UNIDAD DE APRENDIZAJE
DE SISTEMAS EXPERTOS

CRÉDITOS INSTITUCIONALES: 5

MANUAL DE PRÁCTICAS DE LABORATORIO: SISTEMAS EXPERTOS

PROGRAMA DE INGENIERÍA EN COMPUTACIÓN

CU UAEM VALLE DE CHALCO

AUTOR:
MARCO ALBERTO MENDOZA PÉREZ

PERIODO DE REALIZACIÓN:
FEBRERO - JULIO DE 2017

ÍNDICE DE CONTENIDO

PRESENTACIÓN

Las presentes prácticas de laboratorio fueron desarrolladas en apego al programa de la Unidad de Aprendizaje de Sistemas Expertos, con la finalidad de que los alumnos puedan poner en práctica los conocimientos teóricos que se van adquiriendo a lo largo del curso. Esta Unidad de Aprendizaje consta de la siguiente estructura:

1.- Conocer los conceptos básicos de la Inteligencia Artificial.
2.- Manejar la representación del conocimiento.
3.- Computación evolutiva.
4.- Algoritmos bio-inspirados.
5.- Procesos de decisión de Markov y aprendizaje por refuerzo.
6.- Agentes inteligentes y sistemas multi-agentes.

Este manual consta de 3 prácticas, cuya complejidad es acorde al tema que abarca. En este sentido estas prácticas pertenecen a los temas: 2.5.- Lógica proposicional y 2.6.- Lógica de predicados.

Cada práctica consta de Titulo, Objetivos, Introducción, Material y Equipo a Utilizar, Duración, Desarrollo y Evaluación. El alumno debe replicar de forma práctica lo que se encuentra en el apartado de Desarrollo y resolver lo que se pide en el apartado de Evaluación.

De cada práctica, el alumno deberá elaborar su Reporte de práctica correspondiente, el cual lo debe de entregar de forma: digital (infografía o artículo) o impresa (infografía o artículo), con los siguientes puntos: Carátula, Resumen (máx. 150 palabras) y Palabras clave (máx. 3 y ordenadas de forma ascendente), Introducción, Objetivos (General y Específicos), Marco Referencial, Desarrollo (diagramas y evidencias originales con explicación), Resultados (evidencias originales con explicación), Conclusiones Individuales y Bibliografía/Referencias Electrónicas (ambas en formato APA).

PRÁCTICA 1

PROGRAMACIÓN DE ENUNCIADOS DE LÓGICA PROPOSICIONAL Y DE LÓGICA DE PREDICADOS EN PROLOG

OBJETIVOS:

Aprender a programar enunciados de lógica proposicional y de predicados para poder obtener resultados (deducciones, predicciones o conclusiones).

INTRODUCCIÓN:

Se va a hacer uso de los siguientes conceptos:

a) Conocimiento: Es un conjunto de información almacenada mediante la experiencia, el aprendizaje o la introspección (inspección interna). El conocimiento tiene su origen en la percepción sensorial, después llega al entendimiento y concluye en la razón. Se dice que el conocimiento es una relación entre un sujeto y un objeto.

b) Lógica: Lo que es congruente, ordenado y bien estructurado. Es la ciencia del pensamiento y la razón.

c) Proposición: Es una afirmación que comunica una idea verdadera o falsa. De los siguientes ejemplos, establecer el valor de verdad de las siguientes proposiciones:

- q: España es el campeón mundial de soccer del 2010.
  Esta proposición es verdadera, pues España gano la final de soccer del 2010.
- r: Junio es el quinto mes del año.
  La proposición es falsa. Al enumerar los meses se puede apreciar que junio es el sexto mes del año.
- s: 2 elevado a la 3 es 8.
  $2^3 = 8$, la proposición es verdadera.

d) Lógica Proposicional: Es la parte de la lógica que estudia la formación de proposiciones complejas a partir de proposiciones simples y la inferencia de proposiciones a partir de proposiciones, pero sin tener en cuenta la estructura interna de las proposiciones más simples. Las proposiciones equivalen a frases u oraciones del lenguaje hablado.

e) La lógica de Predicados, también llamada lógica de primer orden, es una extensión de la lógica proposicional que usa variables para los objetos. Esta nos permite dar una descripción de la realidad más detallada.

f) Programación Lógica: Constituye una herramienta que tradicionalmente ha provisto de una sólida estructura conceptual para representar conocimiento. Puede ser usada para describir y recuperar conocimiento sobre un dominio de aplicación y para describir tareas como la capacidad de encontrar solución a problemas que normalmente se consideran dentro del campo de la Inteligencia Artificial. Esta es considerada una programación por descripción. El programa se construye describiendo el área de aplicación, esto es: se señala el qué se desea (mediante hechos) pero no el cómo obtenerlo, esto está implícito.

g) Programa Lógico: Se configura como un conjunto de hechos (proposiciones) y de reglas lógicas previamente establecidas, que obtienen conclusiones en base a una serie de preguntas lógicas. Este se transforma en un conjunto de declaraciones formales de especificaciones que deben ser correctas por definición.

h) Sistema de Programación Lógica: El conjunto de hechos y reglas constituyen una descripción, que llega a ser un programa donde combinado con un procedimiento de inferencia independiente de la aplicación hace posible a la computadora sacar conclusiones acerca del área de aplicación y responder preguntas aún cuando estas respuestas no estén explícitamente registradas en la descripción.

i) Prolog: Proviene de PROgrammation en LOGique (programación en lógica), que fue el primer lenguaje de programación lógico y el más conocido y utilizado. Este lenguaje fue desarrollado por el Grupo de Inteligencia Artificial de la Universidad de Marseille, dirigido por Alain Colmerauer, en 1972. Prolog es utilizado para el desarrollo de aplicaciones de inteligencia artificial debido a su forma de representar el conocimiento, facilitando las búsquedas en bases de datos, la escritura de compiladores, la construcción de sistemas expertos, el procesamiento de lenguaje natural y la programación automática.
   Decimos que Prolog es declarativo porque no es imperativo. Es decir, cada “línea de programa” es una declaración, no una orden.

Los predicados en Prolog los llamaremos hechos. Debemos tener en cuenta que:

a) Los nombres de todos los objetos y relaciones deben comenzar con una letra minúscula.
b) Existen 2 tipos de predicados: monádicos (propiedades), como por ejemplo: mujer(jaqueline) y poliádicos (relaciones), como por ejemplo: tiene(henry, dinosaurio).

c) Los objetos, entidades o conceptos se escriben separándolos mediante comas y encerrados entre paréntesis. Son los argumentos.
d) Al final del hecho debe ir un punto ".".

La estructura de un hecho es la siguiente: predicado (arg_1, arg_2, …, arg_n) o relación (objeto_1, objeto_2, ..., objeto_n).

La estructura de un programa en Prolog es:
a) Declarar hechos sobre los objetos y sus relaciones.
b) Definir reglas sobre los objetos y sus relaciones.
c) Hacer consultas sobre los objetos y sus relaciones.

