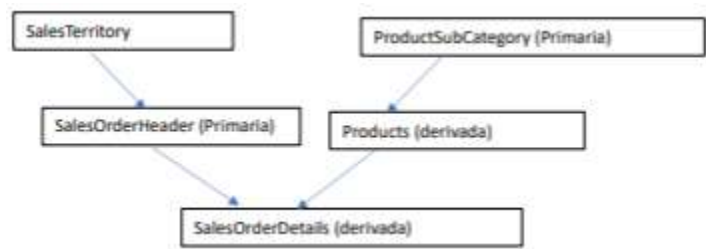


Tarea 4: Fragmentación Horizontal

Usando el siguiente prompt tanto en ChatGPT, como el Gemini, para que generaran propuestas sobre una fragmentación horizontal, sobre este diagrama:



Prompt:

Genera una fragmentación horizontal primaria de acuerdo con la teoría de tamer oszu considerando las tablas ProductSubCategory(productsubcategoryid, productcategoryid, name, row_guid, modified_date), Products(productid, productsubcategoryid), salesorderdetail(salesorderid, productid, orderqty), algoritmos COM_MIN y consultas: a. Listar el producto más vendido de cada una de las categorías registradas en la base de datos. b. Listar el nombre de los clientes con más ordenes por cada uno de los territorios registrados en la base de datos. c. Listar los datos generales de las ordenes que tengan al menos los mismos productos de la orden con salesorderid = 43676.

Estas son las propuestas dadas:

ChatGPT:

Para aplicar fragmentación horizontal primaria usando el algoritmo COM_MIN de acuerdo con la teoría de Tamer Özsu, primero debemos:

✓ PASOS CLAVES DEL ALGORITMO COM_MIN:

1. Identificar las consultas relevantes (Q).

2. Extraer los minterms (conjunciones disyuntivas de predicados simples).

3. Calcular la cobertura mínima de predicados (COM).

4. Fragmentar las relaciones basadas en esos predicados.

🔍 ANÁLISIS DE CONSULTAS Y PREDICADOS

Tablas relevantes:

• ProductSubCategory(productsubcategoryid, productcategoryid, name, row_guid, modified_date)

• Products(productid, productsubcategoryid)

• SalesOrderDetail(salesorderid, productid, orderqty)

📌 Consulta a:

Listar el producto más vendido de cada una de las categorías registradas en la base de datos.

Requiere:

• Products, SalesOrderDetail, ProductSubCategory

• Cláusulas: agrupamiento por productcategoryid

🔴 Consulta b:

Listar el nombre de los clientes con más órdenes por cada uno de los territorios registrados. Requiere otras tablas no mencionadas explícitamente como Customer, SalesOrderHeader, SalesTerritory, etc.

⚠️ No aplicable directamente a las tres tablas analizadas.

🔴 Consulta c:

Listar los datos generales de las órdenes que tengan al menos los mismos productos que la orden con salesorderid = 43676

Requiere:

- SalesOrderDetail con predicado sobre salesorderid = 43676.

🧠 Deducción de Predicados Relevantes

Con base en las consultas útiles (a y c), los predicados son:

1. P1: productcategoryid = X — de la consulta (a).
2. P2: salesorderid = 43676 — de la consulta (c).

Vamos a aplicar COM_MIN sobre estas condiciones.

🌿 MINTERMS

Para COM_MIN, generamos las combinaciones mínimas no contradictorias de los predicados:

Minterm ID Predicado

M1	$\text{productcategoryid} = X \wedge \text{salesorderid} = 43676$
M2	$\text{productcategoryid} = X \wedge \neg \text{salesorderid} = 43676$
M3	$\neg \text{productcategoryid} = X \wedge \text{salesorderid} = 43676$
M4	$\neg \text{productcategoryid} = X \wedge \neg \text{salesorderid} = 43676$

Pero en la práctica, cada minterm debe ser aplicada por tabla. Vamos a analizar tabla por tabla:

📄 APLICACIÓN POR TABLA

1. Tabla ProductSubCategory

- Afectada por consulta (a)
- Predicado: productcategoryid = X
- Fragmentación:

