OPTIMIZACIÓN

Historia de Condiciones Climatológicas y Ambientales en el Valle de Aburrá

Felipe Rodríguez Ángel - Juan Luis Rojas Rincón

1. System R.

DATOS DE ENTRADA:

Algunos de estos datos de entrada (Tuplas y tuplas x bloque) son tomados de los cálculos y proyecciones realizados para la entrega del diseño físico, no se mostrarán otra vez cómo fueron realizados los cálculos por simplicidad pero estos pueden ser accedidos en **este link a las Hojas de Cálculo.**

TABLA	REPORTE
T_R (tuplas relación)	640,618,403
B_R (bloques en los que se acomodan las tuplas de R)	49,278,339
Esquema de Almacenamiento	HEAP
$I_{R\text{ -ID\_Reporte}}$	640,618,403
$I_{R\text{ - Verificacion}}$	3
$I_{R\text{ - Zona\_codigo\_zona, Fecha}}$	355,899,112

INDICES REPORTE

Nombre del Índice	Tabla asociada	Tamaño páginas del índice	Modalidad de índice B+, Bitmap, etc.	Altura de árbol (si aplica)	Columnas que conforman el índice
I_reporte	REPORTE	2186411	B+	3.569	ID_reporte
I_reporte_verificación	REPORTE	2965825	B+	2.772	Verificación
I_reporte_fecha_zona	REPORTE	4187048	B+	3.031	Zona_codigo_zona, Fecha

ESTIMACIÓN DE LAS IMÁGENES

- $\mathbf{I_{Reporte\ -ID\_Reporte}}$

  Se estima que el número esperado de diferentes valores de ID_Reporte sea igual al número de tuplas en la Tabla, esto debido a que es un índice sobre la clave primaria de la relación por lo que todos los valores son distintos.

  Así, tomando la proyección a 20 años de los registros en la Tabla Reporte:

  $\mathbf{I_{Reporte\ -ID\_Reporte}} = 640{,}618{,}403$

- $\mathbf{I_{Reporte\ -Verificacion}}$

  Se estima que el número esperado de diferentes valores de *Verificacion* sea igual a 3, esto es debido a que se tiene un constraint en este atributo que sólo permite ingresar como valores ¨En Verificación¨, ¨Válido¨, ¨Inválido¨.

  Así, tomando la proyección a 20 años de los registros en la Tabla Reporte:

  $\mathbf{I_{Reporte\ -Verificacion}} = 3$

- $\mathbf{I}_{\text{Reporte -fecha,zona\_codigo\_zona}}$

  Potencialmente, podría tenerse un número de *fecha, Zona_codigo_zona* distintas según los segundos del día y el número de zonas ya que cada zona es distinta aunque se tenga registros en la misma hora.

  Sin embargo este sería el caso más extremo, se estima que se ingresa un nuevo reporte al sistema cada 1,8 minutos. Así,dividimos el máximo número de registros en 20 años sobre la estimación de reportes nuevos:

  $\mathbf{I}_{\text{Reporte -fecha-codigo\_zona}} = \frac{640618403}{1.8} = 355{,}899{,}112$

CONSULTA

```
SELECT Id_reporte, nombre_autor
FROM REPORTE
WHERE
Verificacion = 'Valido'
AND Fecha <= to_date('08-06-2010 23:59:59', 'dd-MM-yyyy hh24:mi:ss')
AND Fecha > to_date('08-06-2009 23:59:59', 'dd-MM-yyyy hh24:mi:ss')
AND ZONA_codigo_zona = 'AB123'
```

Esta consulta busca obtener la Id del reporte y el nombre del autor de todos los reportes con Verificación "Valido"que estén entre el 08-06-2010 y el 09-06-2009. Podemos aplicar **SYSTEM R** en esta consulta porque **las condiciones están descompuestas tanto como es posible en condiciones conectadas por AND**.

Clúster
No Clúster

OPCIÓN	DESCRIPCIÓN	COSTO (Formula)	COSTO CALCULADO
OPCIÓN 1	Obtener tuplas de *R* que satisfacen una condición de la forma *A*=*c*. Donde *A* tiene un índice clúster. Sea I_R el tamaño de la imagen del índice;	Costo = **B/I**	No aplica pues no se tienen Índices Clúster.
OPCIÓN 2	Usar un índice clúster en un atributo *A*, donde *A* θ *C* es una de las condiciones C_i y $\theta \in$ (<, <= , >, >=). Luego aplicamos las condiciones restantes al resultado. En promedio leemos:	Costo = $\mathbf{B_R / 2}$	No aplica pues no se tienen Índices Clúster.
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.	Costo = $\mathbf{T_R / I_R}$	En Verificación se tiene un Índice que cumple estas características. 640,618,403 / 3 = 213,539,468
OPCIÓN 4	Si *R* está almacenado en un archivo independiente, podemos simplemente leer todas las tuplas y aplicar las C_i a éstas.	Costo = $\mathbf{B_R}$	49,278,339
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.	Costo = $\boldsymbol{B_R}$	No aplica pues no se posee un Índice Clúster
OPCIÓN 6	Si hay un índice no clúster en un atributo *A* y *A* θ *C* es una condición donde $\theta \in$ (< , > , >= , <=) ; use el índice para obtener las tuplas de R que satisfagan la condición y aplique las otras condiciones al resultado.	Costo = $\mathbf{T_R / 2}$	640,618,403/2 = 320 309 202
OPCIÓN 7	Use un índice no clúster de cualquier clase para hallar las tuplas de *R* y aplicar todas las condiciones a ellas.	Costo = $\mathbf{T_R}$	640,618,403
OPCIÓN 8	Si ninguna de las anteriores opciones es posible, simplemente recupere todos los bloques que podrían contener tuplas de R.	Costo = $\mathbf{T_R}$	640,618,403

Se observa en la tabla anterior que la **Opción 4** posee el menor costo de todas las opciones, por lo tanto se elige esta.