La Tabla 1.1, nos muestra una interpretación de operadores lógicos a su sintaxis en Prolog.

Operador lógico	**Sintaxis en Prolog**
AND (conjunción y)	, (coma)
OR (disyunción o)	; (punto y coma)
NOT (negación no)	\+
IF (implicación si)	:- (dos puntos y guion)

Tabla 1.1 Operadores con clausulas en Prolog.

MATERIAL Y EQUIPO A UTILIZAR:

Para la realización de esta práctica son necesarios los siguientes componentes:

Equipo:
- Computadora.

Software:
- SWI-Prolog.
- Canva.

Material:
- Práctica digital o impresa.

DURACIÓN:
- 120 minutos.

DESARROLLO:

A continuación, se presentan una serie de programas escritos en el lenguaje de programación lógico Prolog:

En la Figura 1.1, se muestra el código en Prolog para los siguientes enunciados: "El auto arranca si su batería funciona y su motor enciende" y "El auto no arranca si su batería funciona y su motor no enciende". Está escrito en el editor de la aplicación de código abierto SWI-Prolog, se compila y se guarda con extensión .pl.

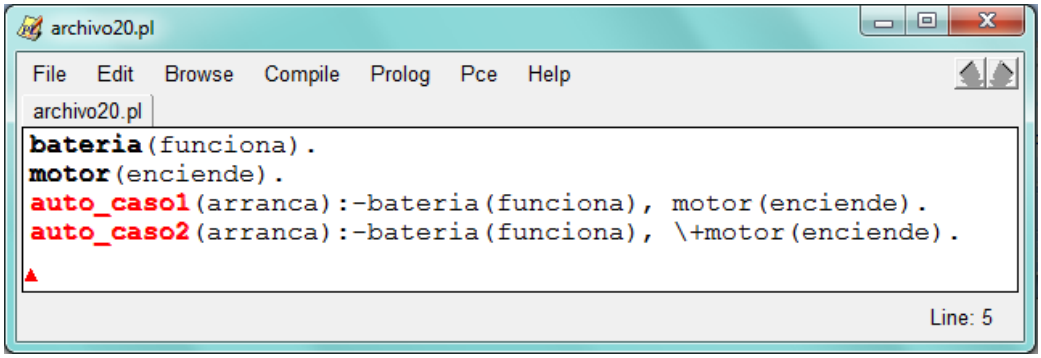


Figura 1.1 Programa con predicados verdaderos y falsos.

La Figura 1.2, nos muestra las consultas correspondientes del programa anterior.

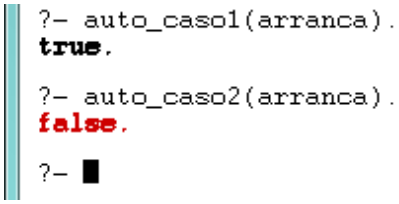


Figura 1.2 Consultas de las 2 reglas del programa anterior.

En la Figura 1.3, se muestra un programa en Prolog con 3 hechos y 1 regla para el siguiente enunciado: "Alberto le gustan los vehículos de la marca Ford".

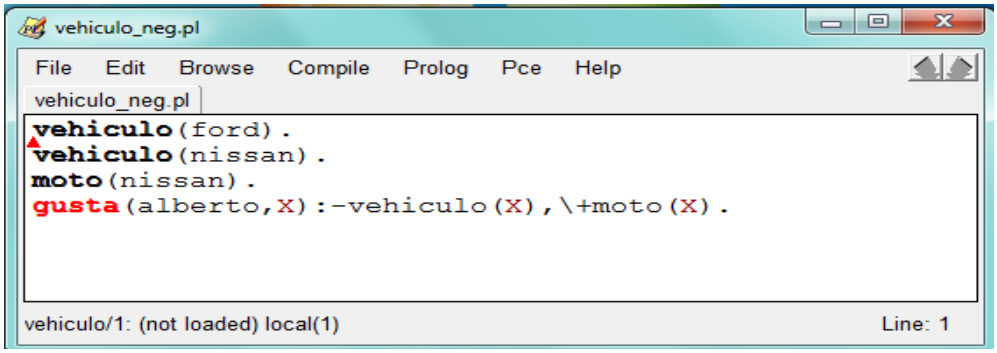


Figura 1.3 Programa con hechos y reglas en Prolog.

La Figura 1.4, nos muestra algunas consultas que se pueden ejecutar del programa anterior.

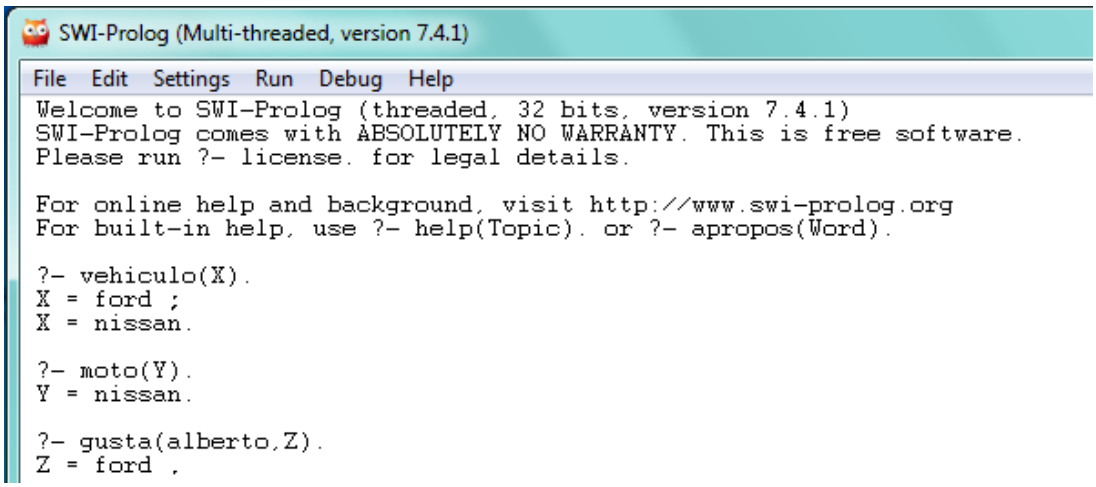


Figura 1.4 Consultas a un programa escrito en Prolog.

En la Figura 1.5, se muestra un programa con 2 hechos y 2 reglas para calcular recorridos cortos o largos.

