TRABAJO PRACTICO FINAL

Departamento: Desarrollo Productivo y Tecnológico
Materia: Conceptos y Paradigmas de Programación
Profesora: Elida Beatriz Leoni
Alumno: Luciano Moliterno 40238958

Gestión de Ventas para Dulces & Delicias

Enunciado

Empresa: Dulces & Delicias
Actividad: Producción y venta de repostería artesanal.
Descripción del Trabajo:
Dulces & Delicias solicitó un sistema que les permitiera gestionar sus productos, clientes y ventas de manera eficiente.
Para ello, desarrolle una base de datos en **Prolog**, donde se almacenan productos, clientes y ventas, junto con consultas específicas para obtener información relevante.
El programa permite listar productos, clientes y ventas, así como analizar compras, determinar clientes frecuentes y calcular ingresos entre otros mas específicos.

Conclusión

El desarrollo del sistema en Prolog permitió una representación declarativa y lógica del conocimiento, facilitando la gestión de reglas y consultas sobre la base de datos. A diferencia de un enfoque imperativo u orientado a objetos, donde tendría que manejar estructuras de datos explícitamente y definir algoritmos de búsqueda y procesamiento de datos, en Prolog simplemente declare los hechos y reglas, dejando que el motor de inferencia resuelva las consultas de manera eficiente.

Si este programa lo hubiese desarrollado en un paradigma imperativo, habría requerido estructuras como listas o bases de datos relacionales, junto con múltiples bucles y condicionales para filtrar y organizar la información. En un paradigma orientado a objetos, se habrían creado clases y métodos específicos para manipular los datos, agregando más complejidad en la implementación. Con Prolog, en cambio, la estructura del conocimiento se define de manera más clara y flexible, permitiendo consultas poderosas con pocas líneas de código.

Codigo

Bases_de_datos.pl:

```
% Productos y sus componentes (producto, precio, lista de ingredientes y cantidades necesarias)
producto(pastel_chocolate, 2000, [(harina, 2), (azucar, 1), (chocolate, 3), (huevo, 2), (manteca, 1)]).
producto(pastel_vainilla, 1800, [(harina, 2), (azucar, 1), (vainilla, 2), (huevo, 2), (manteca, 1)]).
producto(tarta_frutilla, 2200, [(harina, 2), (azucar, 2), (frutilla, 5), (crema, 2), (manteca, 1)]).
producto(tarta_manzana, 2100, [(harina, 2), (azucar, 2), (manzana, 4), (manteca, 2), (canela, 1)]).
producto(cupcake_chocolate, 500, [(harina, 1), (azucar, 1), (chocolate, 2), (huevo, 1), (manteca, 1)]).
producto(cupcake_vainilla, 450, [(harina, 1), (azucar, 1), (vainilla, 2), (huevo, 1), (manteca, 1)]).
producto(alfajor_dulce, 300, [(harina, 1), (azucar, 1), (dulce_de_leche, 2), (chocolate, 1), (manteca,
1)]).
producto(alfajor_maicena, 280, [(harina, 1), (azucar, 1), (dulce_de_leche, 2), (coco, 1), (manteca, 1)]).
producto(budin_naranja, 1500, [(harina, 2), (azucar, 1), (naranja, 2), (huevo, 3), (manteca, 2)]).
producto(budin_limon, 1450, [(harina, 2), (azucar, 1), (limon, 2), (huevo, 3), (manteca, 2)]).

% Stock de productos (producto, cantidad disponible)
stock(pastel_chocolate, 10).
stock(pastel_vainilla, 8).
stock(tarta_frutilla, 5).
stock(tarta_manzana, 6).
stock(cupcake_chocolate, 15).
stock(cupcake_vainilla, 12).
stock(alfajor_dulce, 20).
stock(alfajor_maicena, 18).
stock(budin_naranja, 7).
stock(budin_limon, 9).

% Clientes (nombre, contacto)
cliente(juan, "juan@mail.com").
cliente(maria, "maria@mail.com").
cliente(carlos, "carlos@mail.com").
cliente(laura, "laura@mail.com").

% Ventas (fecha, cliente, producto, cantidad)
venta('2024-02-10', juan, pastel_chocolate, 2).
venta('2024-02-12', maria, tarta_frutilla, 1).
venta('2024-02-15', carlos, cupcake_vainilla, 6).
venta('2024-02-17', laura, budin_limon, 3).
venta('2024-02-18', juan, alfajor_dulce, 4).
venta('2024-02-20', maria, pastel_vainilla, 2).
```

