**Trabajo Optimización**

Daniel Santa Rendón - C.c. 1040050021

Daniel Torres González - C.c. 1020492000

SELECT codVenta, FechaVenta, TotalVenta

FROM venta

WHERE FechaVenta > '03-09-2019 12:00:00' AND TipoVenta = 'online' AND

metodo\_de\_pago = 'pse';

Venta usa un almacenamiento tipo Hash.

Total de bytes ocupados por la tabla ventas = 59.

Tamaño de un bloque = 4096 (Este valor es constante para todo el trabajo)

ClaveCluster Sin índice

El atributo FechaVenta tiene un índice Cluster, los otros dos atributos no tienen ningún índice

|  |  |  |  |
| --- | --- | --- | --- |
| **OPCIÓN** | **DESCRIPCIÓN** | **COSTO** | **Evaluación** |
| Opción 1 | Obtener tuplas de R que satisfacen una condición de la forma A=c. Donde A tiene un índice clúster. Sea el tamaño de la imagen del índice. |  | No factible; el índice clúster no tiene una condición de igualdad. |
| Opción 2 | Usar un índice clúster en un atributo A, donde A θ C es una de las condiciones Ci y θ ϵ ( <, <= , >, >= ). Luego aplicamos las condiciones restantes al resultado. En promedio leemos: |  | Uso índice FechaVenta. |
| Opción 3 | Si hay un índice no clúster que coincida con una condición A=c, use  ese índice para hallar todas las tuplas  que cumplen tal condición y aplique  las otras condiciones a las tuplas  resultantes. |  | No factible; no hay índices no clúster en nuestros registros. |
| Opción 4 | Si R está almacenado en un archivo  independiente, podemos simplemente  leer todas las tuplas y aplicar las Ci a  éstas. |  |  |
| Opción 5 | Si R no está almacenada  independientemente, pero tiene un  índice clúster en un atributo ó en una  colección de atributos, sin importar  que no estén involucrados en la  condición de la consulta, se usa tal  índice para obtener todas las tuplas  de R y aplicar luego a ellas las  condiciones. |  | No factible; las tuplas están almacenadas en un archivo independiente. |
| Opción 6 | Si hay un índice no clúster en un  atributo A y A θ C es una condición  donde θ ϵ ( < , > , >= , <= ) ; use el índice para obtener las tuplas de R que satisfagan la condición y aplique  las otras condiciones al resultado. |  | No factible; no hay índices no clúster en nuestros registros. |
| Opción 7 | Use un índice no clúster de cualquier  clase para hallar las tuplas de R y  aplicar todas las condiciones a ellas. |  | No factible; no hay índices no clúster en nuestros registros. |
| Opción 8 | Si ninguna de las anteriores opciones  es posible, simplemente recupere  todos los bloques que podrían  contener tuplas de R. |  |  |

La mejor opción fue la 2, pues es el único atributo en el WHERE que posee un índice cluster y por lo tanto el costo de la consulta será menor.

**INNER JOIN ENTRE DOS TABLAS**

Se seleccionaron dos tablas del modelo entidad relación, venta y domicilio. El *Join* entre las tablas se hizo por el atributo CodVenta que tienen una relación 1 a 1. Para los registros, 1 de cada 5 ventas tiene domicilio.

***SELECT*** *CodVenta, MetodoPago, FechaVenta, TipoVenta, TotalVenta, NumeroCaja, IDCliente, CodSucursal, CodDomicilio, Entregado, FechaEntrega, FechaEntregado, Descripción, IDEmpleado*

***FROM*** *Venta V* **JOIN** *Domicilio D* **ON** *V.CodVenta=D.CodVenta*

El total de bytes para la tabla venta es 59 bytes y para la tabla Domicilio es 101 bytes.

**Escenario 1**

**Caso 1**

La tabla R tiene 900.000 tuplas que se almacenan en 12.964 bloques, entonces la tabla S tiene 180.000 que se almacenan en 4.439 bloques.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datos de entrada por tabla** | **Estrategia Usada** | **Fórmula usada para costo de la entrada** | **Costo entrada.** |
| Domicilio usa un almacenamiento tipo Hash.  Venta usa un almacenamiento tipo Hash.      Imágenes de atributo con índices.  M = 2.221 | Selección en un producto |  |  |
| Sort Merge |  |  |
| Hash Join. |  |  |

**Caso 2**

La tabla R tiene 5’400.000 tuplas que se almacenan en 77.784 bloques, entonces la tabla S tiene 1’800.000 que se almacenan en 26.631 bloques.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datos de entrada por tabla** | **Estrategia Usada** | **Fórmula usada para costo de la entrada** | **Costo entrada.** |
| Domicilio usa un almacenamiento tipo Hash.  Venta usa un almacenamiento tipo Hash.      Imágenes de atributo con índices.    M = 2.221 | Selección en un producto |  | 959.723,7 |
| Sort Merge |  |  |
| Hash Join. |  |  |

**Caso 3**

La tabla R tiene tuplas que se almacenan en bloques, entonces la tabla S tiene que se almacenan en bloques.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datos de entrada por tabla** | **Estrategia Usada** | **Fórmula usada para costo de la entrada** | **Costo entradqqa.** |
| Domicilio usa un almacenamiento tipo Hash.  Venta usa un almacenamiento tipo Hash.      Imágenes de atributo con índices.  M = 2.221 | Selección en un producto |  | 3’785.608,6 |
| Sort Merge |  |  |
| Hash Join. |  |  |

**Conclusión escenario 1:** Para el caso de menor cantidad de datos el mejor resultado fue por medio de la técnica selección de un producto, pero al aumentar los valores como en el caso 2 y 3, los menores costos fueron datos por la estrategia sort merge, comparando los resultados de esta última en los tres casos se evidencia que su crecimiento es muy estable, mientras que con selección de un producto el crecimiento en costos al aumentar el total de tuplas es muy elevado. Por otro lado, el hash join es el peor de las estrategias en nuestro caso debido al tamaño de las imágenes, pues solo esta estrategia hace uso de ese valor.