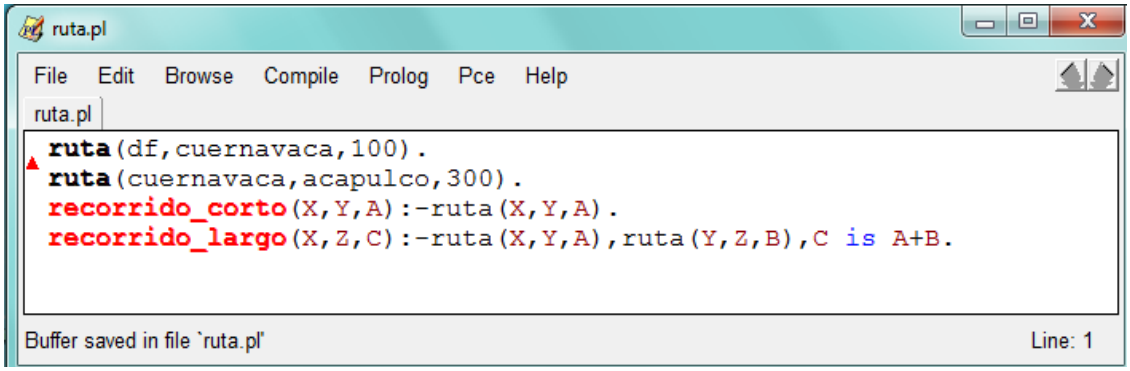


Figura 1.5 Programa de recorridos cortos o largos.

La Figura 1.6, nos muestra las consultas correspondientes del programa anterior.

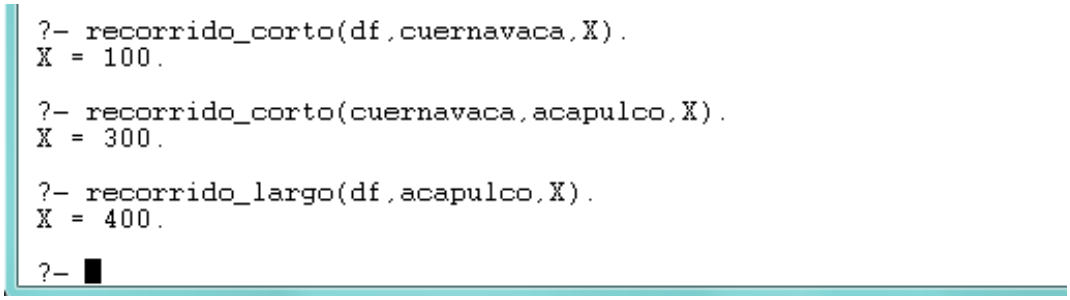


Figura 1.6 Consultas de recorridos.

En la Figura 1.7, se muestra el código en Prolog de un circuito con compuertas lógicas. A partir de la regla que genera el circuito, dibújalo y coloca su tabla de verdad.

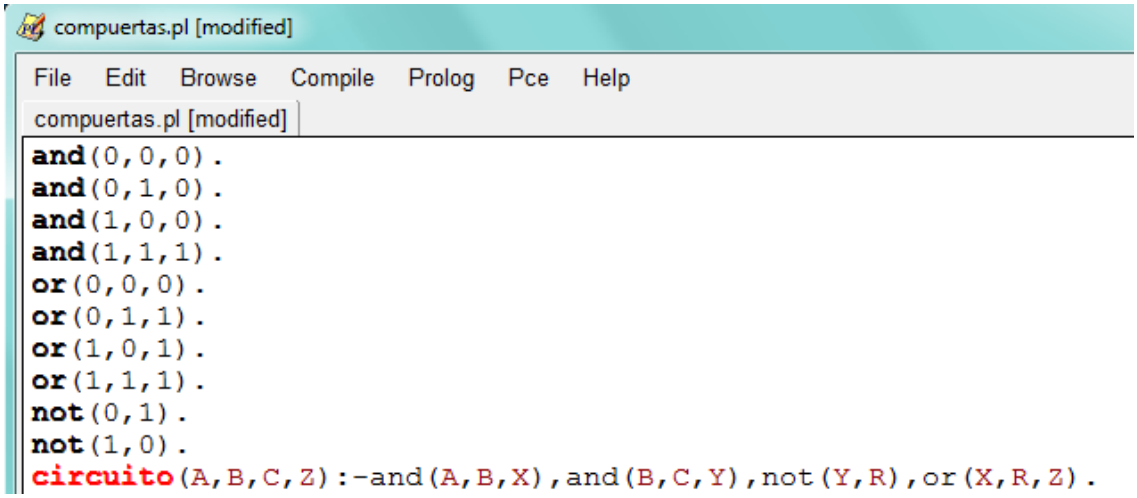


Figura 1.7 Programa de un circuito con compuertas lógicas.

La Figura 1.8, nos muestra las combinaciones del circuito del programa anterior.

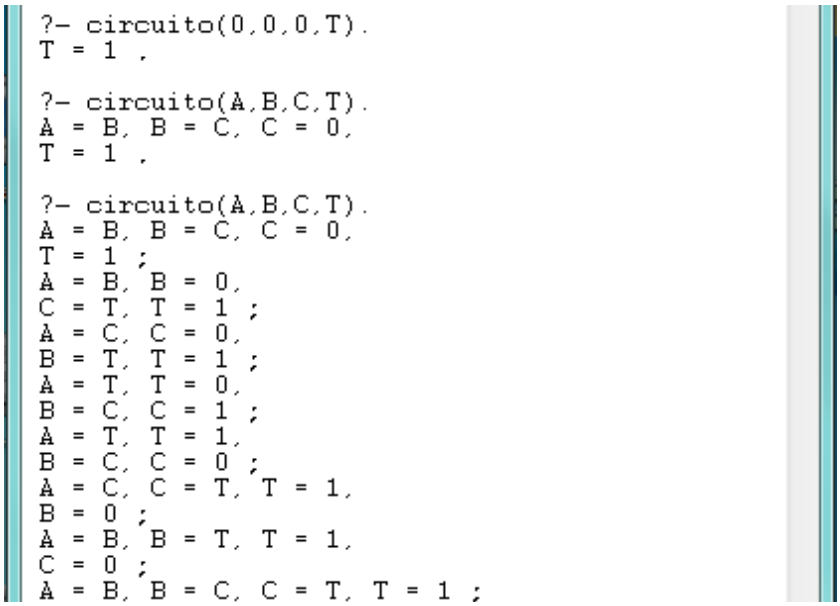


Figura 1.8 Consultas del programa anterior.

EVALUACIÓN:

De los siguientes enunciados, se te pide que elabores su programa correspondiente; el cual deberá contener hechos, reglas y consultas en SWI-Prolog.

1.- De los siguientes enunciados, realiza la lógica de predicados con sus respectivas consultas:

Base de conocimiento:

Regla 1: Si está contento entonces escucha música.

Regla 2: Si tiene radio entonces escucha música.

Regla 3: Si escucha música y tiene una guitarra entonces toca la guitarra.

Hecho 1: Tiene una guitarra.

Hecho 2: Esta contento.

Consultas:

> Esta tocando la guitarra?

> Quien está contento?

> Quien escucha música?

> Esta contento?

2.- De la Figura 1.9, realiza el programa con sus consultas y construye su tabla de verdad.