Podemos elegir la **Opción 4** porque:

- **Reporte está almacenada en un archivo independiente**

En esta estrategia se tiene un Costo de $\mathbf{B_R}$ porque este es el número de bloques en los que se encuentran las Tuplas de la relación, y la forma de ejecutarla es que se aplican las Condiciones que involucran a la Relación en las Tuplas de esta.

2. JOIN entre dos Tablas

A continuación se evaluarán los Costos de Entrada en diferentes escenarios y proyecciones de magnitud en las tablas SENSOR y MEDIDA. El crecimiento de las Tablas está basado en el número de nuevos registros calculados por semana, no se ampliará en este cálculo pues puede ser encontrado en la hoja de cálculo adjunta, pero sí se mostrarán los resultados.

Por otro lado, se definieron los bloques (páginas) con un tamaño de 8192 bytes, se define entonces que se puede tener en memoria **dos bloques de 8192 bytes**.

La Memoria será constante para todos los Escenarios y Casos. Se define **M = 2**.

Variables

T_R → Número de Tuplas en la Tabla Mayor (R).
B_R → Número de Bloques en los que pueden ser almacenadas las Tuplas de la Tabla R.
T_S → Número de Tuplas en la Tabla Menor (S).
B_s → Número de Bloques en los que pueden ser almacenadas las Tuplas de la Tabla S.

$I_{R\text{-}b} = J_b \rightarrow$ Imágen de la Tabla R en el atributo b.
$I_{S\text{-}b} = I_b \rightarrow$ Imágen de la Tabla S en el atributo b.

Constante

M = Número de Bloques en memoria principal = 2
R = Tabla Mayor (Medida)
S = Tabla Menor (Sensor)

2.1 ESCENARIO 1.

CONSULTA:

```
SELECT  *
FROM SENSOR S JOIN MEDIDA R
ON
S.serial = R.sensor_serial
```

Para cumplir con los requisitos de este escenario, no utilizaremos los índices que poseen los atributos serial y Sensor_Serial. Además, a pesar de que estas tablas están definidas en un clúster conjunto, para este escenario las tomaremos como SENSOR en una estructura HEAP y MEDIDA como una estructura HEAP.

El número de seriales distintos en el atributo S.serial es igual al número de Tupĺas en la tabla, esto debido a que se tiene esta como la Clave primaria.

Por otro lado, debido a que la Tabla MEDIDA está dada por una relación débil con SENSOR (su clave primaria posee un atributo de la clave foránea, **sensor_serial**), y en el Modelo del Negocio se tiene la posibilidad de que existan sensores que no han tomado

medidas ambientales (los cuales se estiman en un 0.7% del total de sensores), tenemos que el atributo R.sensor_serial tiene como número de Tuplas distintas el número de sensores que se estima han tomado medidas, osea, el 99.3%.

En la Tabla a continuación se presentan los casos con los respectivos valores por Relación. Es importante aclarar que la columna semana representa la semana representa

		TABLA MEDIDA				TABLA SENSOR			
Casos	**Semana**	**T_R**	**Tamaño tabla Medida bytes**	**B_R**	**I_{R-b}**	**T_s**	**Tamaño tabla Sensor bytes**	**B_s**	**I_{S-b}**
1. 500.000 < Tr < 1.000.000	1	613,872	31,921,344	3,897	397	400	20,000	3	400
2. 5.000.000 < Tr < 8.000.000	9	5,524,848	287,292,096	35,070	3,575	3,600	180,000	22	3,600
3. 10.00.000 < Tr < inf	17	10,435,824	542,662,848	66,244	6,753	6,800	340,000	42	6,800

CASO 1

Datos de entrada por tabla	Estrategia Usada	Fórmula usada para costo de la entrada	Costo entrada.
Esquema de almacenamiento usado por tabla: HEAP Y HEAP Br=3,897 Bs=3 Tr=613,872 Ts=400 Imágenes de atributo con índices. IB=Is.b = 400 JB=Ir.b = 397 M=2	Selección en un producto	$B_R \times \frac{B_s}{M-1} + B_s$	11,694
	Sort Merge	$2B_R \times log_M(B_R) + 2B_s \times log_M(B_s) + B_R + B_s$	96,813
	Hash Join.	$(I_B + 2J_B \times log_M J_B) \times max(1, \frac{B_R}{J_B}) + max(I_B, B_s) \times (1 + 2 \times$	78,456

CASO 2:

Datos de entrada por tabla	Estrategia Usada	Fórmula usada para costo de la entrada	Costo entrada.
Esquema de almacenamiento usado por tabla: HEAP Y HEAP Br=35,070 Bs=22 Tr=5,524,848 Ts=3,600 Imágenes de atributo con índices. IB=Is.b = 3,600 JB=Ir.b = 3,575 M=2	Selección en un producto	$B_R \times \frac{B_s}{M-1} + B_s$	771,562
	Sort Merge	$2B_R \times log_M(B_R) + 2B_s \times log_M(B_s) + B_R + B_s$	1,093,700
	Hash Join.	$(I_B + 2J_B \times log_M J_B) \times max(1, \frac{B_R}{J_B}) + max(I_B, B_s) \times (1 + 2 \times$	950.810

CASO 3:

Datos de entrada por tabla	Estrategia Usada	Fórmula usada para costo de la entrada	Costo entrada.
Esquema de almacenamiento usado por tabla: HEAP Y HEAP Sensor: S Medida: R Br=66,244 Bs=42 Tr=10,435,824 Ts=6,800 Imágenes de atributo con índices. IB=Is.b = 6,800 JB=Ir.b = 6,753 M=2	Selección en un producto	$B_R \times \frac{B_s}{M-1} + B_s$	2,782,290
	Sort Merge	$2B_R \times log_M(B_R) + 2B_s \times log_M(B_s) + B_R + B_s$	2,187,871
	Hash Join.	$(I_B + 2J_B \times log_M J_B) \times max(1, \frac{B_R}{J_B}) + max(I_B, B_s) \times (1 + 2 \times$	1,930,171

CONCLUSIONES DEL ESCENARIO 1

Podemos observar que en el Caso 1 la Selección en un Producto es la estrategia con el costo de entrada menor, el Hash Join es la Estrategia con el segundo costo más bajo y el Sort Merge posee el costo de entrada más alto. Este caso particular se debe a la poca cantidad de Tuplas presentes en el Caso 1 y a la escasa memoria que se definió para la optimización. La cantidad de bloques en memoria es un factor de gran importancia para los Hash Join y Sort Merge, pues los costos involucran operaciones logarítmicas en base a la memoria.