**Escenario 2**

**Caso 1**

La tabla R tiene 5’400.000 tuplas que se almacenan en 77.784 bloques, entonces la tabla S tiene 1’800.000 que se almacenan en 26.631 bloques.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datos de entrada por tabla** | **Estrategia Usada** | **Fórmula usada para costo de la entrada** | **Costo entrada.** |
| Domicilio usa un almacenamiento tipo Hash.  Venta usa un almacenamiento tipo Hash.      Imágenes de atributo con índices.  M= 2.221 | Usando índice en tabla 1 | R Compacta  S tiene índice Clúster |  |
| usando índice en tabla B |  |  |
| usando ambos índices |  | 2’160.000 |

**Caso 2**

La tabla R tiene tuplas que se almacenan en bloques, entonces la tabla S tiene que se almacenan en bloques.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datos de entrada por tabla** | **Estrategia Usada** | **Fórmula usada para costo de la entrada** | **Costo entrada.** |
| Domicilio usa un almacenamiento tipo Hash.  Venta usa un almacenamiento tipo Hash.      Imágenes de atributo con índices.  M = 2.221 | Usando índice en tabla 1 | R Compacta  S tiene índice Clúster |  |
| usando índice en tabla B |  |  |
| usando ambos índices |  | 4’320.000 |

**Conclusión escenario 2:**  Al comparar el crecimiento de los costos usando el índice de la tabla 1 y luego el índice de la tabla B, se observa que el crecimiento es estable pero con un costo mayor cuando se usa el índice de la tabla Ventas(tabla 1). En este escenario la mejor estrategia para los dos casos fue usando el índice en la tabla domicilio (tabla B), esto es debido a que la imagen de venta es mucho mayor que la imagen de domicilio, por lo que el costo se reduce considerablemente. Por el contrario usando los dos índices el costo es mucho más grande a comparación de las dos estrategias anteriores pues, como en el escenario anterior, la imagen de nuestro caso es un valor muy elevado incluso para la tabla S, aún así su crecimiento es estable en ambos casos.

**II. Optimización algebraica**

Tablas a utilizar:

Cliente(#IDCliente, NombreUsuario, DireccionCliente, Contrasena, DistanciaDomicilio, EdadCliente, NombreCompleto, TelefonoCliente)

Venta(#CodVenta, MetodoDePago, FechaVenta, TipoVenta, TotalVenta, NumeroCaja, CodSucursal, IDCliente)

ProdxVenta(#CodVenta, #CodSucursal, #CodProducto, CantidadProductosVenta, SubTotal, PrecioEnVenta, CodPromocion)

La vista será:

CREATE VIEW Ventas\_A\_Clientes\_Registrados

SELECT c.IDCliente, c.NombreCompleto, c.DireccionCliente, c.TelefonoCliente, c.EdadCliente, c.NombreUsuario, v.CodVenta, v.MetodoDePago, v.FechaVenta, v.TipoVenta, v.TotalVenta, v.NumeroCaja, v.CodSucursal, p.CodProducto, p.CantidadProductosVenta, p.SubTotal, p.PrecionEnVenta, p.CodPromocion

FROM cliente c, venta v, prodxventa p

WHERE c.IDCliente = v.IDCliente AND v.CodVenta = p.CodVenta;