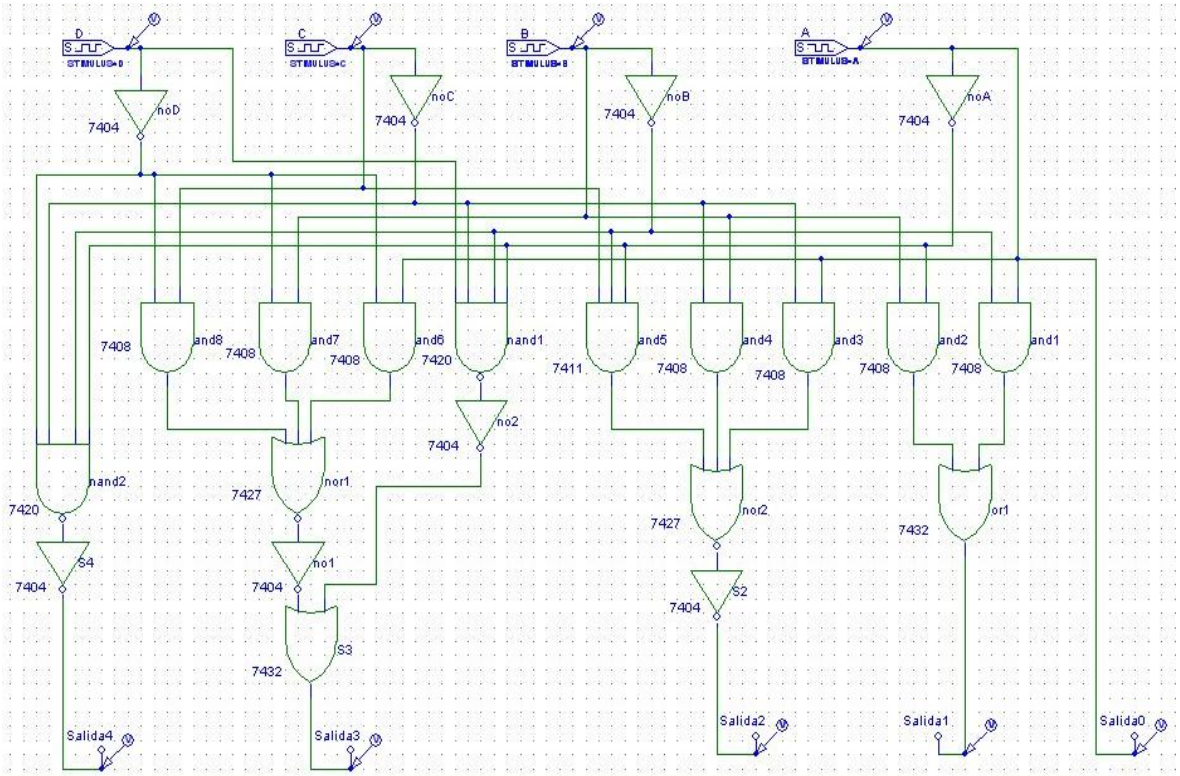


Figura 1.9 Circuito combinacional con compuertas lógicas.

PRÁCTICA 2

PREDICADOS PREDEFINIDOS Y DEDUCCIONES EN PROLOG

OBJETIVOS:

Aprender a programar, utilizando predicados predefinidos en Prolog junto con hechos y reglas para así obtener deducciones de forma lógica, que no son más que los resultados de las preguntas que elaboramos de acuerdo al problema a resolver.

INTRODUCCIÓN:

Una de las ventajas de la programación lógica es que se especifica qué se tiene que hacer (programación declarativa), y no cómo se debe hacer (programación imperativa).

La Tabla 2.1, nos muestra los diferentes tipos de predicados predefinidos que se pueden utilizar en Prolog.

Construcción de objetivos compuestos (conectivas lógicas)	
X,Y	Conjunción de objetivos.
X;Y	Disyunción de objetivos.
\+X	Se cumple si fracasa el intento de satisfacer X.
Control del programa	
true	Objetivo que siempre se cumple.
Fail	Objetivo que siempre fracasa.
!	Corte. Obliga al sistema Prolog a mantener ciertas elecciones.
Operadores aritméticos y relacionales	
X = Y	Se cumple si puede hacer X e Y iguales (unificarlos).
X \= Y	Se cumple si X=Y fracasa.
X == Y	Igualdad más estricta.
X \== Y	Se cumple si fracasa ==.
X < Y	Predicado menor que.
X > Y	Predicado mayor que.
X >= Y	Predicado mayor o igual que.
X =< Y	Predicado menor o igual que.
X is Y	Se evalúa la expresión a la que esta instanciada Y para dar como resultado un número, que se intentará hacer coincidir con X.
X + Y	Operador suma.
X – Y	Operador resta.
X * Y	Operador de multiplicación.

X / Y	Operador de división.
X // Y	Operador de división entera.
X mod Y	Operador de resto de la división entera.
X ** Y	Operador de exponenciación.
X ^ Y	Operador de exponenciación.
Funciones aritméticas	
abs(X)	Devuelve el valor absoluto de la expresión X.
sign(X)	Devuelve -1 si X < 0, 1 si X > 0 y 0 si X = 0.
min(X,Y)	Devuelve el menor de X e Y.
max(X,Y)	Devuelve el mayor de X e Y.
random(X)	Devuelve un entero aleatorio i (0 =< i < X); es determinada por el reloj del sistema cuando se arranca SWI-Prolog.
round(X)	Evalúa la expresión X y la redondea al entero más cercano.
integer(X)	Evalúa la expresión X y la redondea al entero más cercano.
float(X)	Evalúa la expresión X en un número en coma flotante.
floor(X)	Devuelve el mayor número entero menor o igual que el resultado de la expresión X.
ceiling(X)	Devuelve el menor número entero mayor o igual que el resultado de la expresión X.
sqrt(X)	Raíz cuadrada de la expresión X.
sin(X)	Seno de la expresión X (ángulo en radianes)
cos(X)	Coseno de la expresión X (ángulo en radianes).
tan(X)	Tangente de la expresión X (ángulo en radianes).
asin(X)	Arco seno (ángulo en radianes) de la expresión X.
acos(X)	Arco coseno (ángulo en radianes) de la expresión X.
atan(X)	Arcotangente (ángulo en radianes) de la expresión X.
log(X)	Logaritmo neperiano de la expresión X.
log10(X)	Logaritmo en base 10 de la expresión X.
exp(X)	e elevado al resultado de la expresión X.
pi	Constante matemática pi (3.141593).
e	Constante matemática e (2.718282).
Manejo de listas	
[X|Y]	X es la cabeza de Y la cola de la lista.
append(X,Y,Z)	Z es la concatenación de las listas X e Y.
member(X,Y)	X es uno de los elementos de la lista Y.

delete(X,Y,Z)	Borra el elemento Y de la lista X y da como resultado la lista Z.
select(X,Y,Z)	Selecciona el elemento X de la lista Y y da como resultado la lista Z.
last(X,Y)	X es el último elemento de la lista Y.
reverse(X,Y)	Y es la lista invertida de X.
length(X,Y)	Y es la longitud de la lista X.
merge(X,Y,Z)	Siendo X e Y listas ordenadas, Z es la lista ordenada con los elementos de ambas.
sort(X,Y)	Y es la lista ordenada de X (sin elementos duplicados).
msort(X,Y)	Y es la lista ordenada de X.
Predicados de Entrada y Salida	
read(X)	Lee el siguiente término del canal de entrada activo.
nl	Genera una nueva línea en el canal de salida activo.
tab(X)	Escribe X espacios en el canal de salida activo.
write(X)	Escribe el término X en el canal de salida activo.
write_ln(X)	Escribe el término X en el canal de salida activo y salta la línea.

