistemas basados en reglas son: Clips, Jess, JBoss Rules, y Soar.

En el caso de CLIPS son las iniciales en inglés de: C Language Integrated Production System (Lyndon Johnson Space Center (NASA), 2025) (Sistema de Producción Integrado en Lenguaje C), mismo que aporta los paradigmas de programación imperativa y orientada a objetos, esto permite combinar reglas con objetos. Facilita el control de la secuencia de reglas que se ejecutan. Actualmente es de dominio público.

El Jess es derivado de clips, está constituido por un motor de reglas y un ambiente de programación. Desarrollado por Ernest Friedman en los laboratorios Nacionales Sandia, ubicados en Livermore, California. (Janes & Garcia, 2025). Se puede conectar a aplicaciones desarrolladas en Java, a través de un motor de inferencia por medio de expresiones. Para su evaluación utiliza reglas declarativas para trabajar razonamiento, usa la lista de hechos conocidos y un conjunto de reglas que tratan de coincidir con estos hechos en su base de hechos. Incluyen consultas de memoria de trabajo, así como el encadenamiento hacia atrás. Soporte al estándar JSR-94. Usa una variante del algoritmo de Rete para procesar reglas.

El JBoss Rules es software libre escrito en Java, el Drools (o JBoss Rules) (Proctor et al., 2025) es una solución que se enfoca en la operación de reglas de negocios con un motor de reglas de producción orientada a objetos con encadenamiento hacia adelante (forward chaining), usa el algoritmo Rete, consiste en nodos interconectados con diversas características, analizan entradas transmitiendo los resultados al próximo nodo en caso de coincidir. Usa el JSR-94 en motor de reglas de negocio, aplicación o servicio. Usa JCR (JackRabbit) para administrar las reglas, usa JAAS para autorizar el acceso.

El SOAR es un desarrollo computacional para construir sistemas que se tengan que tomar decisiones, fue elaborado en la Universidad de Michigan (Laird, 2025), Ayuda en tareas a través de un agente inteligente como es el caso de: Problemas abiertos o cerrados. Así como para representar conocimientos en su forma procedural, declarativo o episódico, en las que se busque una aproximación de la verdad. Presenta un ambiente integrado de trabajo para todas las tareas.

Metodología

Esta propuesta de manejador de reglas tiene antecedentes de desarrollo la tesis doctoral (Ramirez Romero, 2013), aquí se presenta el avance y mejoras, también se considera la propuesta de Wang y Yang (Wang et al., 2016). Primero se presenta un análisis de la plataforma computacional, posteriormente en la sección de resultados se formaliza con las definiciones a usar más adelante, luego se presenta la implantación y finalmente un ejemplo de regla compuesta.

This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Análisis de la plataforma a usar

La plataforma computacional es aquella que indica los programas o software necesarios para la operación de un programa o aplicación. La propuesta se presenta en la ilustración 1.

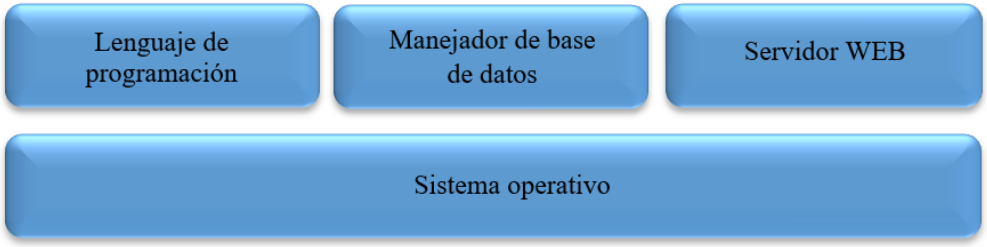

Ilustración 1. Diagrama a bloques de la plataforma necesaria.
Fuente: Elaboración propia.

El sistema operativo sugerido es Linux, con la distribución de fedora core, dada su robustez, ser GNU libre y tiene un manejo adecuado de los recursos. Respecto al manejador de bases de datos la propuesta es MySQL (Smirnova & Tezuysal, 2022), (Grippa & Kuzmichev, 2021), (Widenius et al., 2025) dado que es libre, su desempeño, y su amplia difusión, o bien MariaDB (Widenius, 2025), (Aspin, 2022).

La propuesta de lenguaje de programación es PHP (Lerdof et al., 2025), Se caracteriza por ser rápido en su ejecución.

El servidor de WEB propuesto es HTTPD o Apache httpd server (Behlendorf et al., 2025), es ampliamente usado a nivel mundial en servidores y es (GNU), es configurable, eficiente y corre sobre diferentes plataformas.

El fedora permite trabajar con MySQL + Apache + PHP. O se puede usar: MAMP ó WAMP, que facilita trabajar la propuesta.

Ahora se procede a mostrar en un diagrama a bloques en la ilustración 2, el entorno de funcionamiento de sistema propuesto:

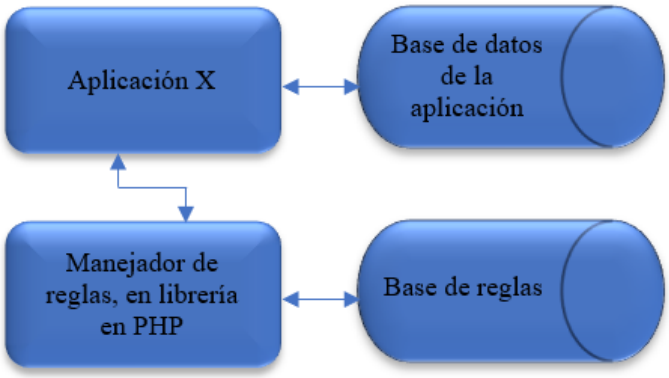

Ilustración 2. Esquema general de entorno.
Fuente: Elaboración propia

En la ilustración 2, se aprecia Aplicación X, que ejemplifica alguna aplicación escrita en PHP, que pueda incluir la librería que contiene el manejador de reglas. Desde la aplicación se incluye el siguiente código:

```
require "reglas_libreria.php";
```

El manejador propuesto almacena las reglas en una base de datos en MySQL, para usar el motor de manejador de bases de datos relacionales y así el esfuerzo puede ser enfocado al desarrollo de mejores algoritmos de búsqueda en árboles que almacenan conocimiento, Se optó por usar el motor MyISAM por su velocidad, con ello se posibilita manejar gran cantidad de reglas sin afectar el tiempo de respuesta.