Para los Casos 2 y 3, debido a que se tienen más tuplas los Costos de las estrategias Hash Join y Sort Merge mejoran en contraste con la Selección en un Producto, esto debido a que los costos de estas dos estrategias involucran operaciones logarítmicas.

Tomando en cuenta estos resultados se elige la Estrategia Hash Join. Se considera que esta es superior a la Selección en Producto porque los rangos de tuplas en los cuales este es menos costoso son muy cortos en relación a la vida útil de la aplicación (para la 9 semana el Hash Join ya es menos costoso, y se hacen proyecciones de al menos 20 años).

2.2 ESCENARIO 2

CONSULTA:

```
SELECT *
FROM SENSOR S JOIN MEDIDA R
ON
S.serial = R.sensor_serial
```

Se hace uso de la consulta utilizada en el Escenario 1, en este caso se mantienen las condiciones de magnitud del escenario anterior con la excepción de que sí **se tienen definidos índices en los atributos del JOIN** (S.serial y R.sensor_serial) y estos **Índices son de tipo Clúster**.

El Objetivo de este escenario es evaluar la distintas estrategias según la utilización de los Índices.

Es Importante recordar que la Memoria no cambia, sigue definiéndose que pueden tenerse 2 bloques de 8192 bytes en memoria

A continuación se especifican los valores de las variables según el caso:

		TABLA MEDIDA				TABLA SENSOR			
Casos	**Semana**	T_R	**Tamaño tabla Medida bytes**	B_R	$I_{R\text{-}b}$	T_s	**Tamaño tabla Sensor bytes**	B_s	$I_{S\text{-}b}$

1. 5.000.000 < Tr < 8.000.000	9	5,524,848	287,292,096	35,070	3,575	3,600	180,000	22	3,600
2. 10.00.000 < Tr < inf	17	10,435,824	542,662,848	66,244	6,753	6,800	340,000	42	6,800

Observación sobre la Fórmula para el JOIN usando un Índice

La fórmula para calcular el costo de entrada (**en S**) es $B_R + \frac{T_R B_S}{I}$, (esto con Índice clúster y relación compacta) pero esto sólo se cumple cuando I <= B_s debido a que B_s dividido la imágen no sería un estimado adecuado de los bloques si esto da como resultado un número menor a uno (como ya sabemos, siempre se debe acceder al menos a un bloque). Cuando I es Mayor o igual a B_s (I>=B_s) estaremos recuperando aproximadamente un bloque de S por cada Tupla de R, por lo que la Fórmula para calcular el Costo de Entrada cambia a B_R+T_R[1]

Caso 1.

Datos de entrada por tabla	**Estrategia Usada**	**Fórmula usada para costo de la entrada**	**Costo entrada.**
Cluster Individual: Medida. R Cluster Individual en Sensor .S Br=35,070 Bs=22 Tr=5,524,848 Ts=3,600	Usando índice en tabla R	$B_S + \frac{T_S B_R}{I}$ B_S+T_S	35,092

Imágenes de atributo con índices. IB=Is.b = 3,600 JB=Ir.b = 3,575 M=2	usando índice en tabla S	$B_R + \frac{T_R B_S}{I}$ B_R+T_R	5,559,848
	usando ambos índices	$I \times (max(1, \frac{B_R}{J}) + max(1,$	38,915

Caso 2.

Datos de entrada por tabla	**Estrategia Usada**	**Fórmula usada para costo de la entrada**	**Costo entrada.**
Las tablas Medida y Sensor poseen cada una un índice clúster Br=66,244 Bs=42 Tr=10,435,824 Ts=6,800 Imágenes de atributo con índices. IB=Is.b = 6,800 JB=Ir.b = 6,753	Usando índice en tabla R	$B_S + \frac{T_S B_R}{I}$ B_S+T_S	66,286
	usando índice en tabla S	$B_R + \frac{T_R B_S}{I}$ B_R+T_R	10,520,068

M=2	usando ambos índices	$I \times (max(1, \frac{B_R}{J}) + max(1,$	73,505

CONCLUSIONES DEL ESCENARIO 2

La estrategia de menor costo fue uniformemente el uso del Índice en la Tabla R en los distintos casos, por lo tanto se elige esta estrategia. Los costos usando ambos índices son similares a los que utilizan únicamente el Índice de la Tabla R, pero no son menores.