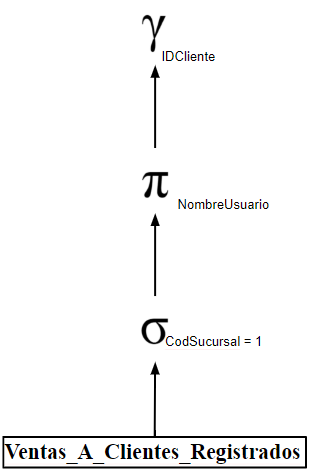
**Consulta a optimizar**

**SELECT NombreUsuario**

**FROM Ventas\_A\_Clientes\_Registrados**

**WHERE CodSucursal = 1**

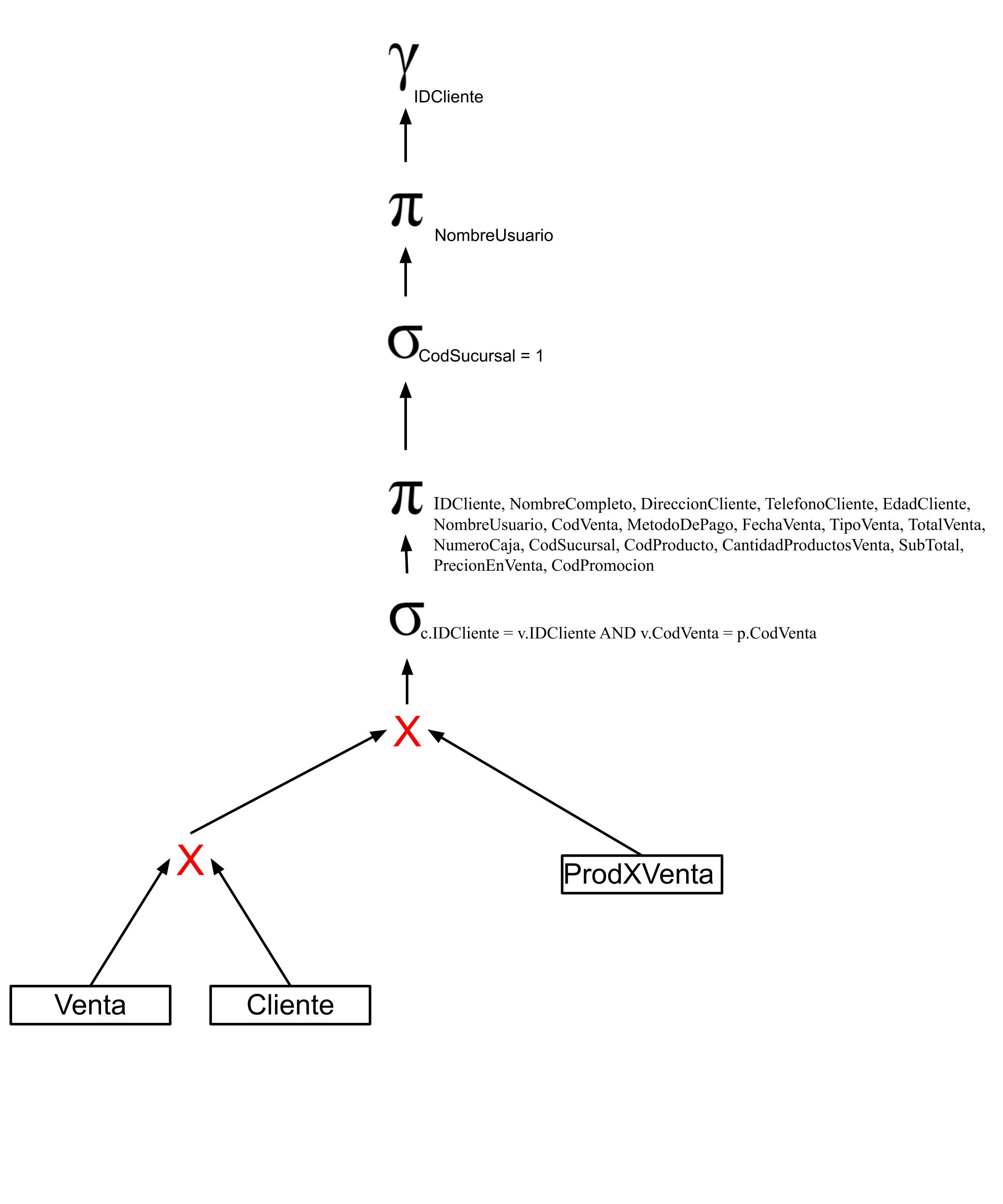
**ORDER BY IDCliente**

****

**Optimización**