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

El sistema propuesto se plantea que se constituya de tres módulos:

- Módulo de carga de reglas a la base de conocimiento.
- Módulo evaluador de reglas.
- Módulo constructor de código SQL a partir de reglas.

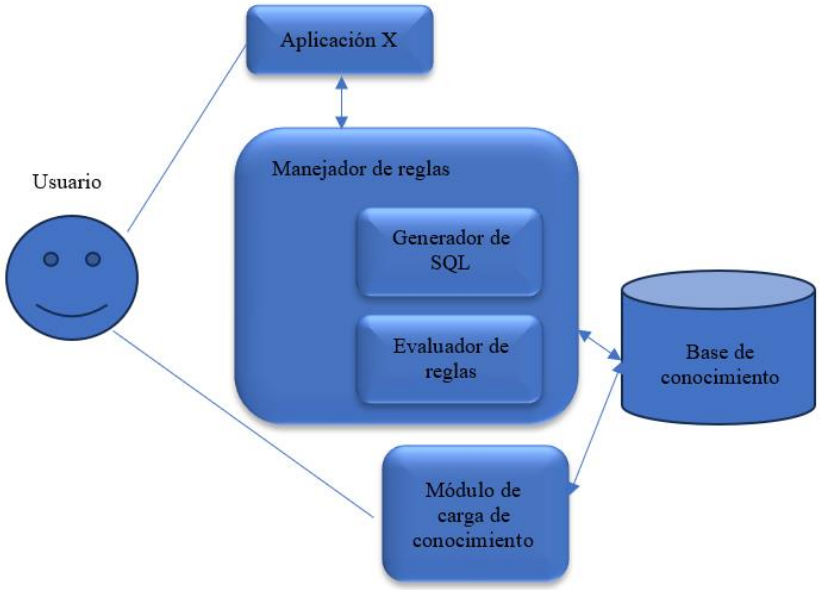

Ilustración 3. Módulos del sistema.
Fuente: Elaboración propia.

Observe la ilustración 3, en el manejador de reglas yace tanto el módulo evaluador de reglas, así como el generador de SQL. El evaluador de reglas esta en permanente uso por cada regla que tenga que ser evaluada, sin embargo, el generador de SQL, solo se activa cuando se evalúa una regla con acceso a base de datos. Por otro lado, observe que el módulo de carga de conocimiento esta fuera de la librería del manejador de reglas, porque este módulo lo que busca es alimentar la base de conocimientos de forma integral y de forma gráfica, sin embargo, se pueden alimentar directamente las reglas a la base de conocimiento.

En este artículo se da énfasis al módulo evaluador o manejador de reglas y aunque los otros módulos (de carga de reglas o ingreso de reglas y constructor de código SQL o traductor reglas a SQL) son parte una parte importante, aquí no se desarrollan.

Resultados y discusión

El producto principal de este trabajo que se mostrará de la siguiente forma: Primero se presentará las definiciones y estructura de datos del árbol de conocimiento, posterior a ello la implantación de la propuesta.

Árbol de conocimientos propuesto.

El manejador de reglas en ocasiones llamado motor de inferencia trabaja directamente con una base de conocimiento, misma que a continuación se define para posteriormente enfocarnos al manejador.

Base de conocimiento

A continuación, se denota la teoría de la Base de Conocimientos (BC en lo sucesivo) y sus componentes.
Definición 1. BC se define formalmente como:

$$BC=Reglas \cup Hechos \cup Func \cup MetaR$$

This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Donde:

- *Reglas*. Las reglas de la Aplicación X. cuyo resultado es cero o uno, Reglas=f(x)={0,1}
- *Hechos*. Reglas con valores constantes.
- *Func*. Funciones, operaciones aritméticas, o a bases de datos, o algún procedimiento y regresan valores o bien datos.
- *MetaR*. Reglas para el manejo de reglas.

Con base en la definición 1, existen diferentes tipos de elementos, en la tabla 1 se presentan algunos ejemplos.

Tabla 1.
Ejemplos de elementos de una BC

Tipo	Ejemplo
Reglas	fecha_aplic <= fecha_actual
Hechos	dias_maximos = 9
Func	fecha_actual
MetaR	tiempo_max_por_regla = 2ms

Fuente: Elaboración propia.

De la tabla 1, la regla del tipo Reglas, recibe como parámetro fecha_aplic y calcula mediante una función implantada en la misma librería en PHP, de tipo *Func* el valor de la fecha_actual, después compara y en caso de coincidir regresa 1, caso contrario regresa 0.

Para el caso de los del tipo *Hechos*, solo se define el valor que puede tomar un parámetro que a su vez es requerido por alguna regla de tipo Reglas.

Las de tipo *Func*, son funciones o procedimientos disponibles en las funciones básicas dentro de la librería del manejador de reglas, por ejemplo, pueden calcular: Fecha actual, fecha máxima, temperatura del CPU, algún dato recogido de algún puerto, etc.

Finalmente, las del tipo *MetaR*, son reglas que rigen la operación general del manejador de reglas. Observe la tabla 1, dice Tiempo_max_por_regla = 2ms, lo que dice que el tiempo máximo que puede emplear en evaluar una regla es de 2 milisegundos. Pero también se puede usar para emplear cierta resolución de conflictos, prioridades o criterios de preferencia de usuario, o bien que para cierta regla debe evaluar la regla, sus hijos y subir un solo nivel y evaluar reglas asociadas al nivel del padre.

Para representar el conocimiento se propone una estructura de árbol abierta, como se aprecia en la ilustración 3.

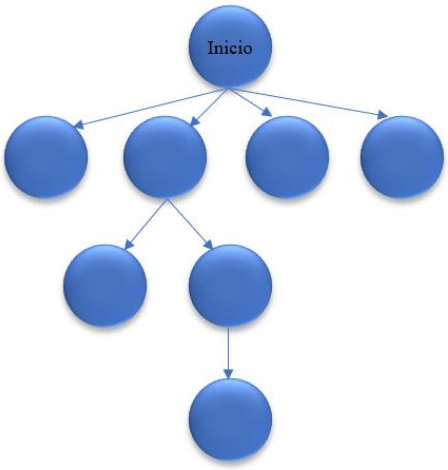

Ilustración 4. Estructura de arbol para representar el conocimiento.
Fuente: Elaboración propia.