Consultas.pl:

```
% Cargar la base de datos desde el archivo 'base_de_datos.pl'
:- consult('base_de_datos.pl').  % Cargar la base de datos en Prolog
% ------------------------------
% CONSULTAS GENERALES
% ------------------------------
% 1. Listar todos los productos disponibles en la base de datos.
% Recorre todos los hechos 'producto/3' y muestra su nombre, precio e ingredientes de cada uno.
listar_productos :-
    producto(Nombre, Precio, Ingredientes), /* Buscar cada producto en la base de datos */
    write(Nombre), write(': $'), write(Precio), /* Mostrar el nombre y precio del producto */
    write(', Ingredientes: '), write(Ingredientes), nl, /* Mostrar los ingredientes */
    fail. /* Forzar el backtracking para obtener todos los productos */
listar_productos.
% 2. Listar todos los clientes registrados en la base de datos.
% Se recorren los hechos 'cliente/2' y se muestra el nombre y el contacto de cada cliente.
listar_clientes :-
    cliente(Nombre, Contacto), /* Buscar cada cliente registrado */
    write(Nombre), write(' - Contacto: '), write(Contacto), nl, /* Imprimir la información del cliente */
    fail. /* Forzar el backtracking para obtener todos los clientes */
listar_clientes.
% 3. Listar todas las ventas registradas.
% Se imprimen la fecha, el cliente, el producto y la cantidad de cada venta en la base de datos.
listar_ventas :-
    venta(Fecha, Cliente, Producto, Cantidad), /* Buscar cada venta en la base de datos */
    write(Fecha), write(': '), write(Cliente), /* Mostrar la fecha y el cliente que realizó la compra */
    write(' compró '), write(Cantidad), write(' de '), write(Producto), nl, /* Mostrar detalles de la venta
*/
    fail. /* Forzar el backtracking para obtener todas las ventas */
listar_ventas.
% 4. Listar el stock actual de cada producto.
% Se recorren los hechos 'stock/2' y se muestra la cantidad disponible para cada producto.
listar_stock :-
    stock(Producto, Cantidad), /* Buscar cada producto y su stock en la base de datos */
    write('Stock de '), write(Producto), write(': '), write(Cantidad), nl, /* Mostrar la cantidad en stock */
    fail. /* Forzar el backtracking para obtener todos los stocks */
listar_stock.
% ------------------------------
% CONSULTAS ESPECÍFICAS
% ------------------------------
% 5. Mostrar todas las compras realizadas por un cliente en particular.
% Se buscan todas las ventas asociadas al cliente y se muestran con fecha, producto y cantidad.
ventas_cliente(Nombre) :-
    venta(Fecha, Nombre, Producto, Cantidad), /* Buscar todas las ventas de ese cliente */
    write(Fecha), write(': '), write(Nombre),
    write(' compró '), write(Cantidad), write(' de '), write(Producto), nl,
    fail. /* Forzar el backtracking para obtener todas las compras del cliente */
ventas_cliente(_).
% 6. Mostrar todas las ventas de un producto específico.
% Se recorren todas las ventas y se imprimen solo aquellas en las que se vendió el producto
indicado.
ventas_producto(Producto) :-
    venta(Fecha, Cliente, Producto, Cantidad), /* Buscar todas las ventas del producto dado */
    write(Fecha), write(': '), write(Cliente),
    write(' compró '), write(Cantidad), nl,
```

```
    fail.
ventas_producto(_).
% 7. Mostrar todas las ventas realizadas dentro de un rango de fechas.
% Se comparan las fechas de las ventas con los límites proporcionados y se imprimen las que cumplen la condición.
ventas_entre(FechaInicio, FechaFin) :-
    venta(Fecha, Cliente, Producto, Cantidad), /* Buscar todas las ventas registradas */
    Fecha @>= FechaInicio, Fecha @=< FechaFin, /* Filtrar por rango de fechas */
    write(Fecha), write(': '), write(Cliente),