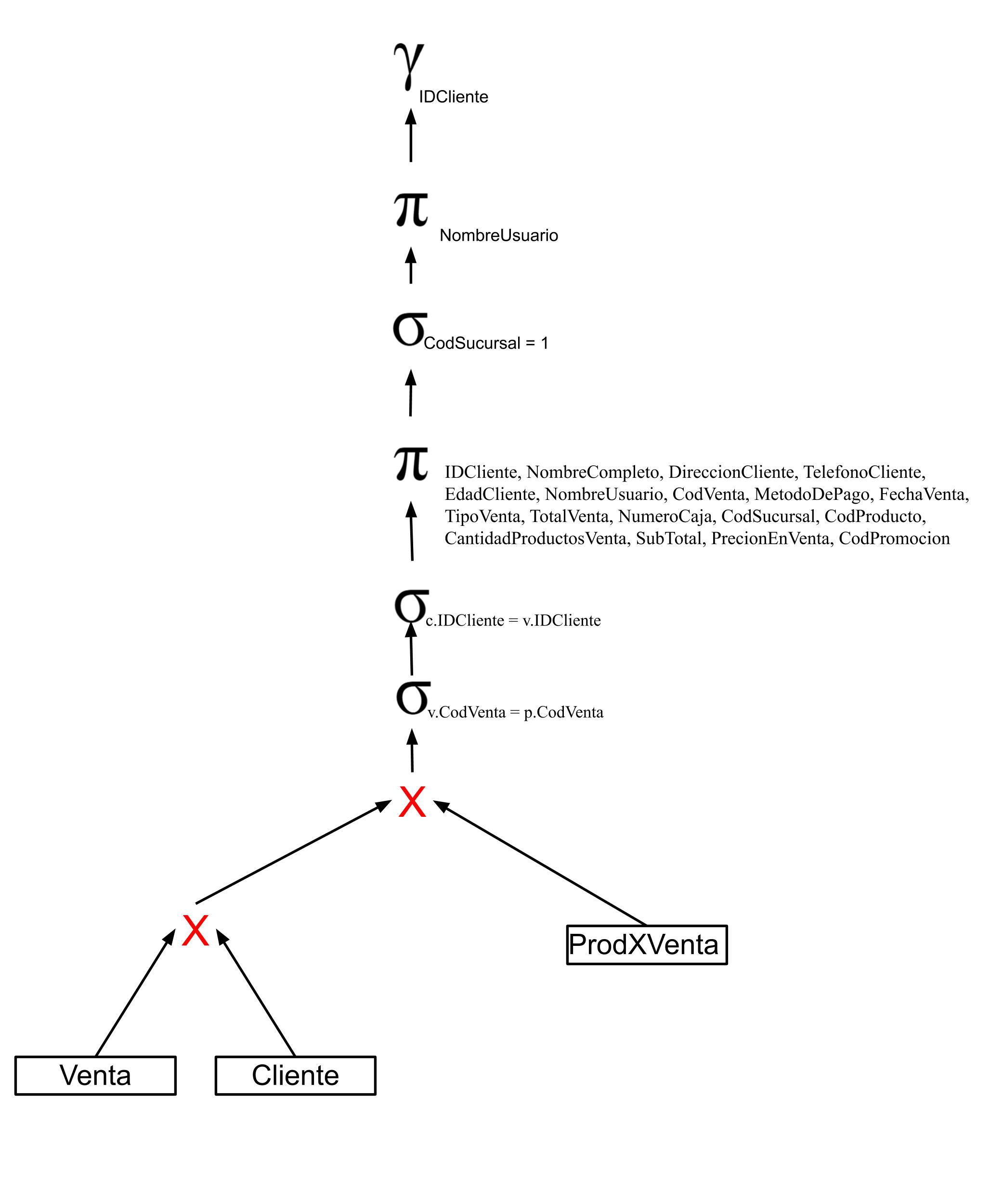
**1.**

Reemplazar la vista por su definición



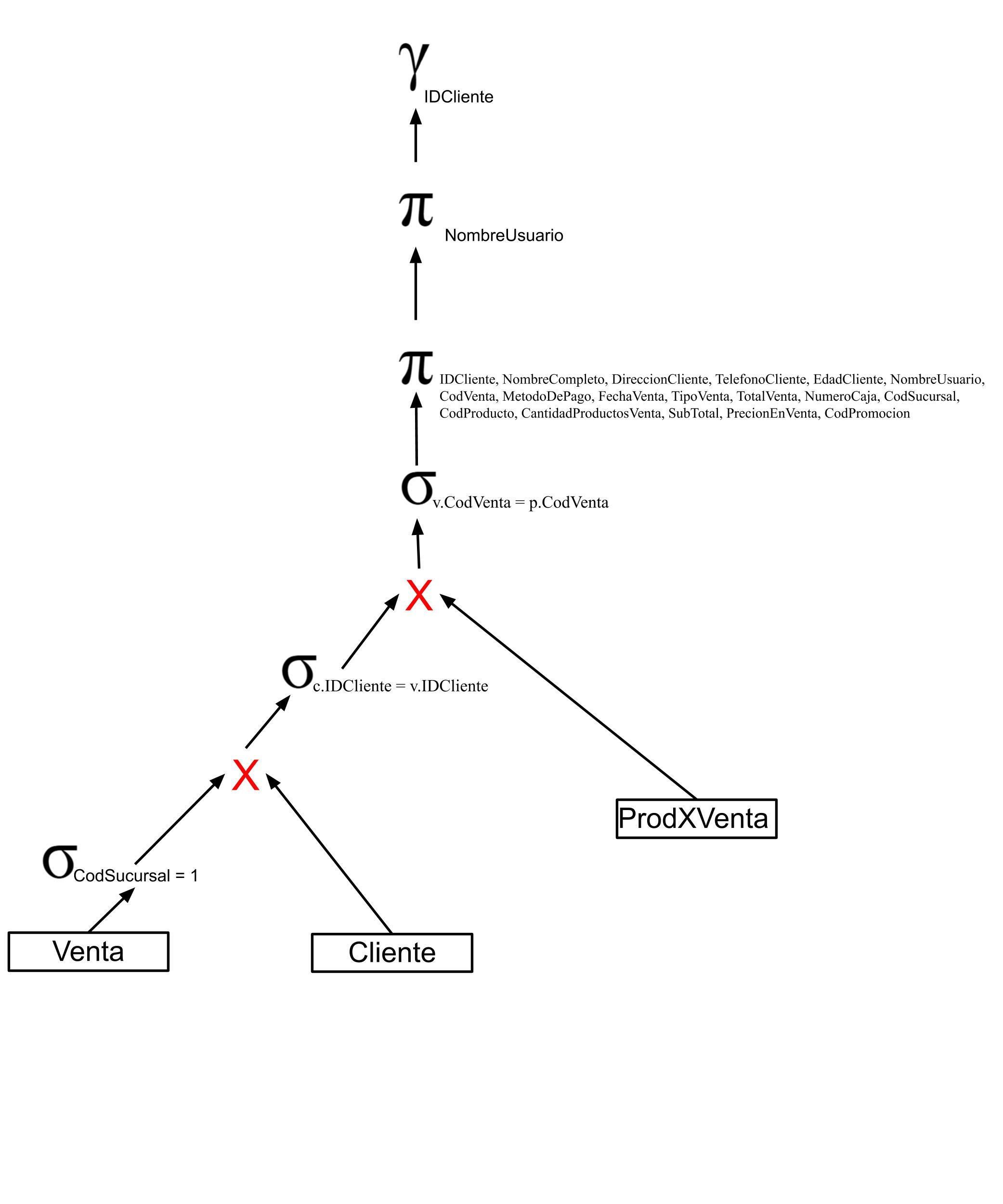
**2.**

Uso ley 4 para separar cada selección con condiciones de la forma F1 ^ F2 en una cascada de selecciones.