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

De la ilustración 3, se dice que cada círculo representa una unidad de conocimiento y por las relaciones permite darle veracidad para cierto entorno.

En la ilustración 4, se aprecia la forma de interconexión de nodos, la idea se toma de (Ramirez Romero, 2013), esta permite construir un árbol general constituido por: m niveles, así como n nodos, para cada nivel.

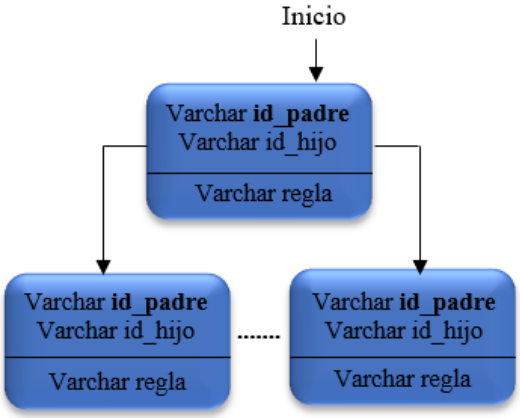

Ilustración 5. Interconexión de nodos.
Fuente: Elaboración propia.

La estructura de datos no lineal de un árbol se traduce en una estructura lineal, pero la estructura lineal puede crecer en cada nodo por un renglón en una base de datos.

Definición 2. Se refine como regla raíz o inicio en lo sucesivo llamada R_{ini} a la primera regla en la estructura de árbol y de la cual se pueden derivar más reglas, R_{ini} => R_1, R_2, …, R_n. y la cual no tiene regla que la anteceda.

$$\forall R_{ini} \mid Padre(R_{ini}) = \{\}$$

Donde: ini representa el inicio.

La definición de la función Padre (R) se da en la definición 3.

Además, solo existe una R_{ini}

En este caso se dice que R_{ini} genera de forma directa a R_1, R_2 y R_n y puede generar hasta n reglas.

Definición 3. Existe una función Padre (R_i) que regresa, generadora de la R_i o también llamada regla antecedente.

R_{ini} => R_i,
Padre (R_i) = R_{ini}

Definición 4. Existe una función Hijo (R_i) que regresa, la(s) regla(s) que se derivan de la regla R_i

R_{ini} => R_i,
R_{ini} => R_k,
Hijo (Rini) = {R_i, …, R_k.}

Definición 5. Se define una BC modelada en una estructura de datos de árbol como: Un conjunto no vacío de expresiones o reglas R_1, R_2, …; R_n de la forma:

$$R_{ini} = \text{Hijo } (R_{ini}) = R_i$$
$$R_{ini} = \text{Hijo } (R_{ini}) = R_j$$
$$R_k = \text{Hijo } (R_j)$$
$$R_n \mid Padre(R_i) \cup R_i \cup Hijo(R_i)$$

This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Donde R_n son reglas del tipo: Metareglas, hechos o funciones, reglas o conceptos necesarios para determinar si una regla es verdadera o falsa.

Definición 6. Línea de razonamiento. Sea BC, un conjunto no vacío de reglas, refiere a una secuencia R_1, R_2, …, R_n de elementos de BC es una línea de razonamiento, para cada R_i (i = 1; …; n-1) las reglas derivadas de esta se representan por la función *Hijo* (R_i) que aparece como parte del ancestro presentada como la función Padre (R_{i+1}) de la regla R_{i+1}. Formalmente, R = R_1; R_2; …; R_n es una línea de razonamiento siempre y cuando sea un conjunto finito de reglas:

$$\forall R_i (1 \leq i > n) \in BC,\ \exists\ R_i\ \subset\ Reglas\ \cup MetaR \mid R_i \subset Hijo(R_i) \wedge Padre(R_{i+1})$$

Definición 7. La función eval(Ci) llamada también de *evalua_regla (Ci)*, recoje un conjunto de datos proporcionadas para el caso a evaluar i, dicha función regresa falso (0) o verdadero (1).

$$eval(C_i) = \{1,0\}$$

Donde C_i, es una condición, se describe más adelante.

Ahora se procede a explicar la representación del manejador de reglas en un sistema computacional.

Implantación del manejador de reglas.

De conformidad con la plataforma previamente propuesta, la implantación se divide en dos partes: La creación de la tabla que guarda las reglas y posteriormente el desarrollo de la librería en lenguaje PHP.
La tabla donde ha de residir la BC según definición 5 se guarda en una tabla llamada reglas con atributos que se aprecian en la ilustración 5.

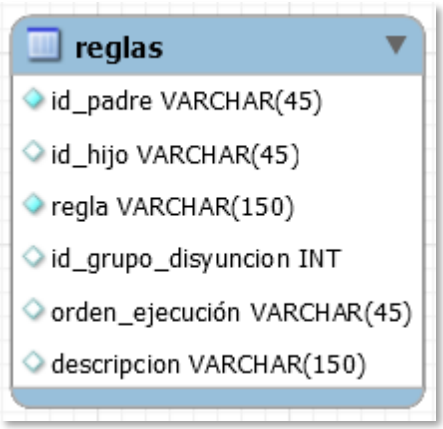

Ilustración 6. Tabla de reglas.
Fuente: Elaboración propia.

De forma breve se explica que los campos *id_padre* e *id_hijo* almacena identificadores, para la regla almacenada en el campo regla, *id grupo disyuncion* guarda un identificador único por grupo de reglas con una operación entre ellas de disyunción lógica tal y como se muestra a continuación.

$$\{R_1, R_2, \ldots, R_n\}\ \in\ G_{id_agrupa} \subset BC\ , n > 1 \mid id\ agrupa\ \in\ \mathbb{Z}^+\ \wedge id\ agrupa > 0$$

Donde G_{id_agrupa}. Es un subconjunto de reglas con un identificador entero positivo para este conjunto de reglas.

El subconjunto $G_{id\ agrupa}$ al ser un conjunto con propiedades disyuntivas y ser evaluada cada regla a través de la función *eval(R_i)*, debe cumplir con la siguiente condición para poderse considerar como verdadera.

$$\sum_{i=1}^{n} eval(R_i) > 0$$

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Donde n es el número total de reglas en este subconjunto.