Tabla 2.1 Predicados predefinidos en Prolog.

Una deducción es una conclusión o inferencia a la cual se llega gracias a la puesta en práctica de un método de razonamiento el cual partirá de conceptos generales o principios universales para llegar a las conclusiones particulares.

La inferencia es la acción y efecto de inferir que quiere decir deducir algo, sacar una consecuencia de otra cosa o conducir a un resultado. Esta surge a partir de una evaluación mental entre distintas expresiones que, al ser relacionadas como abstracciones, permiten trazar una implicación o condición lógica.

Un motor de inferencia se encarga de realizar las búsquedas en la base del conocimiento de acuerdo con los parámetros definidos en la heurística del sistema y que permitirá inferir la solución de un problema o llevar a cabo la toma de decisiones.

La heurística es la capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines. La capacidad heurística es un rasgo característico de los humanos, basada en la experiencia desde cuyo punto de vista puede describirse como el arte y la ciencia del descubrimiento y de la invención o de resolver problemas mediante la creatividad y el pensamiento lateral o pensamiento divergente.

Un sistema experto es un sistema informático que incorpora, en forma operativa, el conocimiento de una o varias personas experimentadas de cierta área, de forma que es capaz de proporcionar respuestas y explicar los mecanismos de razonamiento que ha utilizado para llegar hasta ellas, así como de modificar o ampliar su conocimiento.

MATERIAL Y EQUIPO A UTILIZAR:

Para la realización de esta práctica son necesarios los siguientes componentes:
Equipo:

- Computadora.

Software:

- SWI-Prolog.
- Canva.

Material:

- Práctica digital o impresa.

DURACIÓN:

- 240 minutos.

DESARROLLO:

A continuación, se presentan una serie de programas escritos en el lenguaje de programación lógico Prolog:

En la Figura 2.1, se muestra un programa con reglas, las cuales contienen operaciones aritméticas básicas con predicados predefinidos de Prolog.

archivo11.pl

File Edit Browse Compile Prolog Pce Help

archivo11.pl

```
suma(A, B, C):-C is A + B.
sumar_3_y_duplicar(X, Y) :-Y is(X + 3) * 2.
sumar(Y):-Y is(10 + 3) * 2.
valor_pi(X,Y):-Y is pi * X.
valor(X,Y,Z):- Z is X mod Y.
absoluto(X,Y):-Y is abs(X)-1.
aleatorio1(Y):-Y is random(5).
aleatorio2(X,Y):-Y is random(X)+1.
```

Figura 2.1 Operaciones aritméticas básicas y predicados predefinidos de Prolog.

La Figura 2.2, nos muestra el código para que el usuario pueda introducir datos desde el teclado junto con las consultas a las reglas del programa anterior.

```
?- write(introducir_datos_ ),read(X),read(Y),suma(X,Y,Z).
introducir_datos_ 5.
|: 8.

X = 5,
Y = 8,
Z = 13.

?- sumar_3_y_duplicar(9,Y).
Y = 24.

?- sumar(X).
X = 26.

?- valor_pi(10,Z).
Z = 31.41592653589793.

?- valor(81,2,W).
W = 1.

?- absoluto(4,A).
A = 3.
```

Figura 2.2 Salidas del programa anterior.

En la Figura 2.3, se muestra el código en Prolog con 3 hechos y 2 reglas que determinan cuando una persona es millonaria o pobre, basándose en sus saldos de cuenta.

negacion.pl

File Edit Browse Compile Prolog Pce Help

negacion.pl

```
saldo_cuenta(maria,1000).
saldo_cuenta(flora,3000000).
saldo_cuenta(antonio,2000000).
millonario(X):- saldo_cuenta(X, Y), Y > 1000000.
pobre(X) :- \+millonario(X).
```

Figura 2.3 Programa con 3 hechos y 2 reglas.

La Figura 2.4, nos muestra las consultas correspondientes del programa anterior.

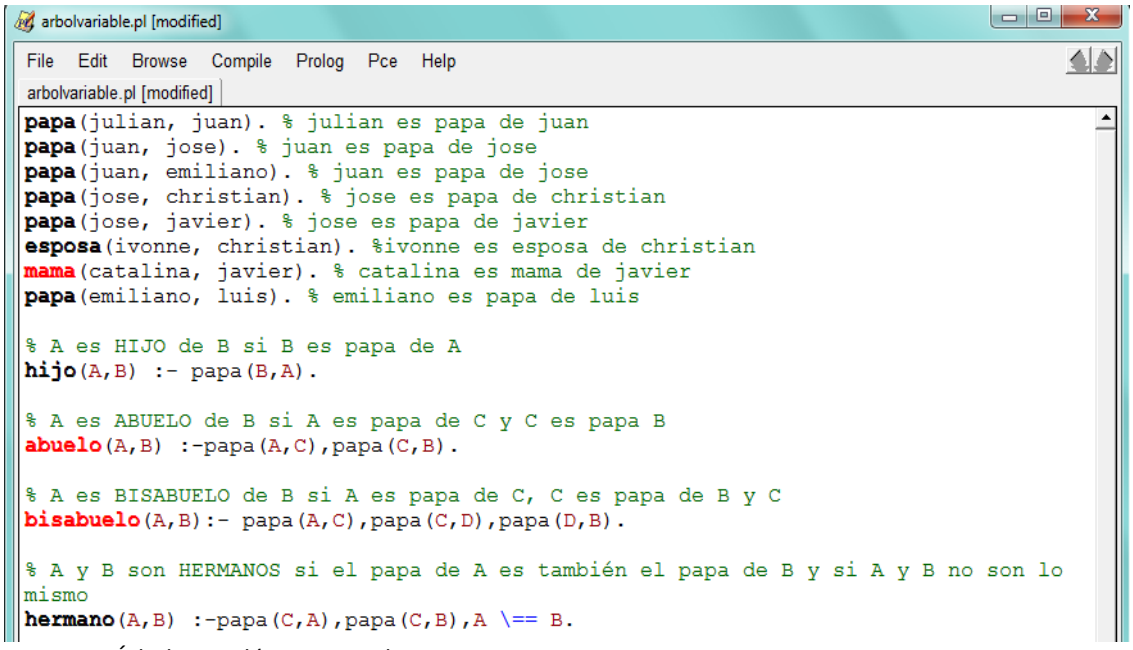
```
?- saldo_cuenta(X,Y).
X = maria,
Y = 1000 ;
X = flora,
Y = 3000000 ;
X = antonio,
Y = 2000000.

?- millonario(X).
X = flora ;
X = antonio.

?- pobre(maria).
true.
```

Figura 2.4 Consultas del programa anterior.