    write(' compró '), write(Cantidad), write(' de '), write(Producto), nl,
    fail.
ventas_entre(_, _).
% ------------------------------
% ANÁLISIS DE COMPRAS
% ------------------------------
% 8. Determinar qué cliente ha realizado más compras (cantidad de transacciones).
% Se recopilan todas las compras, se ordenan y se cuentan cuántas veces aparece cada cliente.
cliente_mas_compras :-
    findall(Cliente, venta(_, Cliente, _, _), Clientes), /* Obtener lista de clientes que han comprado */
    msort(Clientes, Ordenados), /* Ordenar la lista para agrupar elementos iguales */
    sort(Ordenados, Unicos), /* Eliminar duplicados y obtener lista única de clientes */
    obtener_cliente_mas_compras(Unicos, ClienteMax, MaxCompras), /* Determinar quién ha comprado más */
    write('El cliente con más compras es '), write(ClienteMax),
    write(' con '), write(MaxCompras), write(' compras.'), nl.
% 8.1 Función auxiliar para obtener el cliente con más compras
obtener_cliente_mas_compras([], '', 0). /* Caso base: Si no hay clientes, devolver vacío y 0 compras. */
obtener_cliente_mas_compras([Cliente|Resto], ClienteMax, MaxCompras) :-
    findall(_, venta(_, Cliente, _, _), Compras), /* Obtener todas las compras realizadas por el cliente actual */
    length(Compras, Total), /* Contar cuántas compras ha realizado */
    obtener_cliente_mas_compras(Resto, OtroClienteMax, OtroMaxCompras), /* Llamada recursiva para comparar con otros clientes */
    (Total > OtroMaxCompras -> ClienteMax = Cliente, MaxCompras = Total ;  /* Si el cliente actual ha comprado más, lo guardamos */
                    ClienteMax = OtroClienteMax, MaxCompras = OtroMaxCompras). /* Si no, conservar el cliente con más compras hasta ahora */
% 9. Determinar el cliente con la compra de mayor valor total.
% Se calcula el monto de cada compra y se ordenan de mayor a menor para obtener el cliente con el mayor gasto.
cliente_compra_mayor :-
    findall((Cliente, Monto),
        (venta(_, Cliente, Producto, Cantidad),
        producto(Producto, Precio, _),
        Monto is Precio * Cantidad), /* Calcular el monto de cada compra */
        Compras),
    sort(2, @>=, Compras, Ordenadas), /* Ordenar de mayor a menor según el monto */
    Ordenadas = [(Cliente, Monto) | _], /* Obtener la compra con el monto más alto */
    write('El cliente con la compra INDIVIDUAL es '),
    write(Cliente), write(' con un monto mayor de $'), write(Monto), nl.
% 10. Cliente con la compra de mayor monto total desde un valor determinado
% Encuentra clientes cuyo gasto total en todas sus compras sea MAYOR o IGUAL a un monto especificado (MontoMin)
cliente_monto_mayor(MontoMin) :-
    findall((Cliente, Total),
        (   cliente(Cliente, _), /* Buscar todos los clientes registrados */
```

```
        findall(Monto,
          (venta(_, Cliente, Producto, Cantidad),
           producto(Producto, Precio, _),
           Monto is Precio * Cantidad), /* Calcular el total gastado por cada cliente */
          Montos),
        sum_list(Montos, Total), /* Sumar todos los montos */
        Total >= MontoMin  /* Filtrar por monto mínimo */
      ),
      Compras),
  sort(2, @>=, Compras, Ordenadas), /* Ordenar por monto total de mayor a menor */
  listar_clientes_monto(Ordenadas).
% 11. Cliente con la compra de menor monto total desde un valor determinado
% Encuentra clientes cuyo gasto total en todas sus compras sea MENOR o IGUAL a un monto especificado (MontoMax)
cliente_monto_menor(MontoMax) :-
  findall((Cliente, Total),
    (  cliente(Cliente, _),
      findall(Monto,
        (venta(_, Cliente, Producto, Cantidad),
         producto(Producto, Precio, _),
         Monto is Precio * Cantidad),
        Montos),