3. Optimización Algebraica

I. Defina una Vista SQL que involucra tres Tablas de su Sistema.

SQL CREACIÓN DE LA VISTA

CREATE VIEW
view_medicion_sensor_ubicacion
AS
SELECT
medida.Fecha_Hora, medida.valor, medida.sensor_serial, sensor.tipo_medida_Codigo_tipo_medida, ubicacion.cientifico_tipo_documento, ubicacion.documento, ubicacion.barrio_codigo_barrio, ubicacion.descripcion, ubicacion.latitud, ubicacion.longitud
FROM
MEDIDA, SENSOR, UBICACION
WHERE

medida.sensor_serial = sensor.serial **AND** sensor.ubicacion_latitud = ubicacion.latitud
AND sensor.ubicacion_longitud = ubicacion.longitud
;

Esta vista es una reunión natural entre las **Tablas MEDIDA, SENSOR Y UBICACIÓN.**

II. Defina una consulta sobre la vista que realizó

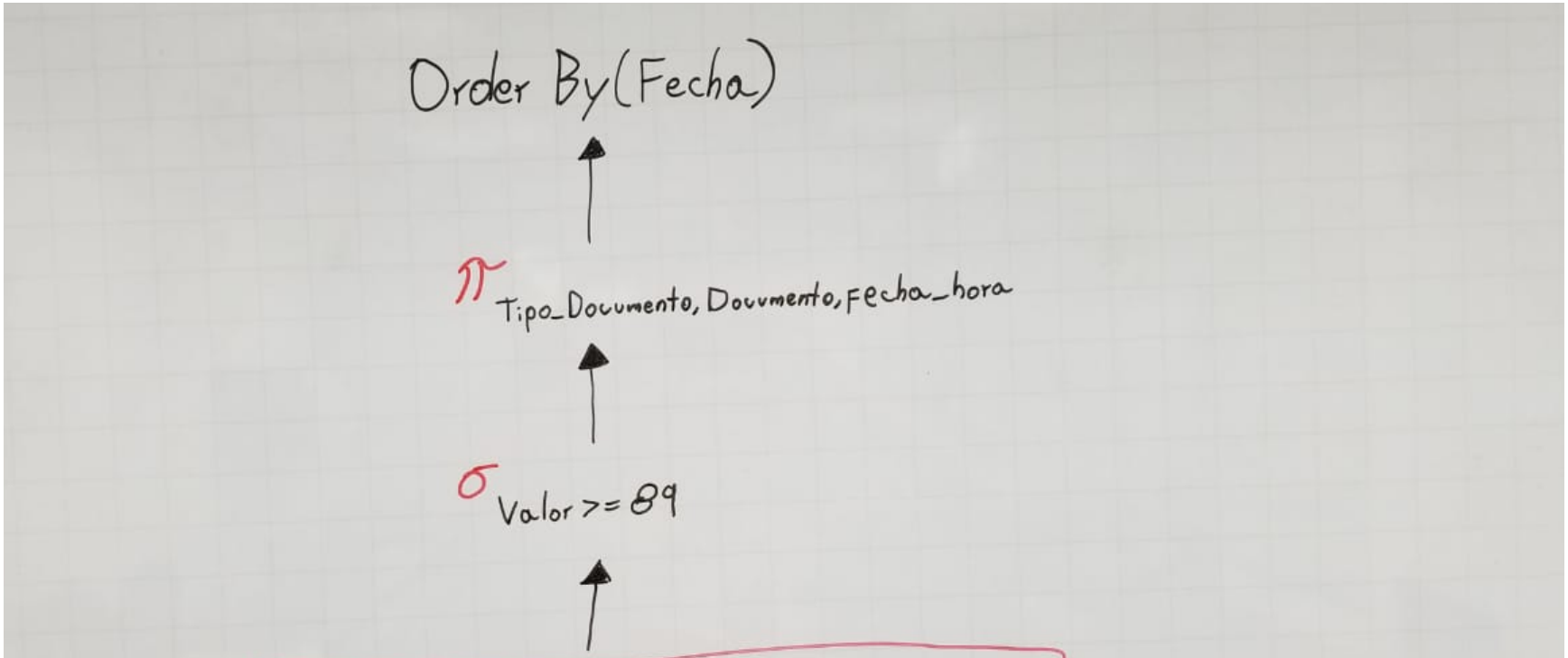
CONSULTA SQL

SELECT
tipo_documento,documento,fecha_hora
FROM
view_medicion_sensor_ubicacion
WHERE
Valor >= 89
ORDER BY
Fecha
;

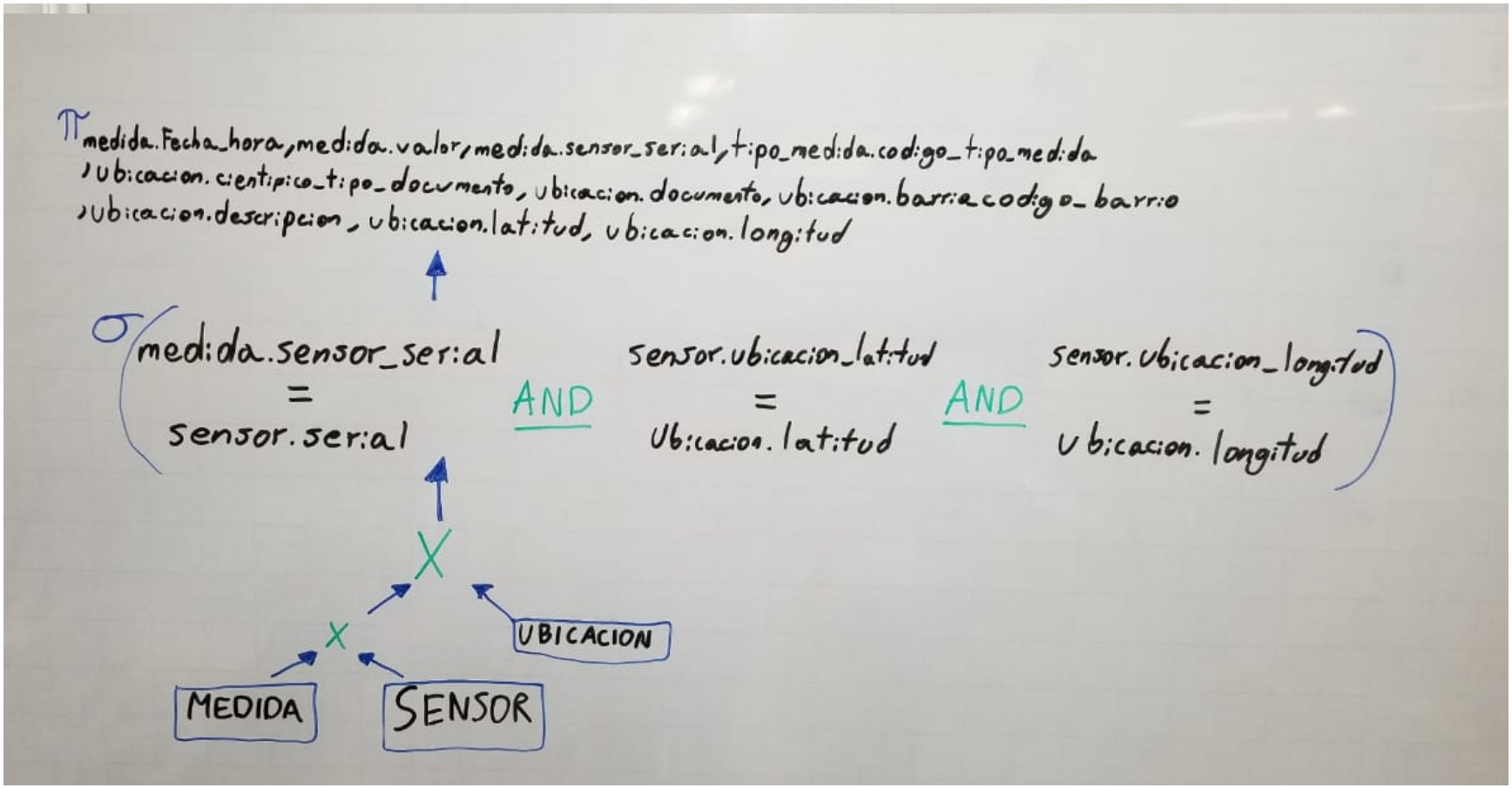
Con esta consulta obtenemos el Tipo de Documento y el Número de documento del científico responsable de las Ubicaciones en donde se han tenido mediciones con valores mayores o iguales a 89.