De la misma entidad reglas de la ilustración 5, el campo *orden_ejecucion* indica el orden en el que han de evaluarse las reglas, y su importancia radica en que cada regla tarda diferente tiempo en ser trabajada y todas deben ser verdaderas para considerar como verdadera cierta condición, dado que se trata de un subconjunto conjuntivo de tal forma que para ahorrar recursos computacionales, el sistema manejador de reglas debe detener la evaluación o exploración de la BC cuando una regla ya no se cumpla.

En primera instancia se deben de colocar las reglas de más rápida ejecución, para disminuir el tiempo de procesamiento.

Función eval(C_i).

En la definición 7 se explicó de forma general lo que hace la función *eval()*, pero esta función es la más importante del sistema manejador de reglas y por tanto a continuación se describe con detalle.

eval(C_i) recibe cadenas de texto que en lo sucesivo se llamarán condiciones, así como el área de conocimiento en la BC donde trabajará en la correspondiente evaluación, posteriormente determinará si es válida o no (cierta o falsa).

La función *eval(Ci)* está diseñada, para que reciba una cadena de entrada con un formato definido y en dicha cadena recibe los parámetros enviados por el usuario, así como una regla donde precisa el lugar de la base de conocimiento donde deberá evaluar.

Al existir la posibilidad de recibir diferente cantidad y tipo de parámetros en la condición de entrada C_i, se optó por recibir una cadena de texto, y posterior a esto se descompone la regla y a través de un análisis sintáctico se obtiene la información requerida.

En la C_i se manejan dos tipos de datos, el primero es una Regla que permite ubicar el nodo dentro de la BC donde se ha de trabajar, es una especie de índice. Los siguientes datos se refieren a parámetros necesarios y proporcionados por el usuario, agente o aplicación y que sirven para evaluar las reglas en el entorno indicado por la regla de ubicación recibida.

El parámetro de ubicación en si es una regla, dentro de la BC y se maneja como variable que se llama $concepto_cve, para este ejemplo asume el valor de 16, como se puede observar:

$concepto_clave = 16;

Por otro lado, los parámetros proporcionados pueden representar diferentes aspectos, así como una o más variables. Para el siguiente ejemplo se maneja el tiempo en dos variables: minutos y fecha, como se ve puede ver:

$minutos = 5;
$fecha = "2016-06-24";

Cuando ya se han cargado tanto el parámetro de ubicación como las variables a evaluarse en la BC, se procede a construir una cadena de texto, pero respetando cierta sintaxis de un pequeño lenguaje de representación, mismo que puede entender la función *eval(Ci)*. A continuación, se presenta la cadena o condición:

$condicion = "Regla: con_clave = 16, Var:fecha_aplica = $fecha, Var:minutos=$minutos";

Este ejemplo es un caso práctico de construcción, escrito para el lenguaje de programación PHP, por lo que las variables comienzan con $, las palabras reservadas del pequeño lenguaje propuesto son: Regla, Var.

El formato propuesto es:

Regla: $R_1 = \$P_1$**, Var:** $R_2 = \$P_2$**, … ,*Var:*** $R_n = \$P_n$

This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Donde:

T={Regla, :, =, ‘,’, Var} Son palabras reservadas y en el ejemplo anterior están en negritas.
$Pn, Parámetros proporcionados por el usuario
$Rn, Reglas que ha de trabajar y que pertenecen a la BC.

Este formato usa la coma (,) como separador entre elementos a procesar, para el caso en que dentro de los parámetros exista más de un elemento para cierto tipo de parámetro se usa el separador punto y coma (;) como se aprecia en el siguiente ejemplo:

Regla: $R_1 = \$P_1$**, Var:** $R_2 = (e_1;\ldots;e_m)$**, … ,*Var:*** $R_n = \$P_n$

Donde:

( ) Elementos de lenguaje para representar conjunto de datos.
$P2 es reemplazado por $(e_1;\ldots;e_m)$

Siendo ex, elementos a ser pasados como un subconjunto a la regla, a continuación, un ejemplo:

Var:x=(2;4)

Ya construida la Ci, se transmite a la función *eval()*

$resultado = *evalua regla($condicion);*

La función *eval(C_i)* producto principal de este trabajo se presenta en el diagrama de flujo en la ilustración 6.