De las Figuras 2.5 y 2.6, se muestra el código en Prolog de un árbol genealógico con hechos, reglas y consultas. A partir de los hechos, podemos deducir por medio de reglas, quienes son hijos, abuelos, bisabuelos y hermanos, etc. De esta figura deberás obtener también: ¿quiénes son primos, tíos y cuñados?




```
papa(julian, juan). % julian es papa de juan
papa(juan, jose). % juan es papa de jose
papa(juan, emiliano). % juan es papa de jose
papa(jose, christian). % jose es papa de christian
papa(jose, javier). % jose es papa de javier
esposa(ivonne, christian). %ivonne es esposa de christian
mama(catalina, javier). % catalina es mama de javier
papa(emiliano, luis). % emiliano es papa de luis

% A es HIJO de B si B es papa de A
hijo(A,B) :- papa(B,A).

% A es ABUELO de B si A es papa de C y C es papa B
abuelo(A,B) :-papa(A,C),papa(C,B).

% A es BISABUELO de B si A es papa de C, C es papa de B y C
bisabuelo(A,B):- papa(A,C),papa(C,D),papa(D,B).

% A y B son HERMANOS si el papa de A es también el papa de B y si A y B no son lo
mismo
hermano(A,B) :-papa(C,A),papa(C,B),A \== B.
```

Figura 2.5 Árbol genealógico en Prolog.

```
?- papa(X,Y).
X = julian,
Y = juan ;
X = juan,
Y = jose ;
X = juan,
Y = emiliano ;
X = jose,
Y = christian ;
X = jose,
Y = javier ;
X = emiliano,
Y = luis.

?- hijo(X,Y).
X = juan,
Y = julian ;
X = jose,
Y = juan ;
X = emiliano,
Y = juan ;
X = christian,
Y = jose ;
X = javier,
Y = jose ;
X = luis,
Y = emiliano.

?- abuelo(X,Y).
X = julian,
Y = jose ;
X = julian,
Y = emiliano ;
X = juan,
Y = christian ;
X = juan,
Y = javier ;
X = juan,
Y = luis ;
```

Figura 2.6 Consultas del Árbol genealógico en Prolog.

En la Figura 2.7, se muestra un programa con una serie de hechos y reglas, que diagnostican y recetan una enfermedad.

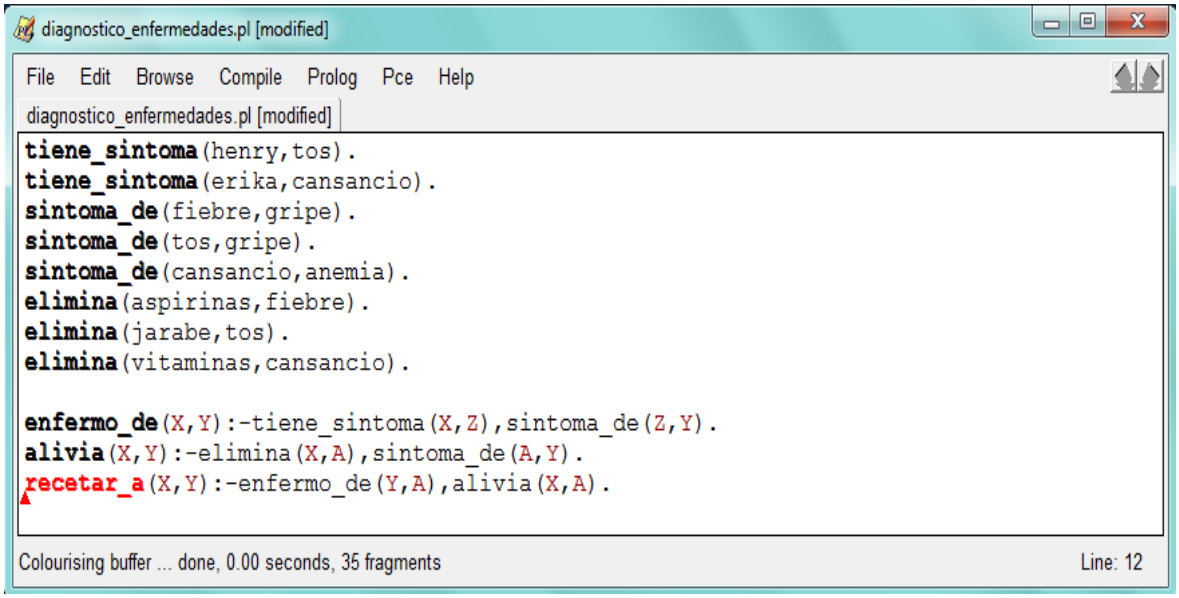


Figura 2.7 Programa realizado en Prolog que sirve para diagnosticar y recetar una enfermedad.

La Figura 2.8, nos muestra la ejecución de las consultas del programa anterior, las cuales nos sirven para conocer de que está enferma la persona y con que se puede aliviar.

```
?- alivia(X,Y).
X = aspirinas,
Y = gripe ;
X = jarabe,
Y = gripe ;
X = vitaminas,
Y = anemia.

?- recetar_a(X,Y).
X = aspirinas,
Y = henry ;
X = jarabe,
Y = henry ;
X = vitaminas,
Y = erika.

?- enfermo_de(X,Y).
X = henry,
Y = gripe ;
X = erika,
Y = anemia.
```
Figura 2.8 Consultas a un programa escrito en Prolog.

EVALUACIÓN:
De los siguientes enunciados, se te pide que elabores su programa correspondiente; el cual deberá contener hechos, reglas y consultas en SWI-Prolog.

1.- El usuario deberá introducir valores para calcular el área de las siguientes figuras geométricas: circulo, esfera, cuadrado y triangulo.

2.- Al usuario le debe permitir introducir sus datos: Matricula, Nombre, Calificación 1er parcial, Calificación 2do parcial. El programa deberá devolver su Promedio y su Situación “aprobado” o “reprobado”.

3.- Calcular el capital total, considerando los siguientes datos introducidos por el usuario: capital invertido, número de años e interés.

4.- En la fábrica de asientos AutoMotor, el pago a sus empleados está basado en una tarifa diaria más un incentivo, el cual depende del número de asientos producidos durante el día. El programa deberá calcular el salario de cada empleado. Por ejemplo, a un salario básico de $60 por hora y con $1.80 de incentivo por cada asiento producido por encima de 30 unidades, un empleado que ensambla 82 asientos recibirá: ($60)(8 hrs) + (82 unidades producidas - 30 unidades)($1.80) = $573.6

5.- Elabora y dibuja el árbol genealógico de tu familia.