III. Expresión Algebraica y Optimización

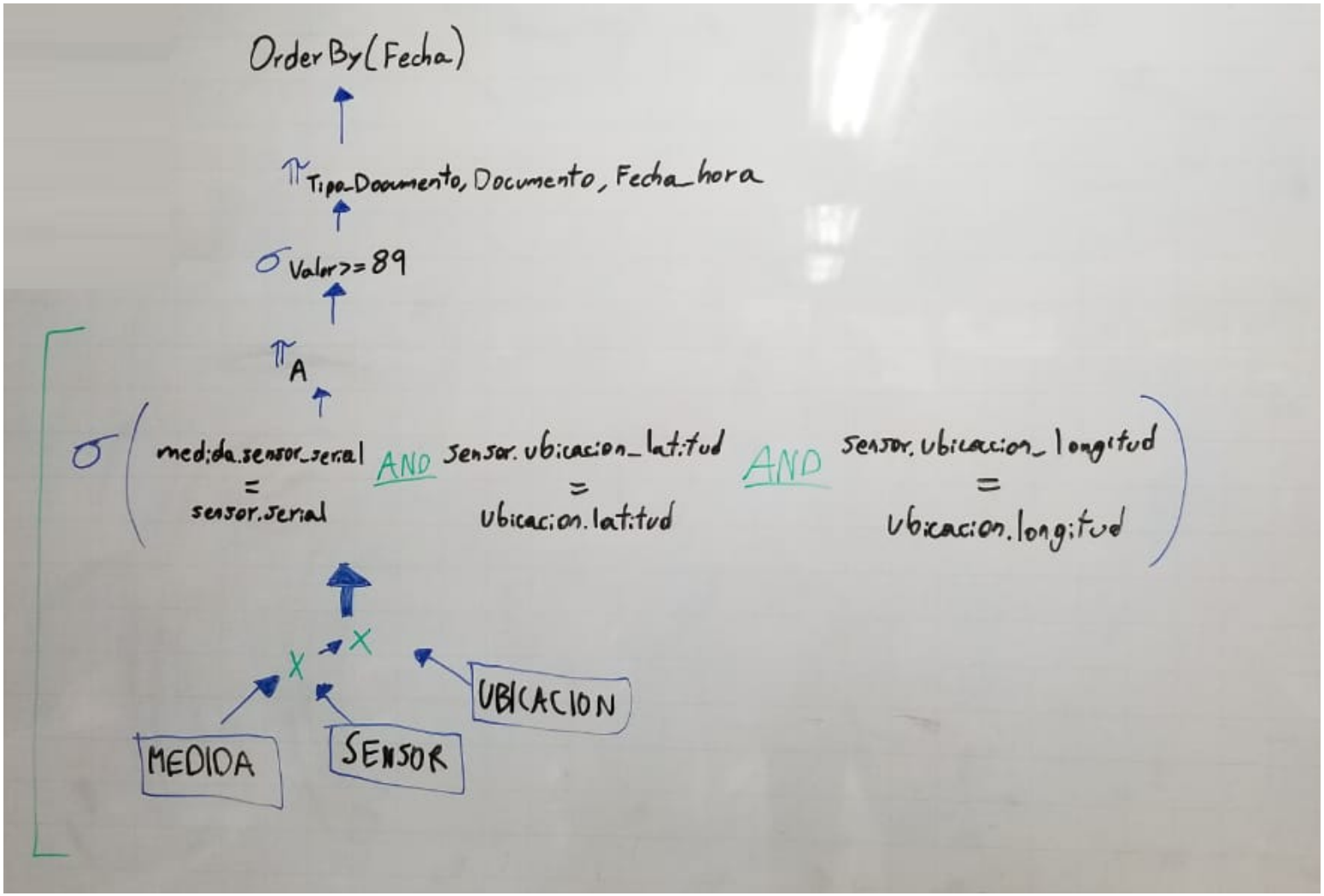
- Consulta en álgebra relacional.