**3 .**

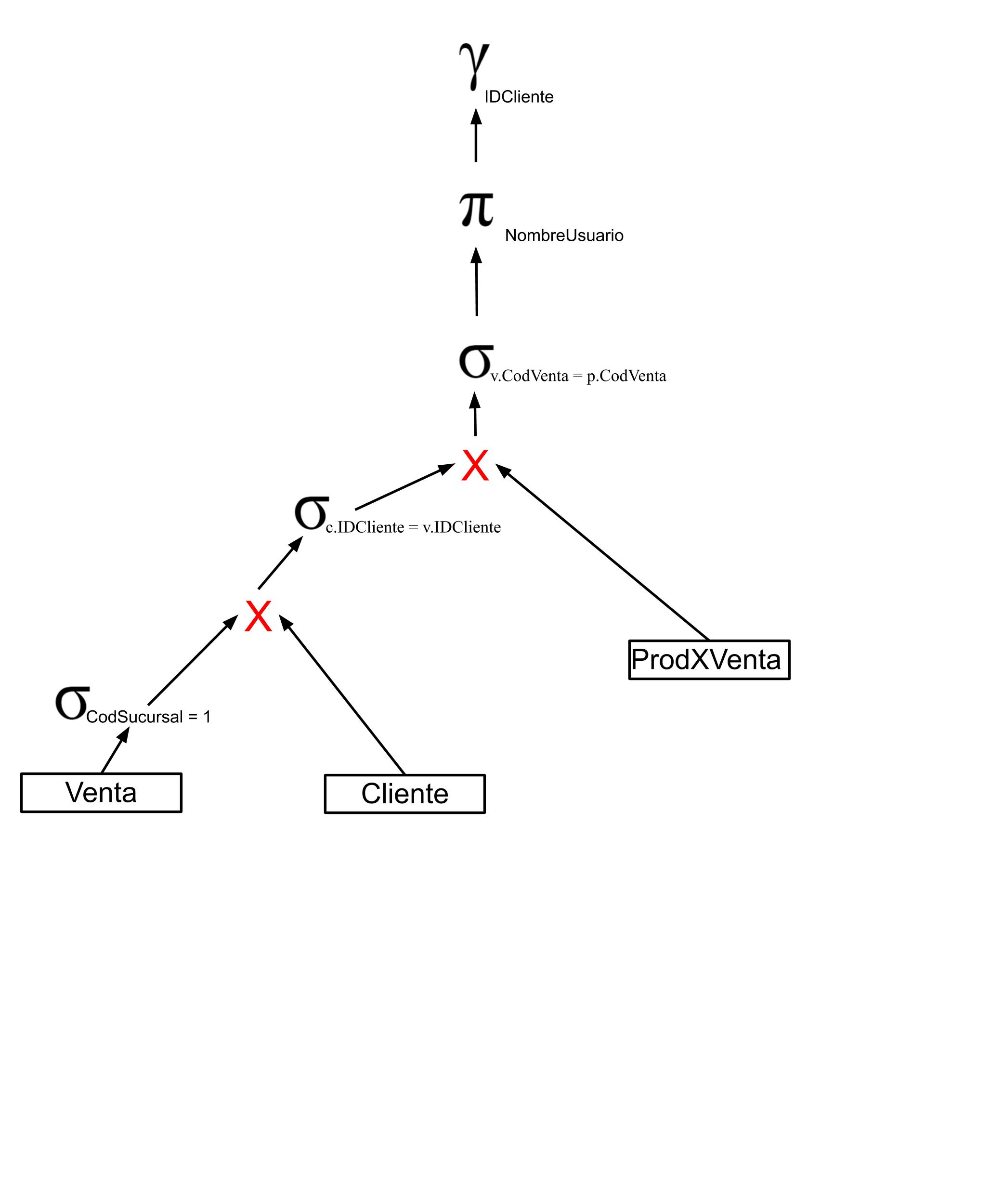
Uso leyes 4 a 8 para mover cada selección tan abajo como sea posible en el arbol.