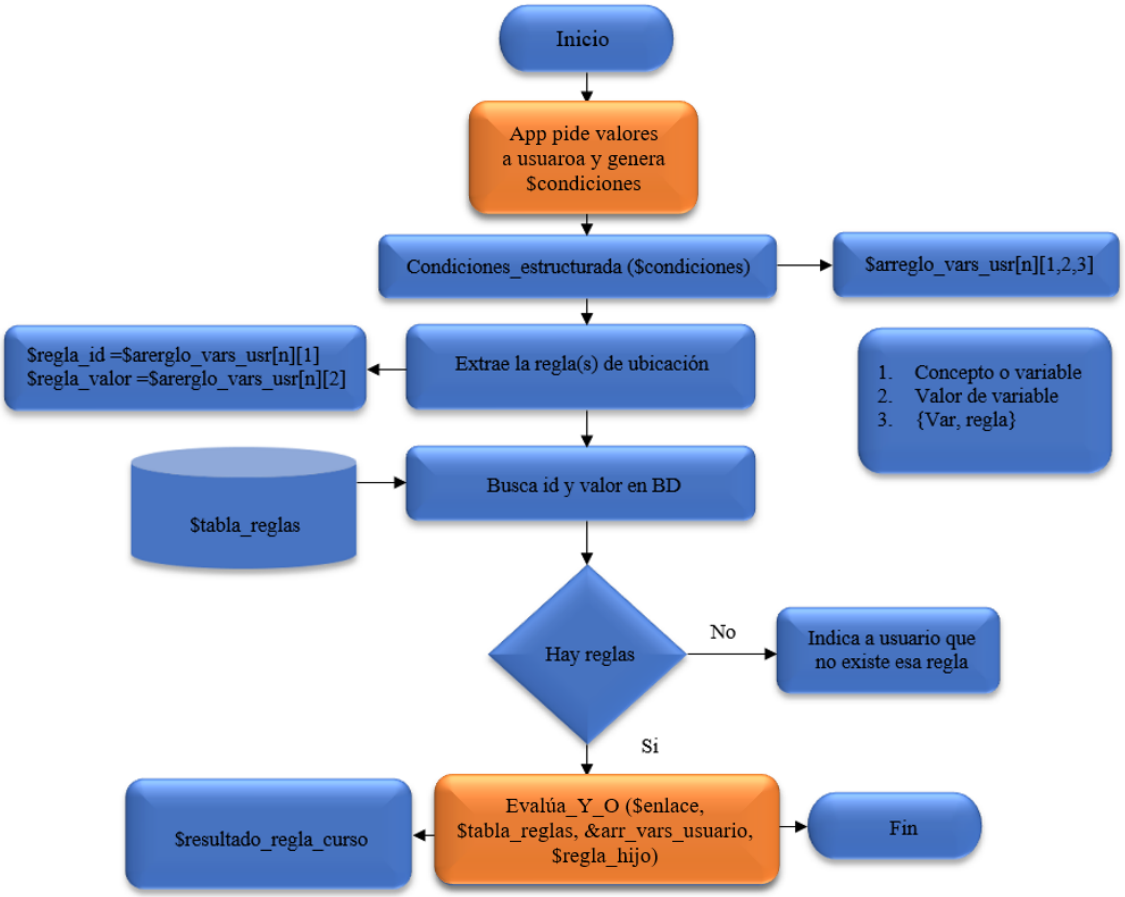

Ilustración 7. Diagrama de flujo de eval(ci).
Fuente: Elaboración propia.

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Ahora se da una descripción de los aspectos más importantes del funcionamiento interno de la función *eval()* representado en la ilustración 7.

En esta ilustración se presenta funciones con borde naranja para identificarlas y en color azul lo demás. La función recibe C_i que es una cadena compuesta que incluye los datos del usuario y una regla de ubicación en el árbol de conocimiento, la función *condiciones_estructura(...)* separa variables de reglas y los pasa a un arreglo bidimensional dinámico, llamado $arreglo_vars_usr, posterior a ello con la regla de ubicación, determina una posición en el árbol que contiene la BC, luego extrae las reglas que aprueba el criterio *hijo*(R_i), después se llama la función *Evalua_Y_O(...)* cuyo diagrama de flujo se muestra en la ilustración 8, luego crea un ciclo donde evaluará cada una de las reglas que devolvió la función y continuará mientras el resultado unitario de cada una de ellas sea 1 o verdadero

Ahora se tiene la función propuesta para trabajar con las reglas conjuntivas o disyuntivas como se ve en la ilustración 8, primero desde inicio, procede a buscar si existen reglas en la BC (denotada por $tabla_reglas), cuyo padre sea la regla que se le está pasando, en caso de existir reglas las distingue por que el valor de *regla_tipo* es vacío, luego determina si se trata de una regla del tipo disyuntivo evaluando *id_agrupa* considerando lo siguiente:

$$id_agrupa \in \mathbb{Z}^+$$

Si cumple la condición procede a buscar las demás reglas con ese identificador posteriormente toma el conjunto de reglas y comienza por ver si la primera del conjunto tiene hijos, en caso afirmativo se llama recursivamente a la función *evalua_Y_O(...)*

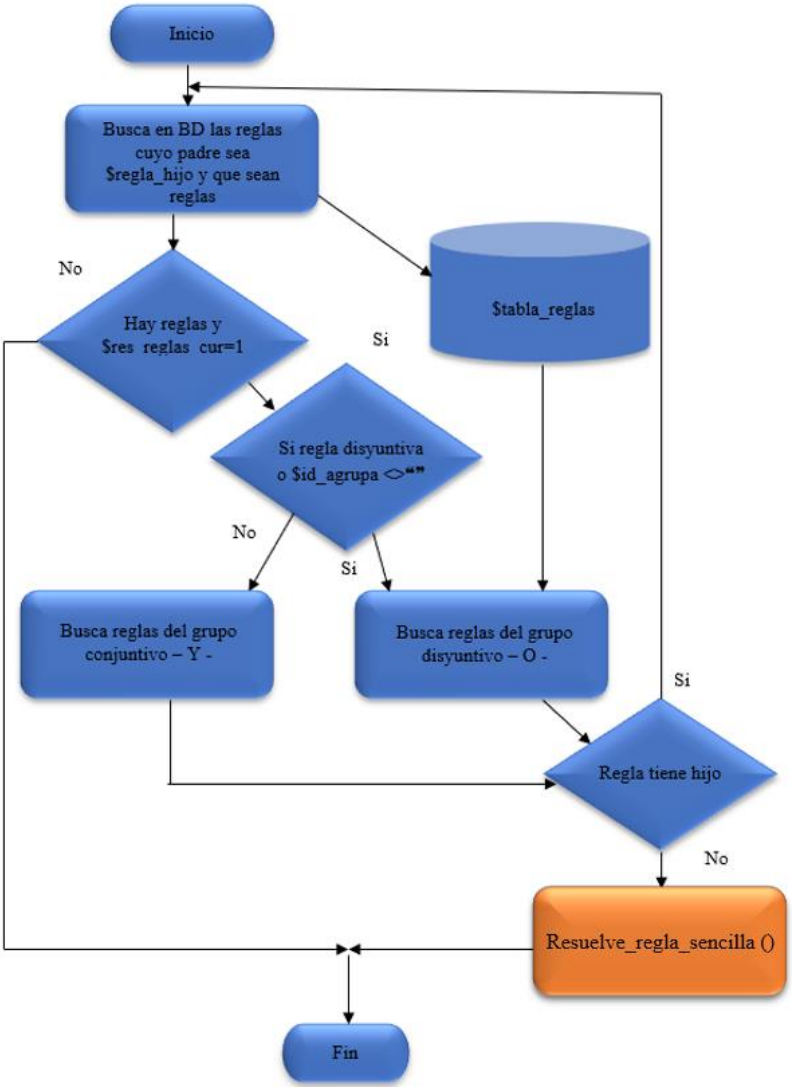

Ilustración 8. Diagrama de flujo de evalua_Y_O(Ci).
Fuente: Elaboración propia.