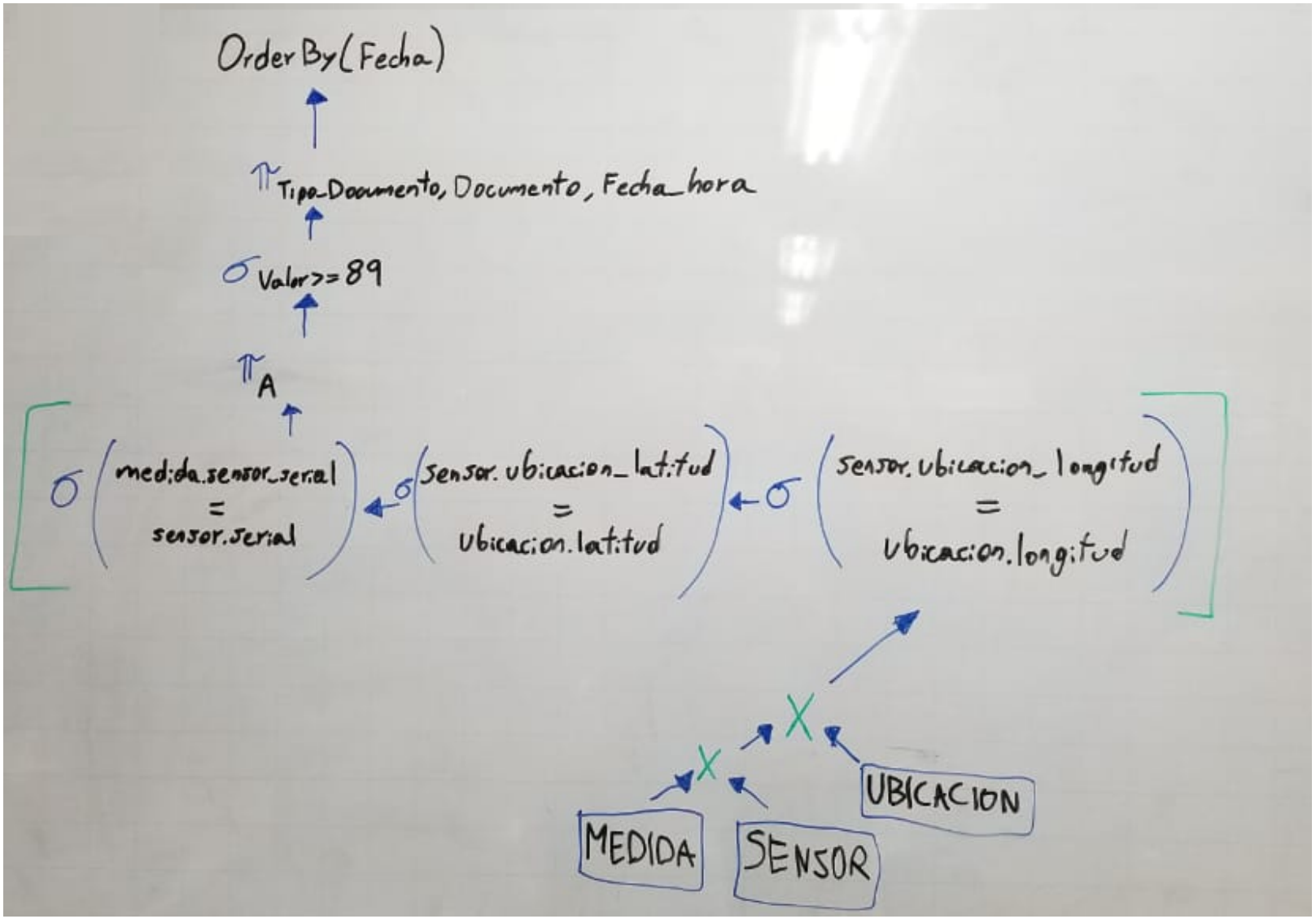
- Vista expandida en Álgebra Relacional



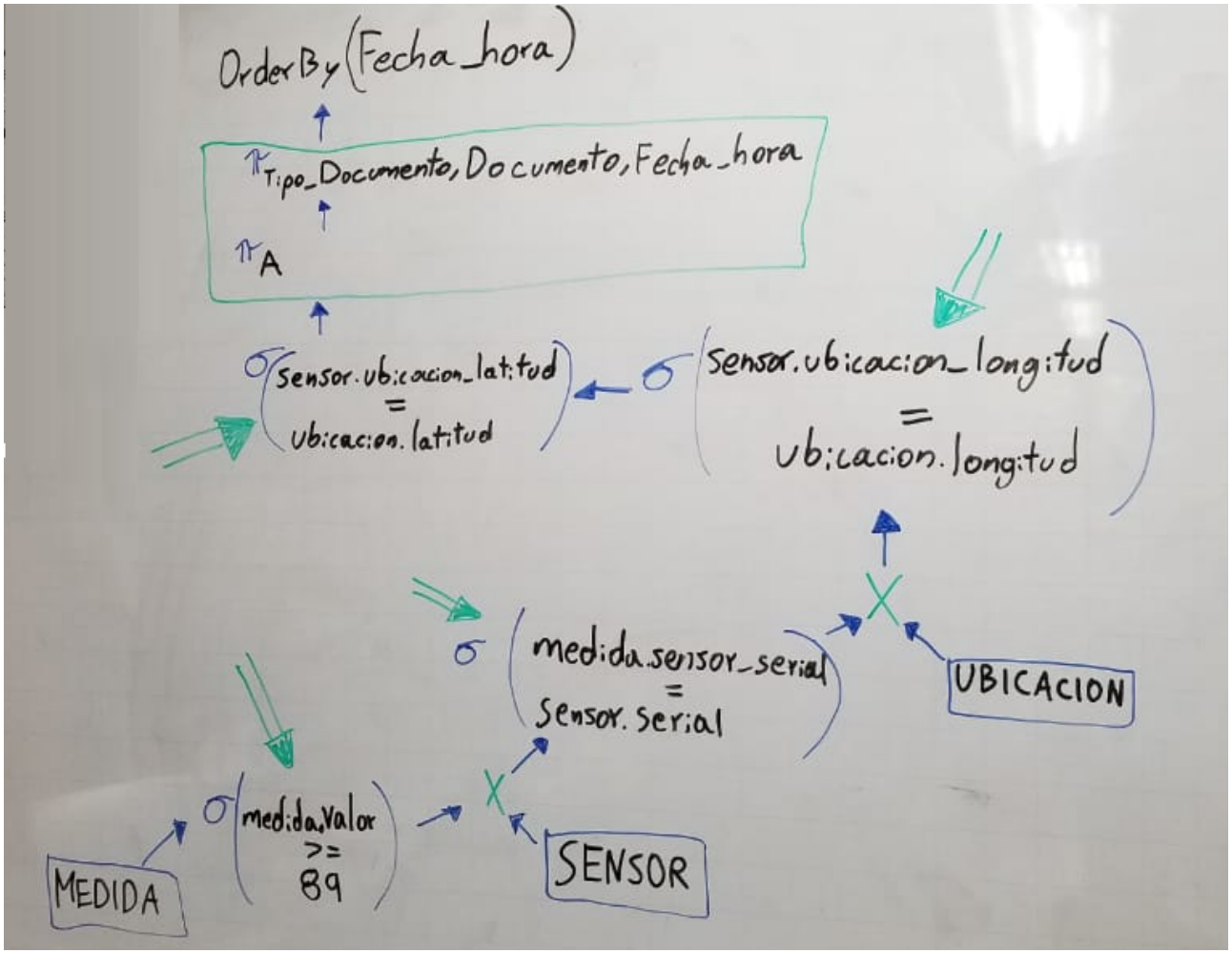
- Vista reemplazada + consulta de usuario, se utiliza la siguiente definición : *A = (medida.Fecha_Hora, medida.valor, medida.sensor_serial, sensor.tipo_medida_Codigo_tipo_medida, ubicacion.cientifico_tipo_documento, ubicacion.documento, ubicacion.barrio_codigo_barrio, ubicacion.descripcion, ubicacion.latitud, ubicacion.longitud)*



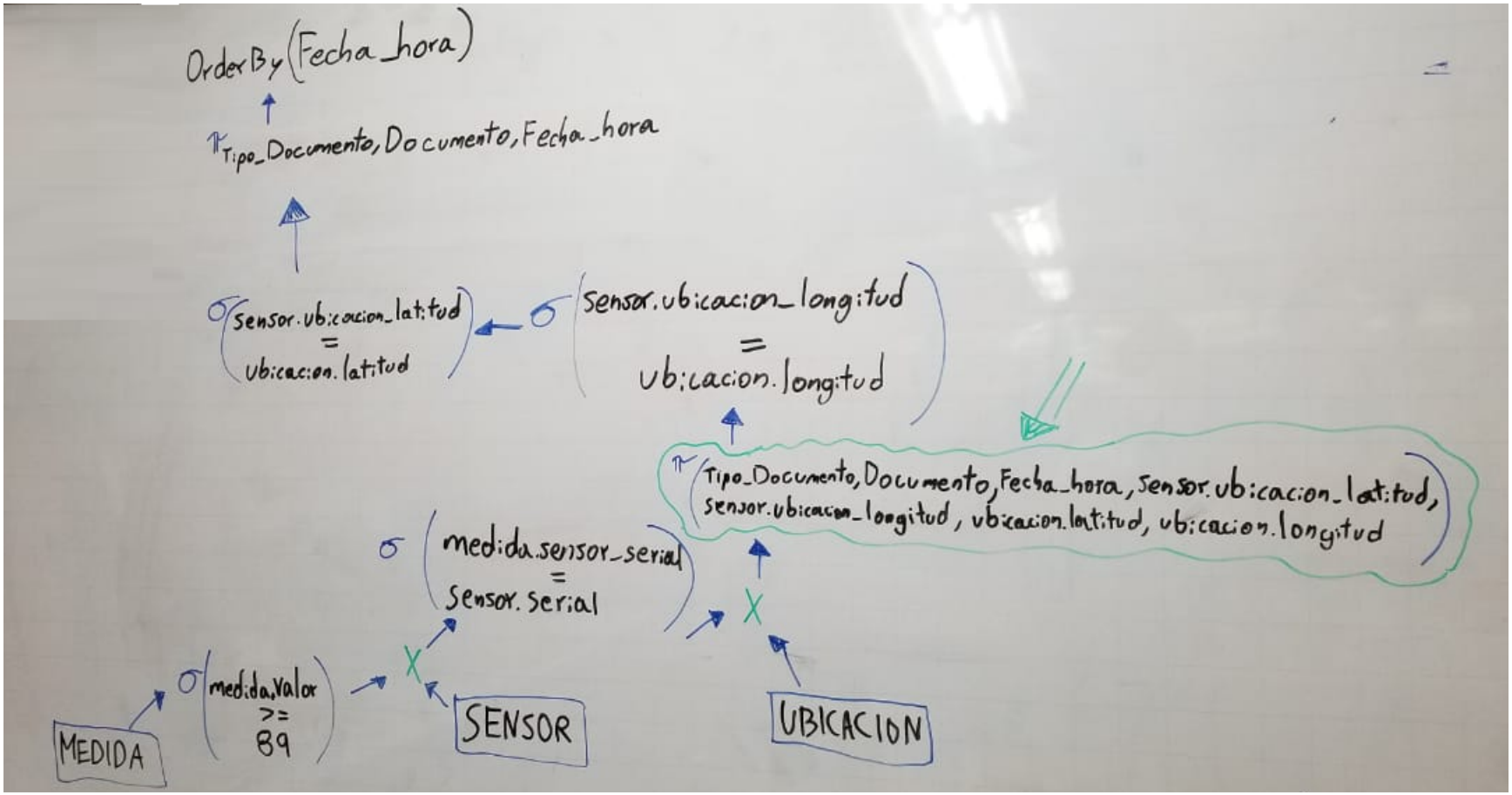
- Aplicación de la regla 4 *(cascada de selecciones)* para separar las selecciones de la forma F1 AND F2