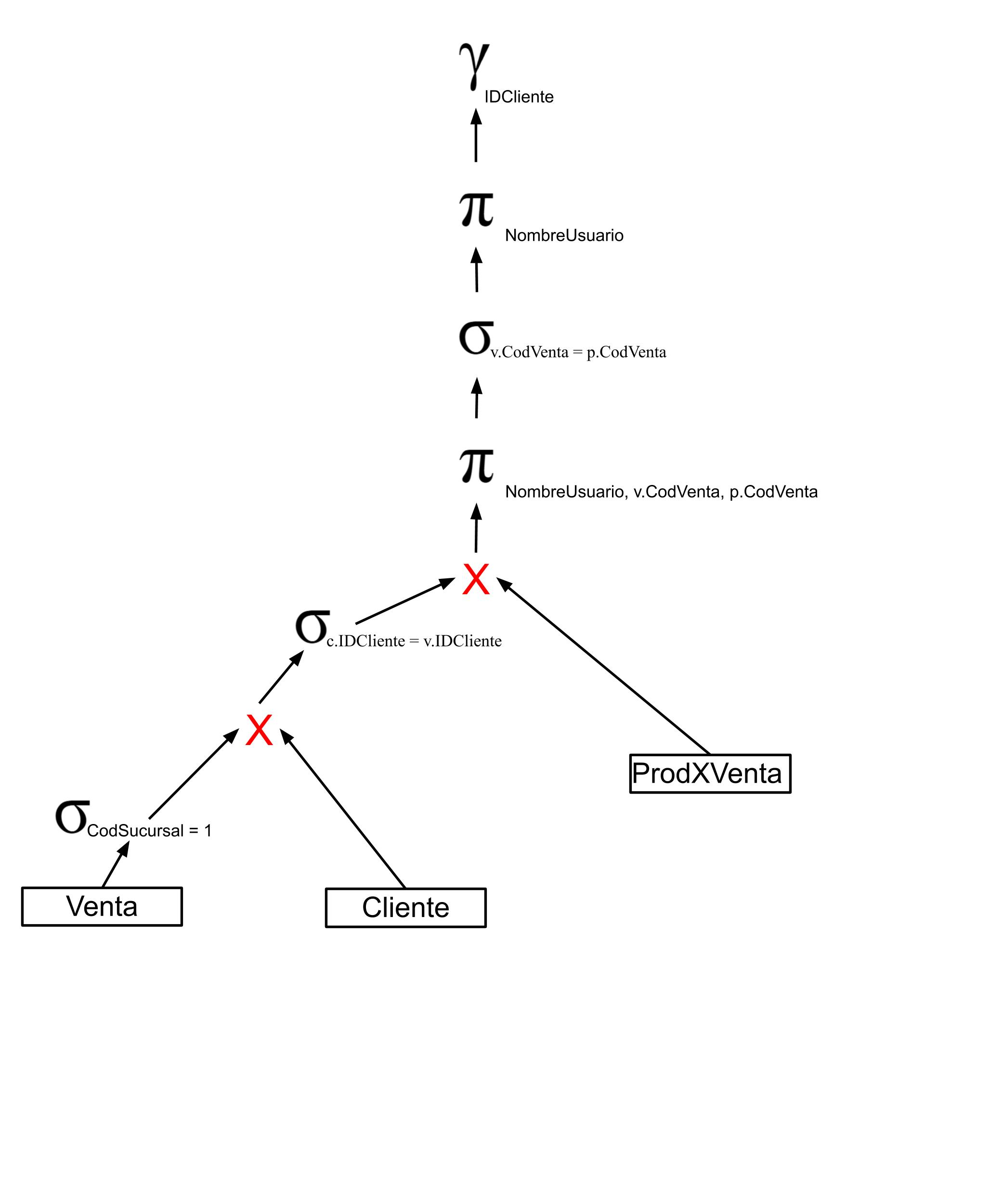
**4.**

Uso reglas 3, 5, 10 y 11 para mover cada proyección tan abajo en el árbol como sea posible.