6.- Describe la carta de un restaurante por medio de reglas. Los objetos que interesan aquí son los platos que se pueden consumir y una serie de relaciones que clasifican estos platos en entradas, platos a base de carne o de pescado (plato fuerte) y postres. Este menú es un pequeño banco de datos que se expresa de la siguiente manera:

entrada(ensalada).	entrada(sopa).	entrada(crema_3_quesos).
carne(milanesa).	carne(arrachera).	carne(pollo_asado).
pescado(empapelado).	pescado(salmon).	
postre(flan).	postre(helado).	postre(crepa).

PRÁCTICA 3

LISTAS, BASES DE DATOS Y BASES DE CONOCIMIENTOS EN PROLOG

OBJETIVOS:

Aprender a programar listas con bases de conocimientos en Prolog y acceso a bases de datos creadas en MySQL.

INTRODUCCIÓN:

Una lista en Prolog es un conjunto de nombres de objetos o átomos, separados por comas y encerrados en paréntesis cuadrados.
[elemento1, elemento2, ..., elementoN]
Los miembros de una lista deben ser nombres válidos de objetos, pero todos los miembros deben ser declaraciones de un mismo dominio. Ejemplos:
[‘Jaky’, ‘Erika’, ’Henry’, ’Karen’]
[498.67, 215.39]

La lista puede ser vista, como un objeto con dos partes: [*Cabeza*|*Cola*]
• La *cabeza* de la lista, conformada por el primer elemento.
• La parte restante de la lista, llamada la *cola*.

Una base de datos es un almacén que nos permite guardar un conjunto de información relacionada que se encuentra agrupada ó estructurada.

En una base de conocimientos se almacena el conocimiento proporcionado por el experto humano en un área de la ciencia o el conocimiento humano. En Prolog se expresa por medio de un conjunto de hechos y reglas.

MATERIAL Y EQUIPO A UTILIZAR:

Para la realización de esta práctica son necesarios los siguientes componentes:
Equipo:

- Computadora.

Software:

- SWI-Prolog.
- Canva.
- MySQL.
- Driver ODBC MySQL.

Material:

- Práctica digital o impresa.

DURACIÓN:

- 240 minutos.

DESARROLLO:

Parte 1.- A continuación, se presentan una serie de programas escritos en el lenguaje de programación lógico Prolog:

En la Figura 3.1, se muestra la estructura de una lista.

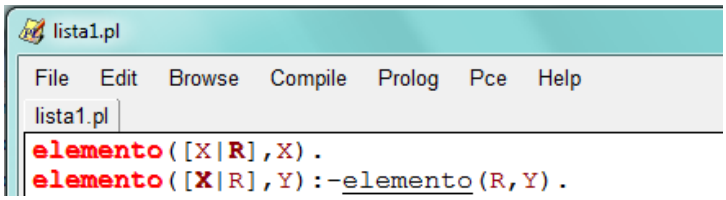


Figura 3.1 Programa con listas en Prolog.

La Figura 3.2, nos muestra la ejecución de la consulta del programa anterior, la cual nos muestra los elementos que ingresamos a la lista que se creó anteriormente.

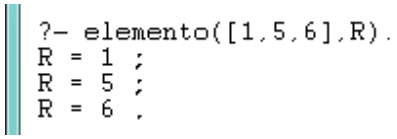


Figura 3.2 Consulta a un programa escrito en Prolog.

En la Figura 3.3, se muestran operaciones aritméticas con listas como: suma, obtener el número de elementos de una lista, producto, suma acumulativa y ciclos for.

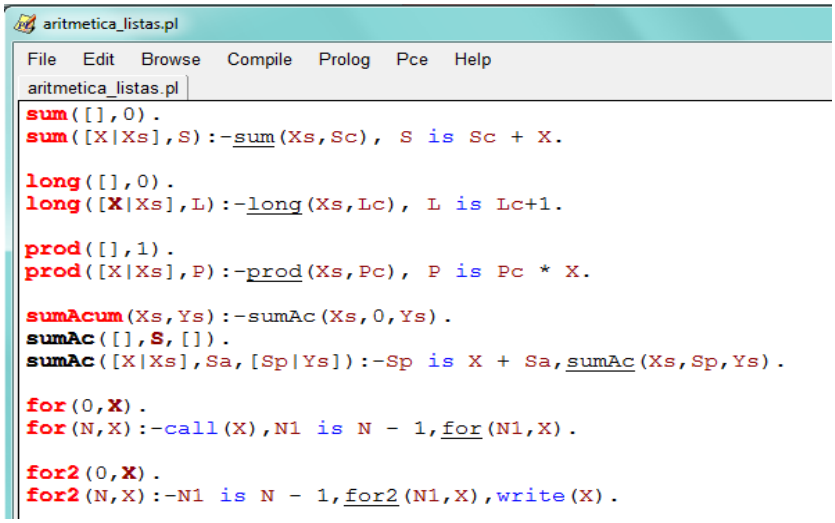


Figura 3.3 Programa con operaciones aritméticas con listas en Prolog.

La Figura 3.4, nos muestra la ejecución de las consultas del programa anterior, como la suma de los elementos de una lista, su longitud, el producto de sus elementos, la suma acumulativa de los elementos de una lista y 2 ciclos for que hacen exactamente lo mismo; como es imprimir en pantalla un símbolo un determinado número de veces.

```
?- sum([3,5,6],S).
S = 14.

?- long([3,5,7,2],D).
D = 4.

?- prod([2,4,5,1],Y).
Y = 40.

?- sumAcum([9,4,5],O).
O = [9, 13, 18].
```

```
?- for(5,write('a')).
aaaaa
true .

?- for2(4,'c').
cccc
true .
```

Figura 3.4 Introducción de valores en listas para mandar llamar sus operaciones.

EVALUACIÓN:

De los siguientes enunciados, se te pide que elabores su programa correspondiente; el cual deberá contener listas, hechos, reglas y consultas en SWI-Prolog.

1.- El usuario deberá ingresar el intervalo de valores que quiere que el programa imprima. Por ejemplo: que imprima la numeración del 100 al 1000.

2.- El usuario debe introducir los siguientes datos: Matricula, Nombre, Número de semestres concluidos, Promedio general por semestre. El programa deberá obtener el promedio general acumulativo por semestre e indicar el porcentaje de variación de cada periodo.

En la Figura 3.5, se muestra la tabla productos con sus registros en el DBMS MySQL.

```
Símbolo del sistema - mysql -uroot -p
mysql> show tables;
+------------------+
| Tables_in_tienda2 |
+------------------+
| productos        |
+------------------+
1 row in set (0.02 sec)

mysql> select * from productos;
+--------+--------+
| marca  | nombre |
+--------+--------+
| nissan | note   |
| nissan | note   |
| nissan | note   |
| nissan | note   |
| nissan | note   |
| nissan | note   |
| nissan | note   |
| nissan | note   |
| nissan | sturu  |
| nissan | sturu  |
+--------+--------+
10 rows in set (0.05 sec)
```

Figura 3.5 Tabla con registros en el DBMS MySQL.