      sum_list(Montos, Total),
      Total =< MontoMax  /* Filtrar por monto máximo */
    ),
    Compras),
  (  Compras == [] /* Si la lista está vacía, no hay clientes que cumplan la condición */
  ->  write('No hay clientes con compras menores o iguales a $'), write(MontoMax), nl
  ;  sort(2, @=<, Compras, Ordenadas), /* Ordenar de menor a mayor */
     listar_clientes_monto(Ordenadas)
  ).
% 10.1-11.1 Función auxiliar para imprimir los clientes con sus montos de compra
listar_clientes_monto([]).
listar_clientes_monto([(Cliente, Monto) | Resto]) :-
  write('Cliente: '), write(Cliente), write(', Monto total: $'), write(Monto), nl,
  listar_clientes_monto(Resto).
% ------------------------------
% INFORMACIÓN SOBRE PRODUCTOS
% ------------------------------
% 12. Listar cada producto con los ingredientes necesarios para su elaboración.
% Se recorren los hechos 'producto/3' e imprime el nombre y la lista de ingredientes de cada producto.
listar_ingredientes :-
  producto(Producto, _, Ingredientes),
  write(Producto), write(' requiere: '), write(Ingredientes), nl, fail.
listar_ingredientes.
% ------------------------------
% CONSULTAS ADICIONALES
% ------------------------------
% 13. Obtener una lista de clientes que compraron un producto específico.
% Se buscan todas las ventas donde aparece el producto y se imprimen los nombres de los clientes correspondientes.
clientes_que_compraron(Producto) :-
  venta(_, Cliente, Producto, _),
  write(Cliente), nl, fail.
clientes_que_compraron(_).
```

```
% 14. Calcular la cantidad total vendida de un producto.
% Se suman todas las unidades vendidas del producto especificado.
cantidad_vendida(Producto, Total) :-
    findall(Cantidad, venta(_, _, Producto, Cantidad), Cantidades),
    sum_list(Cantidades, Total). /* Sumar todas las cantidades vendidas */
% 15. Calcular los ingresos totales generados por las ventas de un producto.
% Se obtiene la cantidad total vendida y se multiplica por el precio del producto.
ingresos_por_producto(Producto, Ingresos) :-
    cantidad_vendida(Producto, CantidadTotal), /* Obtener la cantidad total vendida */
    producto(Producto, Precio, _), /* Obtener el precio del producto */
    Ingresos is CantidadTotal * Precio. /* Calcular los ingresos */
% 16. Calcular los ingresos totales de la empresa sumando todas las ventas.
% Se calculan los montos de cada venta y se suman para obtener el total.
ingresos_totales(Total) :-
    findall(Ingreso,
    (venta(_, _, Producto, Cantidad),
    producto(Producto, Precio, _),
    Ingreso is Precio * Cantidad), /* Calcular ingreso por cada venta */
    ListaIngresos),
    sum_list(ListaIngresos, Total). /* Sumar todos los ingresos */
% 17. Identificar clientes frecuentes, es decir, aquellos que han comprado más de una vez.
% Se cuentan las compras por cliente y se imprimen aquellos con más de una compra.
clientes_frecuentes :-
    findall(Cliente, venta(_, Cliente, _, _), Clientes), /* Obtener lista de clientes que han comprado */
    msort(Clientes, Ordenados), /* Ordenar para agrupar clientes iguales */
    sort(Ordenados, Unicos), /* Eliminar duplicados */
    listar_frecuentes(Unicos).
% 17.1 Función auxiliar para imprimir clientes frecuentes
listar_frecuentes([]).
listar_frecuentes([Cliente|Resto]) :-
    findall(_, venta(_, Cliente, _, _), Compras), /* Obtener todas las compras del cliente */
    length(Compras, Total), /* Contar cuántas compras ha realizado */
    Total > 1, /* Filtrar solo aquellos con más de una compra */
    write(Cliente), write(' compró '), write(Total), write(' veces.'), nl,
    listar_frecuentes(Resto).
listar_frecuentes([_|Resto]) :- listar_frecuentes(Resto).
```