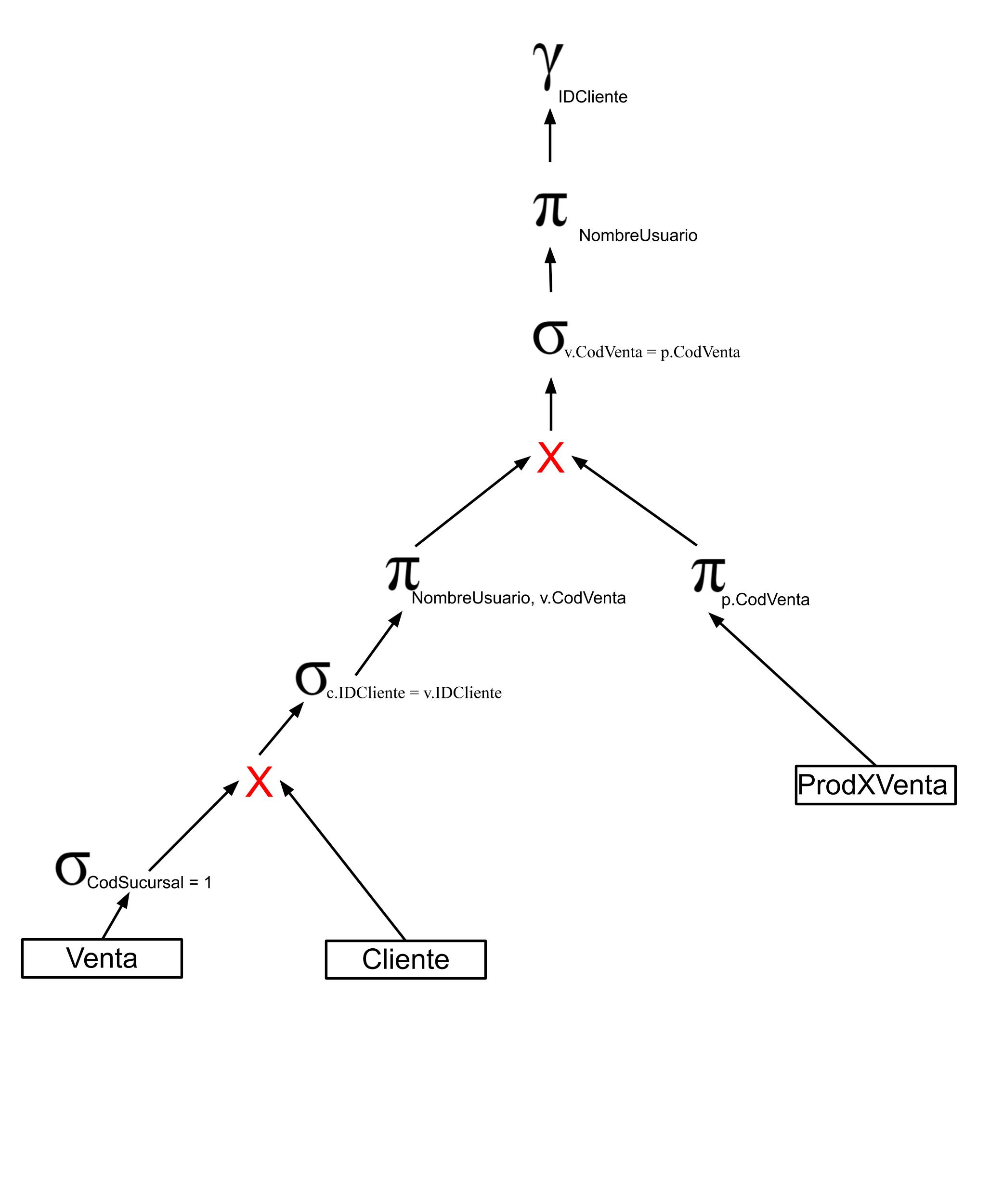
* Uso regla 3 para suprimir la proyección más intern



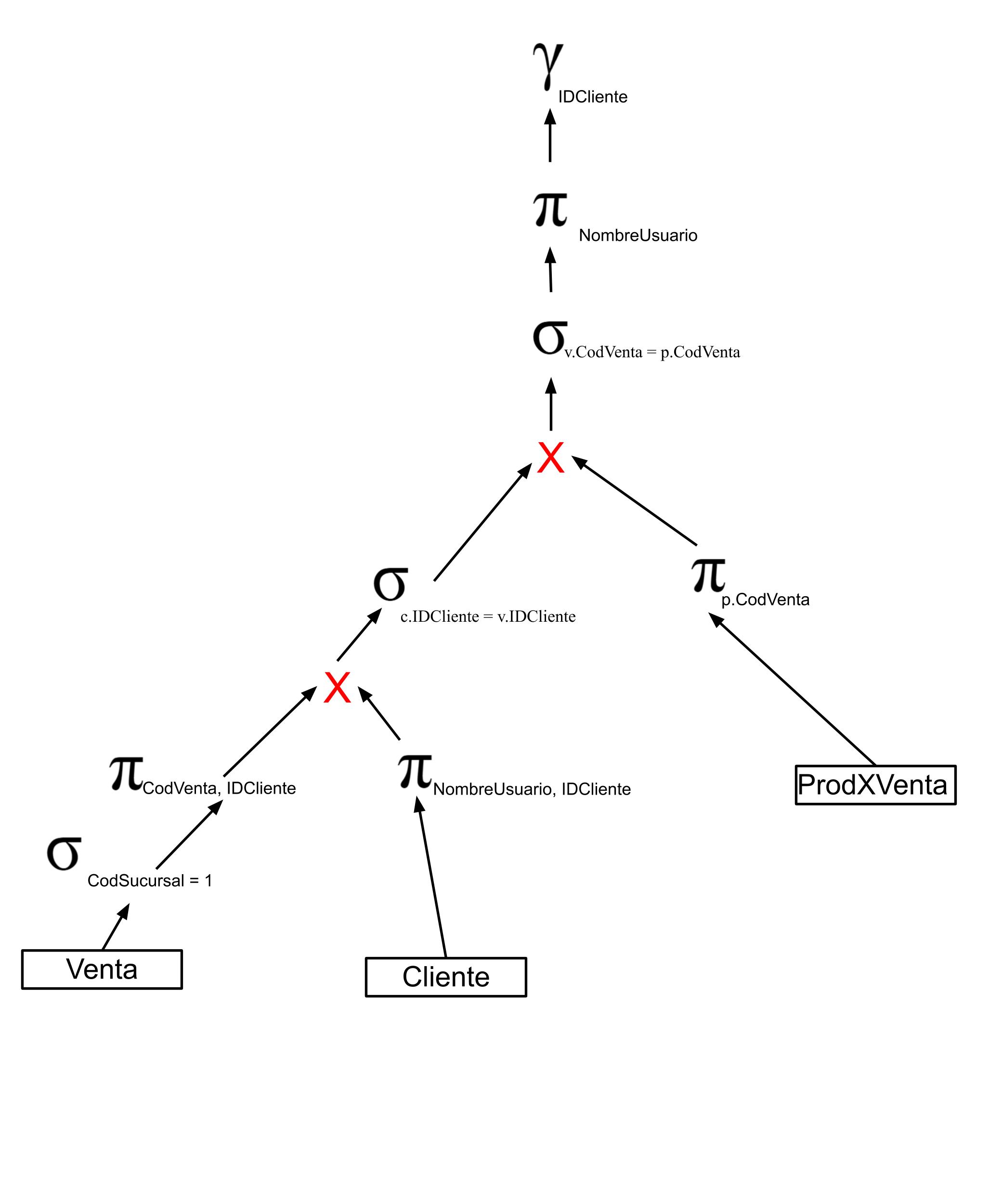
* Se agrega una proyección con los atributos que serán necesarios posteriormente