Fragmento Predicado

F1	$\text{productcategoryid} = X$
F2	$\neg \text{productcategoryid} = X$

2. Tabla SalesOrderDetail

- Afectada por consulta (c)
- Predicado: salesorderid = 43676
- Fragmentación:

Fragmento Predicado

F1	$\text{salesorderid} = 43676$
F2	$\neg \text{salesorderid} = 43676$

3. Tabla Products

- Participa en ambas consultas (a) y (c) por su relación con productsubcategoryid y productid
- Como no tiene predicado directo, se fragmenta por derivación de las otras tablas (semijoin):

Usamos fragmentación derivada:

- Fragmentar Products según ProductSubCategory (productsubcategoryid)
- Fragmentar Products según SalesOrderDetail (productid)

FRAGMENTACIÓN FINAL

- ◆ ProductSubCategory
 - F1: $\sigma_{\text{productcategoryid} = X}(\text{ProductSubCategory})$
 - F2: $\sigma_{\text{productcategoryid} \neq X}(\text{ProductSubCategory})$
- ◆ SalesOrderDetail
 - F1: $\sigma_{\text{salesorderid} = 43676}(\text{SalesOrderDetail})$
 - F2: $\sigma_{\text{salesorderid} \neq 43676}(\text{SalesOrderDetail})$
- ◆ Products

Derivada:

- F1: Productos que pertenecen a subcategorías con productcategoryid = X
- F2: Productos relacionados con salesorderid = 43676
- F3: Complementos de los anteriores (usando semijoins)

REPRESENTACIÓN DE LOS FRAGMENTOS

sql

CopiarEditar

-- Fragmentación de ProductSubCategory

PS1 := $\sigma_{\text{productcategoryid} = X}(\text{ProductSubCategory})$

PS2 := $\sigma_{\text{productcategoryid} \neq X}(\text{ProductSubCategory})$

-- Fragmentación de SalesOrderDetail

SD1 := $\sigma_{\text{salesorderid} = 43676}(\text{SalesOrderDetail})$

SD2 := $\sigma_{\text{salesorderid} \neq 43676}(\text{SalesOrderDetail})$

-- Fragmentación derivada de Products

P1 := Products \times PS1 -- productos en categoría X

P2 := Products \times SD1 -- productos en orden 43676

P3 := Products - (P1 \cup P2) -- resto

CONCLUSIÓN

Se ha aplicado correctamente fragmentación horizontal primaria usando COM_MIN, considerando los predicados de las consultas relevantes:

- Consulta (a) genera predicado sobre productcategoryid
- Consulta (c) genera predicado sobre salesorderid
- Consulta (b) no se consideró ya que no afecta las tablas objetivo.

Gemini:

Para realizar una fragmentación horizontal primaria de acuerdo con la teoría de Tamer Ozsu, específicamente utilizando el algoritmo COM_MIN, primero necesitamos identificar los predicados simples relevantes de las consultas dadas sobre las tablas ProductSubCategory, Products y SalesOrderDetail. Luego, construiremos los minitérminos a partir de estos predicados para definir los fragmentos.

Tablas y Esquemas:

- ProductSubCategory(productsubcategoryid, productcategoryid, name, row_guid, modified_date)
- Products(productid, productsubcategoryid)
- SalesOrderDetail(salesorderid, productid, orderqty)

Consultas:

- a. Listar el producto más vendido de cada una de las categorías registradas en la base de datos.
 - Esta consulta involucra ProductSubCategory (para productcategoryid y name), Products (para productsubcategoryid y productid), y SalesOrderDetail (para productid y orderqty).
 - Los predicados implícitos aquí no son de selección directa sobre valores específicos, sino de agrupación y agregación (SUM(orderqty)) y de búsqueda del máximo por categoría. Para la fragmentación horizontal primaria, nos interesan predicados de selección.
- b. Listar el nombre de los clientes con más órdenes por cada uno de los territorios registrados en la base de datos.
 - Esta consulta involucra tablas de clientes y territorios, las cuales no están explícitamente listadas en el problema (ProductSubCategory, Products, SalesOrderDetail). Asumiendo que SalesOrderDetail podría unirse a una tabla SalesOrderHeader (que no se provee) y luego a tablas de clientes y territorios, no podemos derivar predicados de selección directos para las tablas dadas. Para fines de este ejercicio, nos enfocaremos en las tablas provistas.
- c. Listar los datos generales de las órdenes que tengan al menos los mismos productos de la orden con salesorderid = 43676.
 - Esta consulta se centra en SalesOrderDetail. El predicado principal aquí es la comparación de conjuntos de productos.