This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Caso contrario llama a la función *resuelve_regla_sencilla()* ilustración 9, por otro lado, en caso de que no haya sido una regla disyuntiva, la atiende como regla conjuntiva, después ve si tiene hijos, si es afirmativo se llama recursivamente a la función evalua_Y_O (…), caso contrario llama a la función *resuelve_regla_sencilla()* ilustración 9, como sucedió con las reglas disyuntivas, de esta forma puede explorar en profundidad el árbol de conocimiento o BC.

Ahora se procede a explicar la función *resuelve_regla_sencilla (…)* presentada en la ilustración 9. Para la evaluación de cada regla del conjunto $R_h \epsilon$ hijo (R_i), esta función es cíclica y en resumen toma una regla y la va sintetizando, dicho de otra forma, en cada ciclo resuelve algo y lo reduce hasta el punto en que solo queda cierto o falso, en ese contexto se explica a detalle.

De la ilustración 9, continua si el largo de la cadena es mayor que 1 lo que quiere decir que hay algo que procesar o sintetizar, además que la bandera *$continua_proceso* sea verdadera o 1, caso contrario regresa *$regla_valor* y termina la función con el resultado, si continua llama la función *descompone regla($regla valor)* y vacía la regla en forma de cadena de texto al arreglo bidimensional *$arr_regla_curso [] []*, que en si descompone la regla elemento por elemento.

Posteriormente llama a la función *Busca elem en vars(…)* cuya función es determinar los elementos de la regla que se identifican como variables y que hayan sido proporcionadas por el usuario a través de la cadena *$condicion*, luego analiza en otra condicional si puede continuar y que todos los elementos hayan podido ser identificados o que se haya determinado la naturaleza del elemento para su posterior tratamiento. La naturaleza de los elementos que se usan para construir reglas se muestra en la tabla 2.

Tabla 2.
Tipos de datos en los elementos de reglas.

Tipo	Descripción
A	Agrupador: (,)
C	Manejo de Conjuntos. Está contenido en …
CB	Es un campo de una tabla en una base de datos
Co	Define constantes
E	Números Entero
ES	Bases de datos (Esquemas)
F	Función implementada en una en lenguaje PHP
FB	Función a con acceso a base de datos
L	Operadores lógicos: disyunción: OR | ó conjunción AND &
OB	Operador para bases de datos para fijar un rango de búsqueda, en el entorno de SQL: BETWEEN
R	Operador de comparación: >, <
S	Separador
T	Representa una tabla en una base de datos
V	Variables proporcionadas por el usuario
Vig	Vigencia de una regla, esta no requiere datos externos

Fuente: Elaboración propia.

Las reglas pueden ser compuestas y se distinguen tres tipos básicos:

- Sección de reglas relacionadas a bases de datos.
- Sección de regla relacionada a operaciones aritméticas
- Sección de reglas con operador relacional entre dos variables, o variable y constante por ejemplo<, >. <>, <=, >=

Con esta propuesta se pueden construir reglas sencillas, pero también reglas más elaboradas para ello se sigue un criterio de evaluación y resolución en forma de árbol, resuelve de forma recursiva, aunque su implantación es con simples ciclos, la reducción de la regla, sigue un orden de precedencia para ir trabajar las operaciones anidadas, empezando por operaciones que devuelvan un valor, tal es el caso de las búsquedas a bases de datos, después el cálculo de funciones implantadas en sistema que igual que las

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

operaciones a bases de datos devuelven solo un valor, después contempla las operaciones aritméticas dado que ya no existen variables y al final la comparación lógica para determinar su veracidad o falsedad de la regla.

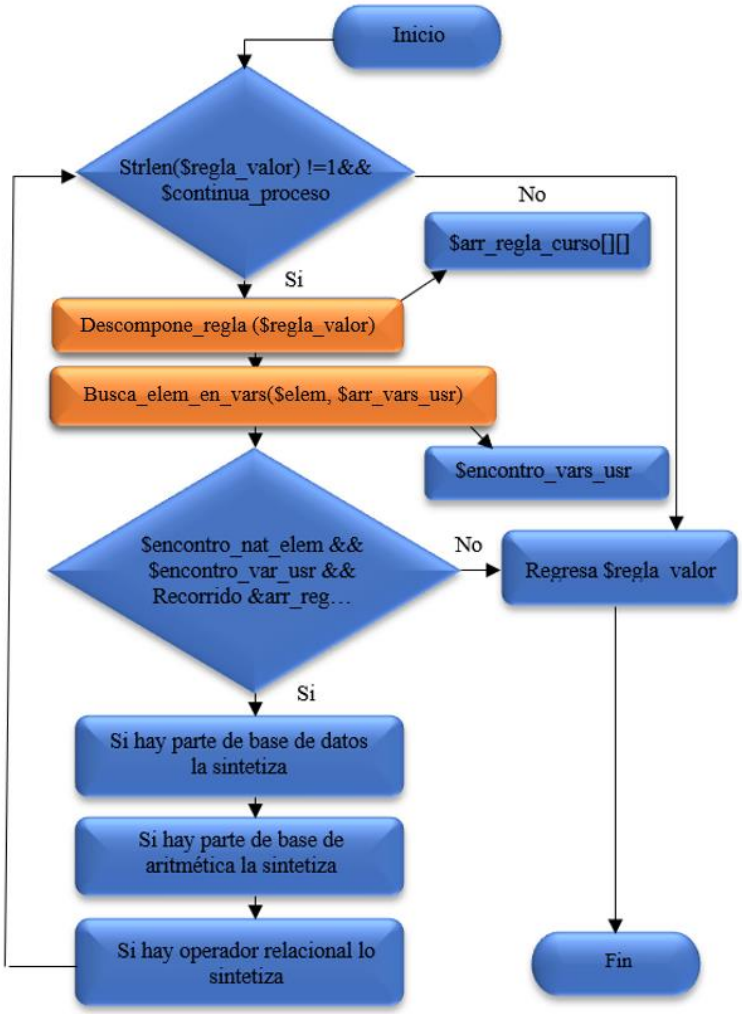

Ilustración 9. Diagrama de flujo de resuelve_regla_sencilla(...).
Fuente: Elaboración propia.

Ahora se procede a explicar los tipos básicos de reglas.

Reglas relacionadas a bases de datos.