* Uso regla 10 para conmutar la proyección con el con el producto cartesiano

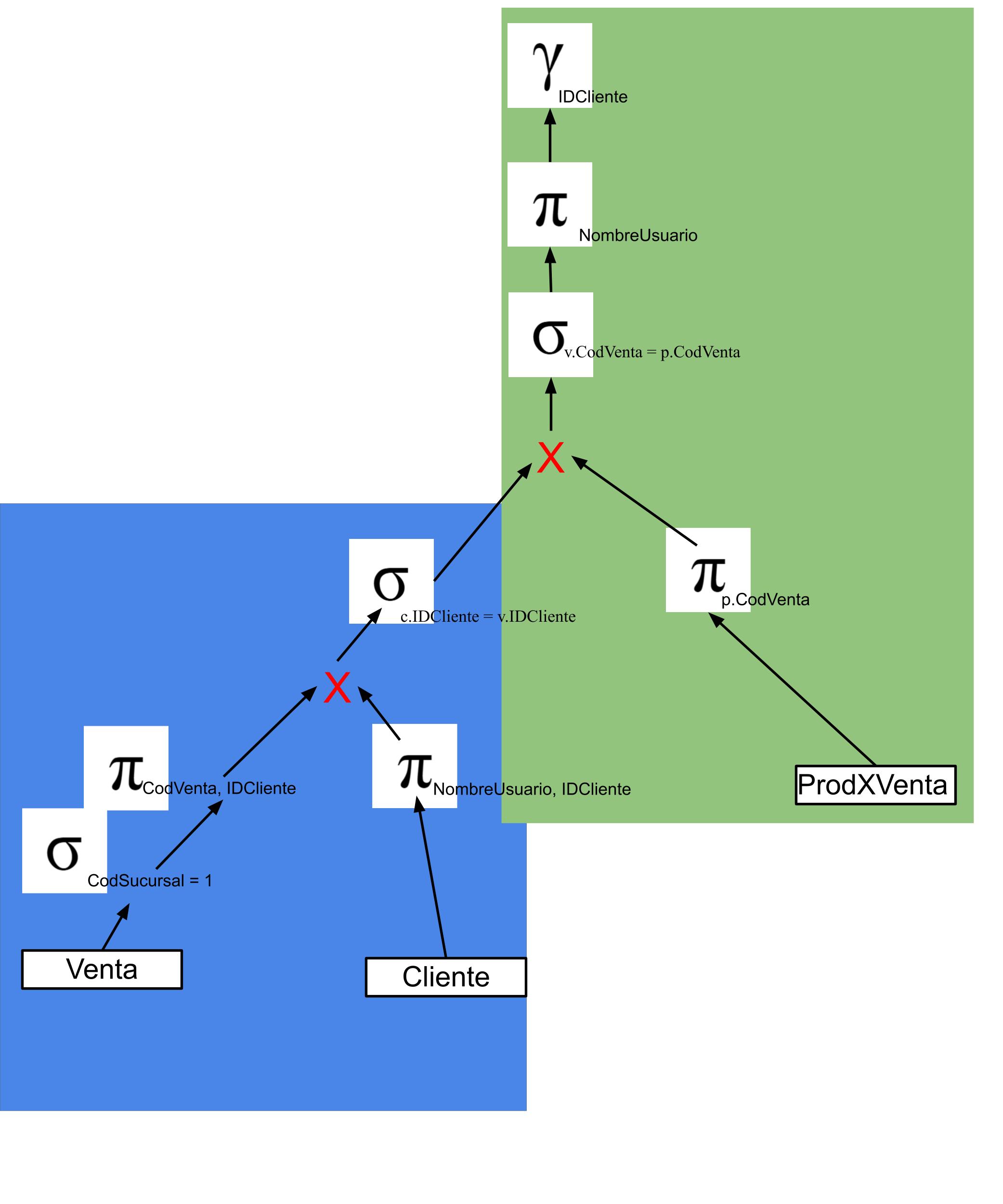


* Uso reglas 5.2 y 10 para mover la proyección tan abajo en el árbol como sea posible



**6.**

Particionar los nodos interiores del árbol resultante en grupos



πNombreUsuario**(**σventa.CodVenta=ProdxVenta.CodVenta**((**σcliente.IDCliente=venta.IDCliente**(**πCodVenta, IDCliente**(**σCodSucursal=1**(**Venta**)))x(**πNombreUsuario, IDCliente**(**Cliente**)))x(**πCodVenta**(**ProdXVenta**))))**