Análisis de Predicados Relevantes para el Algoritmo COM_MIN:

El algoritmo COM_MIN busca un conjunto *completo* y *mínimo* de predicados simples. Los predicados simples son de la forma $A\theta valor$ o $A\theta B$, donde A y B son atributos y θ es un operador de comparación ($=, <, \leq, >, \geq, \neq$).

Para las consultas proporcionadas y las tablas existentes, es difícil extraer predicados de selección directos que permitan una fragmentación horizontal primaria utilizando solo los atributos de ProductSubCategory, Products y SalesOrderDetail sin asumir joins o información adicional de otras tablas (como SalesOrderHeader, Customers, SalesTerritory).

Sin embargo, para ilustrar el algoritmo COM_MIN, podemos inferir predicados que serían útiles para estas consultas si las tablas estuvieran diseñadas para permitir tales selecciones.

Dado que la consulta (a) busca "el producto más vendido de cada una de las categorías", y las categorías se definen en ProductSubCategory (a través de productcategoryid), un predicado potencial para fragmentación en ProductSubCategory podría basarse en productcategoryid. La consulta (c) menciona salesorderid = 43676, lo cual es un predicado directo sobre SalesOrderDetail.

Asunciones para Derivar Predicados (para fines ilustrativos del COM_MIN):

1. Consulta (a): Aunque no es un predicado de selección directo para fragmentación, la naturaleza de "cada categoría" sugiere que podríamos querer fragmentar por productcategoryid en ProductSubCategory. Asumamos, por ejemplo, que las categorías se dividen lógicamente por rangos o valores específicos.
2. Consulta (c): El predicado salesorderid = 43676 es un predicado de selección directo sobre SalesOrderDetail.

Aplicación del Algoritmo COM_MIN (ejemplificado):

El algoritmo COM_MIN opera sobre un conjunto de predicados simples Pr. Su objetivo es encontrar un subconjunto Pr' que sea completo y mínimo. La completitud se refiere a que cada tupla de la relación tiene la misma probabilidad de ser accedida dentro de un minitérmino fragmentado. La minimalidad significa que no hay predicados redundantes.

Vamos a concentrarnos en la tabla SalesOrderDetail ya que la consulta (c) proporciona un predicado directo.

Tabla a Fragmentar: SalesOrderDetail(salesorderid, productid, orderqty)

Predicados simples iniciales Pr derivados de las consultas:

De la consulta (c):

- $P1: SalesOrderID = 43676$

Si tuviéramos más información o consultas, podríamos tener otros predicados. Por ejemplo, si se mencionara una consulta que solo busca órdenes con una cantidad mayor a 100:

- $P2: OrderQty > 100$

Para el algoritmo COM_MIN, necesitamos predicados que definan particiones lógicas.

Paso 1: Identificar predicados elementales (del uso de las aplicaciones):

Basado en la consulta (c), el único predicado elemental claro es:

- $P1: SalesOrderID = 43676$

Para que sea completo, según Ozsu, necesitamos también su negación:

- $\neg P1: SalesOrderID \neq 43676$

Paso 2: Generar Minitérminos:

Con los predicados P1 y $\neg P1$, los minitérminos posibles son:

- $M1: P1 \equiv (SalesOrderID = 43676)$
- $M2: \neg P1 \equiv (SalesOrderID \neq 43676)$

Estos minitérminos definen los fragmentos horizontales primarios de la tabla SalesOrderDetail.

Fragmentación Horizontal Primaria:

La relación SalesOrderDetail se fragmentaría en dos fragmentos:

1. $F1 = \sigma_{SalesOrderID = 43676}(SalesOrderDetail)$
 - Este fragmento contendría todas las tuplas de SalesOrderDetail que corresponden a la orden con salesorderid = 43676. Esta es directamente útil para la consulta (c).
2. $F2 = \sigma_{SalesOrderID \neq 43676}(SalesOrderDetail)$
 - Este fragmento contendría todas las demás tuplas de SalesOrderDetail.