Estos tipos de reglas son las más complicadas porque toma la regla que no es SQL estándar y lo convierte a SQL estándar, que hasta el momento puede ser ejecutado por MySQL en sus versiones recientes a la elaboración de este artículo. A continuación, se describe como se realiza este proceso.

Se contemplan dos subtipos:

- Consultas sencillas a BD
- Consultas en las que se involucra funciones de grupo: count(campo de tabla), sum(campo de tabla), max(campo), avg(campo de tabla)

El código SQL se construye a partir de la idea de fragmentar el código acorde al SQL estándar por ejemplo una consulta sencilla se puede construir de secciones constantes y otras variables como se puede ver a continuación:

```
$consulta = "SELECT $componentes_Sel FROM $tabla";
```


This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Se puede ver que las palabras reservadas *SELECT* y *FROM* permanecerán en las consultas, y $componentes_Sel y $tablas, son variables que se les podrá asignar valores durante el proceso.

En el mismo código se puede apreciar que se está construyendo una cadena que posteriormente será enviada al manejador de bases de datos.

Siendo esta una cadena se puede construir dinámicamente por ejemplo se le puede agregar un condicional a la primera cadena como se ve a continuación:
if(!empty($condi))

$consulta .= " WHERE $condi";

Las consultas pueden generar respuestas de dos tipos, como se ve en los ejemplos siguientes:

SELECT * FROM antilavado.registroop WHERE id_cuenta = 2
SELECT SUM(campo1) FROM antilavado.tabla WHERE id_cuenta = 2;

Para el caso 1 se ha considerado solo decir si es cierto (1) o falso (0), Sin embargo, para el caso 2. Regresa un valor numérico ∈ R.

El problema radica en que las reglas que contienen sección de base de datos pueden regresar 1 o 0, otro regresa un valor para ingresarse en una cadena.

Reglas relacionadas a operaciones aritméticas.

Esta parte procesara reglas o secciones de reglas como:

Var1 + 4 – Var2, u otras operaciones contenidas en op={+, -, *, /}

Reglas con operador relacional.

Para las reglas de comparación o con operador relacional se usa un mecanismo de que simplifica reglas al simple caso de: a operador b, tal que operador = {<, >, <=, >=, <>, =} y en la evaluación solo regresa cierto (1) si se cumple la condición o falso (0) en caso contrario.

A continuación, se presenta la forma de procesamiento y sintetizado de una regla compuesta.

Considere la siguiente regla:

dias_entran + SUMA (dias E num_empleado & tpo_categ & anio_aplica & con_clave C BDincidencias.faltas) < 13

En este caso primero ejecuta la consulta base de datos, *SUMA (dias E num_empleado & tpo_categ & anio_aplica & con_clave C BDincidencias.faltas)* que dice: Suma los días que pertenecen al esquema, tabla BDincidencias.faltas.

- C – Representa que se debe encontrar
- !C Representa que no se debe encontrar.

Luego *Variables usuario={num empleado, tipo categ, anio aplica, con clave},* son reconstruidos como condiciones de SQL con los valores pasados por el usuario.

Suponga que el resultado devuelto por el manejador de reglas ha sido 5, entonces ese valor es reemplazado por toda la cadena que lo genero como se ve a continuación:

dias_entran + 5 < 13

Después tal como se ve en la ilustración 8 se vuelve a llamar la función de forma cíclica dado que el tamaño de la regla no es de longitud 1.

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Ahora dado que se cuenta con la naturaleza de días_entran y es de tipo 'V' se le asigna el valor proporcionado por el usuario, suponga que vale 2, entonces al reemplazar queda así:

$$2+5 <13$$

Dado que aún no cumple la condición de longitud de tamaño se vuelve regresa después de descomponer la regla en sus elementos se encuentra por orden de precedencia la opción aritmética y procesa la suma de las dos constantes, y reemplaza como se hizo anteriormente quedando de la siguiente forma.

$$7<13$$

Al no cumplir la condición de longitud de la longitud se regresa y vuelve a ejecutar el proceso, pero ahora solo ingresa en la sección de operador relacional lo que devuelve que es cierto ó 1.

Finalmente, la expresión es verdadera por lo que la función *resuelve_regla_sencilla(...)*, regresa 1.

Conclusiones

A la presentación de este artículo el desarrollo de este proyecto ha llevado 10 años, y ha tenido muchos cambios acordes a las pruebas e implantaciones realizadas, asimismo por el nuevo conocimiento adquirido referente a estructuras de datos e inteligencia artificial.

Esta propuesta en la práctica es viable. Como muestra de dicha viabilidad esta propuesta ha sido implantada en el sistema de incidencias del Instituto Politécnico Nacional en México, así como en los trabajos de titulación: "Sistema experto para la detección de padecimientos de la columna vertebral mediante reconocimiento de imágenes y procesamiento del historial clínico" (Mariscal Avendaño & Mendoza Ruíz, 2021), "Sistema WEB para la administración de incidencias del personal, adaptable a diversos tipos de organización" (Fuentes Arteaga & Miranda Quintero, 2024), "Nephro-healthcare- Aplicación móvil de apoyo para personas con Insuficiencia Renal Crónica" (Solis, 2022), "Sistema experto web aplicado a orientación vocacional para alumnos aspirantes al IPN en nivel Superior" (Orta Cisneros et al., 2022) y en la tesis doctoral titulada "Sistema experto para la detección de transacciones bancarias sospechosas de lavado de dinero" (Sabas González, 2019).

Derivado de la implementación del evaluador de reglas propuesto en un entorno de producción, en el área de recursos humanos en el Instituto Politécnico Nacional en México, con 100 usuarios concurrentes, se obtuvieron los siguientes resultados: Se contabilizaron 514293 evaluaciones de las reglas diversas, incluidas consultas a bases de datos que son las más lentas, con un tiempo promedio de 0.042 segundos por evaluación, lo cual se indica que el tiempo promedio por regla es menor al segundo.

También se puede concluir que esta propuesta soporta muchas reglas, de diferente tamaño y complejidad, aunque lo recomendable es construir reglas sencillas, aunque sean más. También se pudo implantar reglas referentes a la temporalidad y veracidad de las reglas en diferentes entornos.

En trabajos futuros se contempla:

- Disminuir el tiempo de ejecución.
- Crear interfaz gráfica e intuitiva para creación de reglas que conserven la integridad de la BC.
- Ampliar funcionalidad en código SQL.