Parte 2.- En la Figura 3.6, se muestra la configuración del origen de datos ODBC. El Conector/ODBC se deberá descargar e instalar antes de mandarlo a llamar desde el Administrador de orígenes de datos ODBC de Windows.

Nombre del origen de datos

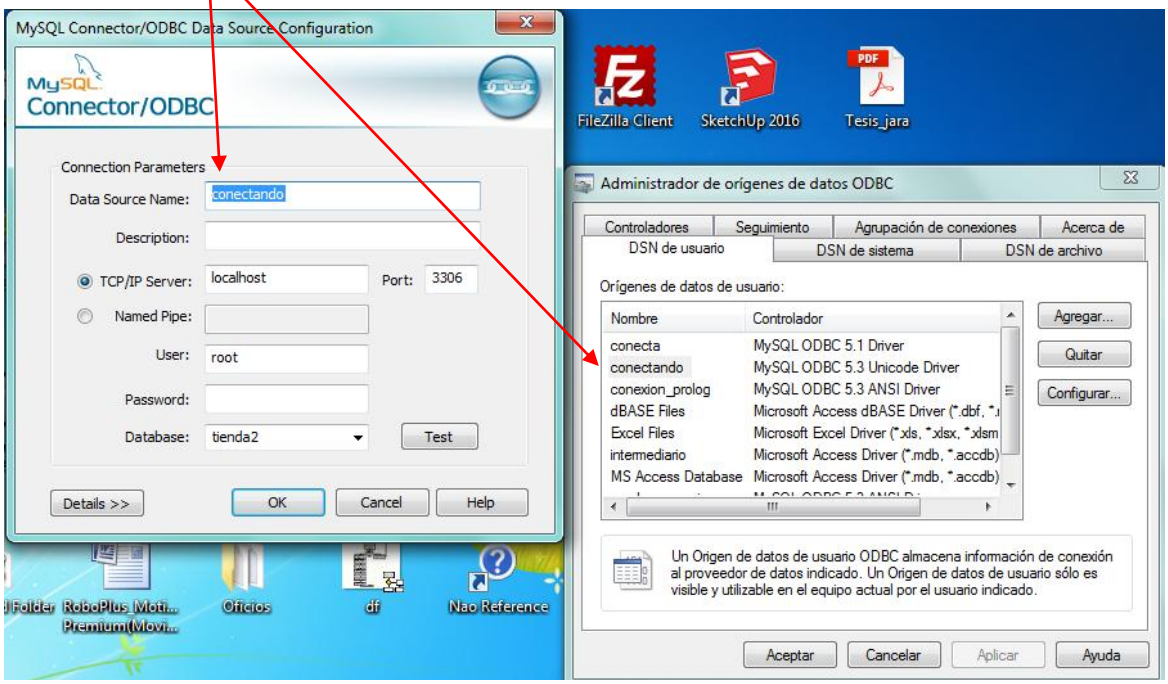


Figura 3.6 Conector/ODBC de MYSQL y creación de un origen de datos con ayuda del Administrador de orígenes de datos ODBC de Windows.

En la Figura 3.7, se muestran las reglas en Prolog para hacer la conexión con la base de datos que fue creada anteriormente en MySQL junto con sus instrucciones DML.

Nombre del origen de datos

Nombre del usuario y contraseña de la base de datos

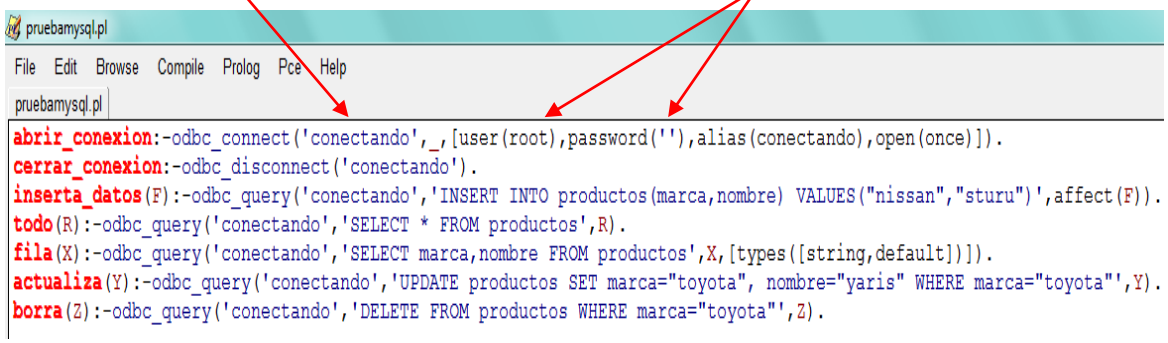


Figura 3.7 Instrucciones SQL para la conexión con la base de datos e instrucciones DML.

La Figura 3.8, nos muestra las consultas pertenecientes a las reglas del programa anterior.

```
?- abrir_conexion.

true.

?- |    fila(X).
X = row("nissan", note) ;
X = row("nissan", note) ;
X = row("nissan", note) ;
X = row("nissan", note) ;
X = row("nissan", note) ;
X = row("nissan", note) ;
X = row("nissan", note) ;
X = row("nissan", note) ;
X = row("nissan", sturu) ;
X = row("nissan", sturu).
```

Figura 3.8 Consultas al programa anterior.

EVALUACIÓN:

Utilizando el ejemplo anterior, de los siguientes enunciados deberás elaborar un programa que contenga lo siguiente: base de datos en MySQL, conexión con base de datos, instrucciones SQL DDL y DML, listas, hechos, reglas y consultas en SWI-Prolog.

1.- El usuario deberá consultar e ingresar datos a la tabla productos con el predicado read()., desde el área de consulta de SWI-Prolog.
2.- El usuario deberá actualizar y eliminar datos de la tabla productos con el predicado read()., desde el área de consulta de SWI-Prolog.
3.- Colocar 2 reglas para crear una base de datos y una vista desde prolog; posteriormente introducir los datos para crear la base de datos y la vista con el predicado read().,desde el área de consulta de SWI-Prolog.

REFERENCIAS

1. Canva (2017). Página oficial de Canva. Recuperado el 2 de junio de 2017, de https://www.canva.com/
2. Giarratano Riley (2001). SISTEMAS EXPERTOS: PRINCIPIOS Y PROGRAMACIÓN. México: THOMSON LEARNING.
3. Nilsson Nils J. (2001). INTELIGENCIA ARTIFICIAL. UNA NUEVA SÍNTESIS. México: Mc Graw Hill.
4. Russell Stuart y Norvig P. (2004). INTELIGENCIA ARTIFICIAL. UN ENFOQUE MODERNO. México: PEARSON.
5. Santos Reyes José y Duro Fernández Richard J. (2005). EVOLUCIÓN ARTIFICIAL Y ROBÓTICA AUTÓNOMA. México: Alfaomega Ra-Ma.
6. SWI-Prolog 7.5.9 (2017). Página oficial de SWI Prolog. Recuperado el 1 de junio de 2017, de http://www.swi-prolog.org/