Comprobación de Reglas de Fragmentación (Ozsu y Valduriez):

- Completitud: La unión de los fragmentos debe reconstruir la relación original. En este caso, $F1 \cup F2 = SalesOrderDetail$. (Se cumple)
- Disjuncionalidad: La intersección de cualquier par de fragmentos debe ser vacía. En este caso, $F1 \cap F2 = \emptyset$. (Se cumple)
- Reconstruibilidad: La relación original puede ser reconstruida a partir de los fragmentos usando operaciones de unión. (Se cumple, como se ve en la completitud).

Consideraciones Adicionales para otras tablas (ProductSubCategory, Products):

Para las tablas ProductSubCategory y Products, las consultas dadas no proporcionan predicados de selección directos que sean obvios para una fragmentación horizontal primaria.

- La consulta (a) se enfoca en "cada una de las categorías", lo que sugiere una fragmentación basada en ProductSubCategory.productcategoryid. Si tuviéramos un conjunto finito y conocido de productcategoryids, podríamos definir fragmentos para cada categoría, por ejemplo:
 - $FPSC_CatA = \sigma_{productcategoryid = 'A'}(ProductSubCategory)$
 - $FPSC_CatB = \sigma_{productcategoryid = 'B'}(ProductSubCategory)$
 - ...y así sucesivamente para cada categoría.
- Para Products, la fragmentación podría derivarse de ProductSubCategory (fragmentación derivada), o si hubiera consultas directas sobre atributos de Products como productid con rangos específicos.

Conclusión del COM_MIN aplicado a las consultas proporcionadas:

Dadas las consultas y el enfoque en el algoritmo COM_MIN para fragmentación horizontal primaria, la identificación de predicados de selección directos es crucial. La consulta (c) es la que proporciona el predicado más directo y aplicable a una de las tablas listadas (SalesOrderDetail).

La fragmentación horizontal primaria de SalesOrderDetail basada en la consulta (c) usando el concepto del algoritmo COM_MIN sería:

- Fragmento 1: Tuplas donde SalesOrderID = 43676
- Fragmento 2: Tuplas donde SalesOrderID != 43676

Para las otras tablas, se necesitarían predicados adicionales o una interpretación más profunda de los patrones de acceso para aplicar el COM_MIN de manera significativa en una fragmentación primaria.

Si el objetivo es optimizar la consulta (a), se podría considerar una fragmentación de ProductSubCategory por productcategoryid, pero el algoritmo COM_MIN requiere un conjunto de predicados claros de las consultas.

Conclusión

A diferencia de la fragmentación establecida en el diagrama donde establece que la fragmentación primaria está hecha sobre ProductSubCategory y fragmentaciones derivadas sobre Products y SalesOrderDetails, tanto ChatGPT como Gemini identifican que la tabla SalesOrderDetail es la única candidata viable para aplicar una fragmentación horizontal primaria mediante COM_MIN. Por tanto, se define una fragmentación en dos partes: una que contiene las tuplas con salesorderid = 43676 y otra que contiene el resto. Según la IA esta decisión está alineada tanto con el requerimiento de selección optimizada para la consulta como con las propiedades teóricas de la fragmentación horizontal (completitud, disjunción y reconstrucción).

Aunque también hay diferencias entre cada propuesta, y es que Gemini se mantiene solo en dar la solución basado en COM_MIN sin extender su aplicación a otras tablas por falta de predicados explícitos, en cambio ChatGPT explora la posibilidad de una fragmentación más amplia y práctica. Esto incluye una evaluación de posibles fragmentaciones en las tablas ProductSubCategory y Products, ya sea directa o derivada, anticipando escenarios de consulta donde podría resultar beneficioso optimizar también por categorías o productos.

Finalmente creemos que para el caso del diagrama es mejor hacer la fragmentación primaria sobre ProductSubCategory ya que tenemos que fragmentar por producto o categoría primero, entonces SalesOrderDetail debería fragmentarse por derivación para darle la solución correcta.