Comandos y Funciones Más Usadas en el Código

Carga de Base de Datos

- :- consult('base_de_datos.pl').

Carga la base de datos desde un archivo externo llamado **base_de_datos.pl.**
Esto permite que los hechos (**producto/3**, **cliente/2**, **venta/4**, etc.) estén disponibles en la ejecución de Prolog.

Consultas Generales

- listar_productos.

Lista todos los productos disponibles con su precio e ingredientes.
Utiliza **producto(Nombre, Precio, Ingredientes)**. para recuperar los datos.
fail. fuerza la búsqueda de más coincidencias.

- listar_clientes.

Muestra todos los clientes registrados en la base de datos con su contacto.
Usa **cliente(Nombre, Contacto).** para obtener los datos.

- listar_ventas.

Muestra todas las ventas registradas en la base de datos con fecha, cliente, producto y cantidad.
Usa **venta(Fecha, Cliente, Producto, Cantidad).** para recorrer los datos.

- listar_stock.

Muestra el stock actual de cada producto.
Usa **stock(Producto, Cantidad).** para recuperar la información.

Consultas Específicas

- ventas_cliente(Nombre).

Muestra todas las compras realizadas por un cliente específico.
Busca en **venta/4** todas las transacciones que correspondan al **Nombre.**

- ventas_producto(Producto).

Lista todas las ventas donde se haya comprado un producto específico.

- ventas_entre(FechaInicio, FechaFin).

Muestra todas las ventas realizadas dentro de un rango de fechas.
Usa **Fecha @>= FechaInicio, Fecha @=< FechaFin** para filtrar los resultados.

Análisis de Compras

- cliente_mas_compras.

Encuentra el cliente que ha realizado más compras en total.
Usa **findall(Cliente, venta(_, Cliente, _, _), Clientes).** para obtener la lista de clientes que han comprado.
Luego cuenta cuántas veces aparece cada cliente.

- cliente_compra_mayor.

Determina el cliente con la compra individual de mayor valor.
Usa **Monto is Precio * Cantidad** para calcular el valor de cada compra y **sort/3** para ordenarlas de mayor a menor.

- cliente_monto_mayor(MontoMin).

Encuentra clientes cuyo gasto total sea mayor o igual a un monto dado.
Usa **findall(Monto, venta(_, Cliente, Producto, Cantidad), Montos), sum_list(Montos, Total).**

- cliente_monto_menor(MontoMax).

Encuentra clientes cuyo gasto total sea menor o igual a un monto dado.
Similar a **cliente_monto_mayor/1**, pero filtra por **Total =< MontoMax.**

Información sobre Productos

- listar_ingredientes.

Lista todos los productos junto con sus ingredientes.
Usa **producto(Producto, _, Ingredientes).** para obtener los datos.

Consultas Adicionales

- clientes_que_compraron(Producto).

Muestra una lista de clientes que compraron un producto específico.

- cantidad_vendida(Producto, Total).

Calcula cuántas unidades de un producto han sido vendidas en total.
Usa **findall(Cantidad, venta(_, _, Producto, Cantidad), Cantidades), sum_list(Cantidades, Total).**

- ingresos_por_producto(Producto, Ingresos).

Calcula los ingresos totales generados por la venta de un producto.

- ingresos_totales(Total).

Calcula los ingresos totales de la empresa sumando todas las ventas.

- clientes_frecuentes.

Muestra clientes que han comprado más de una vez.
Usa **length(Compras, Total), Total > 1** para filtrar a los clientes frecuentes.

Funciones Auxiliares

- obtener_cliente_mas_compras/3

Función auxiliar para encontrar el cliente con más compras.
Usa recursión para comparar cuántas compras ha hecho cada cliente.

- listar_clientes_monto/1

Función auxiliar para listar clientes con sus montos de compra.

- listar_frecuentes/1

Función auxiliar para filtrar e imprimir clientes que han comprado más de una vez.