Referencias bibliográficas

Aguilera-Venegas, G., Roanes-Lozano, E., Rojo-Martínez, G., & Galán-García, J. L. (2023). A proposal of a mixed diagnostic system based on decision trees and probabilistic experts rules. *Journal of Computational and Applied Mathematics, 427*, 1-15. https://doi.org/10.1016/j.cam.2023.115130

Aspin, A. (2022). *Querying MariaDB: Use SQL Operations, Data Extraction, and Custom Queries to Make your MariaDB Database Analytics more Accessible.* USA: BPB Publications.

Behlendorf, B., McCool, R., Fielding, R., & Hartill, R. (2025). *Apache HTTPD*. Obtenido de HTTP Server Project: https://httpd.apache.org/

Font Fernández, J. M. (2008). *Sistemas de representación de conocimiento basados en reglas*. (Tesis de maestría). Universidad Politécnica de Madrid, España.

This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)

Fuentes Arteaga, D., & Miranda Quintero, D. A. (2024). *Sistema WEB para la administración de incidencias del personal, adaptable a diversos tipos de organización.* (Trabajo terminal - Tesis). Escuela Superior de Cómputo, Instituto Politécnico Nacional, Ciudad de México.

Grippa, V., & Kuzmichev, S. (2021). *Learning MySQL: Get a Handle on Your Data*. Estados Unidos de Norteamérica: O'Reilly.

Hou, B., Xue, M., Liu, J., & Wu, Z. (2025). Multi-output extended belief rule-base system and its parameter learning schemes. *Applied Soft Computing*, *170*, 1-20, https://doi.org/10.1016/j.asoc.2024.112687

Laird, J. (2025). *The Soar cognitive architecture.* Recuperado de https://web.eecs.umich.edu/~soar/ijcai16/Tutorial-2016-basic.pdf

Lerdof, R., Suraski, Z., & Gutmans, A. (2025). *PHP*. Obtenido de PHP: https://www.php.net/

Lyndon Johnson Space Center (NASA). (2025). *CLIPS Rule Based Programming Language*. Obtenido de https://sourceforge.net/projects/clipsrules/

Mariscal Avendaño, I., & Mendoza Ruíz, G. (2021). *Sistema experto para la detección de padecimientos de la columna vertebral mediante reconocimiento de imágenes y procesamiento del historial clínico.* (Trabajo terminal - Tesis). Escuela Superior de Cómputo, Instituto Politecnico Nacional. Ciudad de México.

Martinez-Llario, J., Coll, E., Núñez-Andrés, M., & Femenia-Ribera, C. (2017). Rule-based topology system for spatial databases to validate complex geographic datasets. *Computers & Geosciences*, 103, 122-132. https://doi.org/10.1016/j.cageo.2017.03.013

Orta Cisneros, S., Cisneros Palacios, J. C., & Sandoval Lluvias, R. D. (2022). *Sistema experto web aplicado a orientación vocacional para alumnos aspirantes al IPN en nivel Superior.* (Trabajo terminal - Tesis). Escuela Superior de Cómputo, Instituto Politécnico Nacional. Ciudad de México.

Pawuś, D., Paszkiel, S., & Porażko, T. (2025). Expert system supporting automatic risk classification and management in idiopathic membranous nephropathy based on rule sets and machine learning. *Biomedical Signal Processing and Control, 109,* 1-21. https://doi.org/10.1016/j.bspc.2025.107989

Proctor, M., Verlaenen, K., & Tirelli, E. (2025). *Drools*. Obtenido de Drools: https://www.drools.org/

Ramirez Romero, T. A. (2013). *Sistema computacional para la administración y operación de reglas de diverso origen, implantado sobre bases de datos abierta.* Ciudad de México: Instituto Politecnico Nacional, Tesis doctoral. Recuperado de https://acortar.link/uj9ay0

Sabas González, J. F. (2019). *Sistema experto para la detección de transacciones bancarias sospechosas de lavado de dinero.* (Tesis doctoral). Ciudad de México: Escuela Superior de Ingeniería Mecánica y Eléctrica, Instituto Politécnico Nacional.

Janes, M., & Garcia, N. (2025). *Sandia Labs news*. Obtenido de Jess lives: Latest version of popular productivity-boosting software tool is released for licensing. Recuperado de Sandia National Laboratories. https://www.sandia.gov/labnews/2006/12/08/061208-3/

Santander - Open Academy. (2023). *Sistemas Expertos: el impulso de la Inteligencia Artificial* . Obtenido de https://www.santanderopenacademy.com/es/blog/sistemas-expertos.html

Smirnova, S., & Tezuysal, A. (2022). *MySQL Cookbook: Solutions for Database Developers and Administrators*. Estados Unidos: O'Reilly.

Solis, A. (2022). *"Nephro-healthcare" Aplicación móvil de apoyo para personas con Insuficiencia Renal Crónica.* (Trabajo terminal - Tesis). Escuela Superior de Cómputo, Instituto Politécnico Nacional. Ciudad de México.

Tchuente, D., Lonlac, J., & Kamsu-Foguem, B. (2024). A methodological and theoretical framework for implementing explainable artificial intelligence (XAI) in business applications. *Computers in Industry*, *155,* 1-10. https://doi.org/10.1016/j.compind.2023.104044

Wang, W., Yang, M., & Seong, P. H. (2016). Development of a rule-based diagnostic platform on an object-oriented expert system shell. *Annals of Nuclear Energy, 88*, 252-264. https://doi.org/10.1016/j.anucene.2015.11.008

Widenius, M. (2025). *MariaDB Foundation*. Obtenido de MariaDB: https://mariadb.org/

Widenius, M., Axmark, D., & Larsson, A. (2025). *MySQL*. Obtenido de MySQL: https://www.mysql.com/

Zolfagharnasab, M. H., Damari, S., Soltani, M., & Ng, A. (2025). A novel rule-based expert system for early diagnosis of bipolar and Major Depressive Disorder. *Smart Health, 35*, 1-13. https://doi.org/10.1016/j.smhl.2024.100525

This article presents no conflicts of interest. This article is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). Reproduction, distribution, and public communication of the work, as well as the creation of derivative works, are permitted provided that the original source is cited.

Creative Commons Attribution 4.0 International (CC BY 4.0)