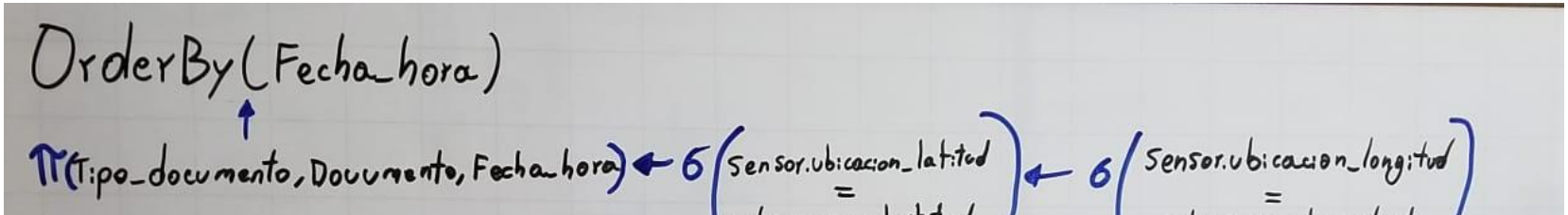
- Aplicación de la regla 3 *(Cascada de proyecciones)* para simplificar las proyecciones. A continuación aplicación las leyes 4 a 8 para mover las selecciones al nivel más bajo como sea posible en el árbol (indicado por las **flechas verdes)**



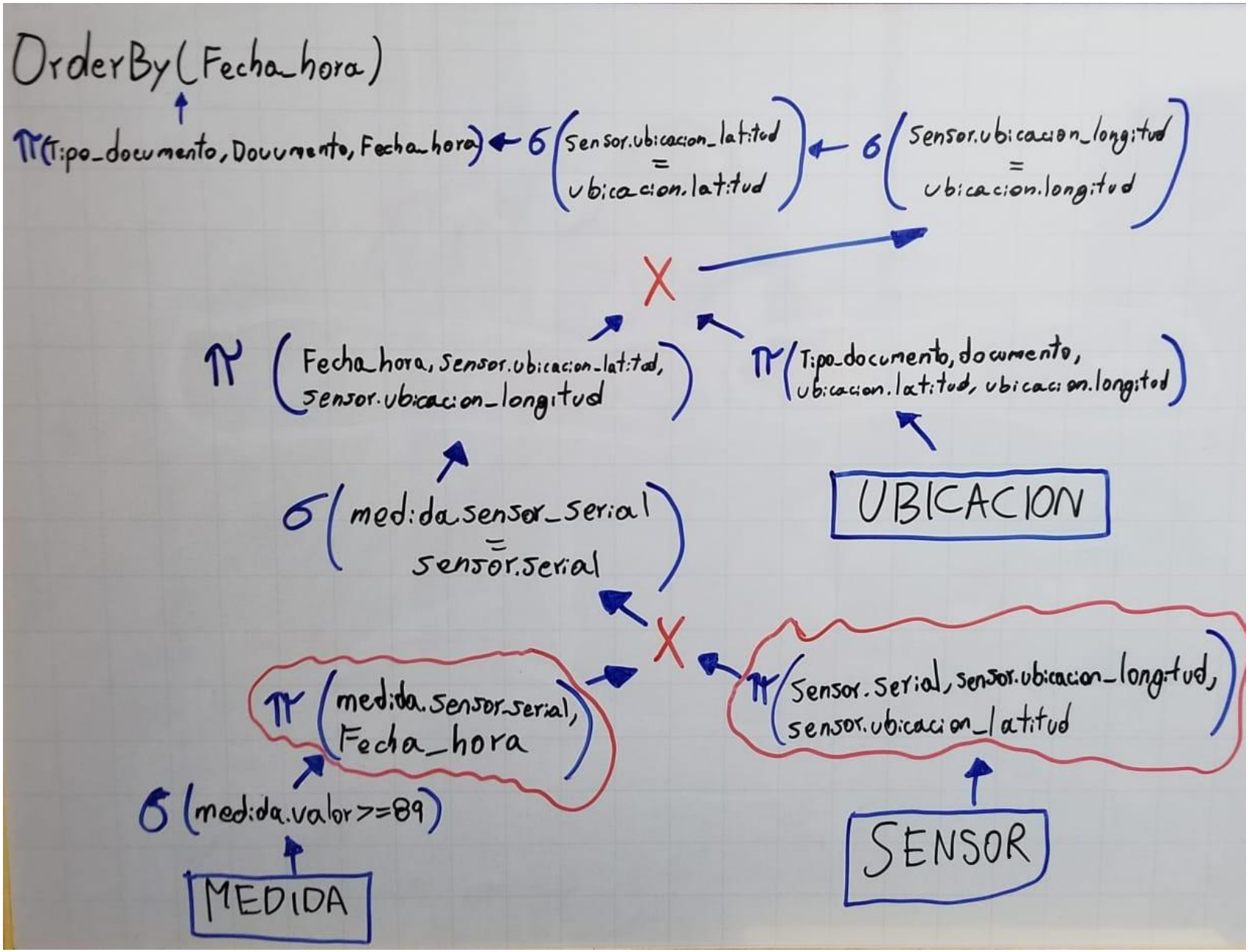
- Aplicación de la regla 5.2 *(Conmutando selecciones y proyecciones)* para agregar una proyección con los atributos necesarios para la optimización



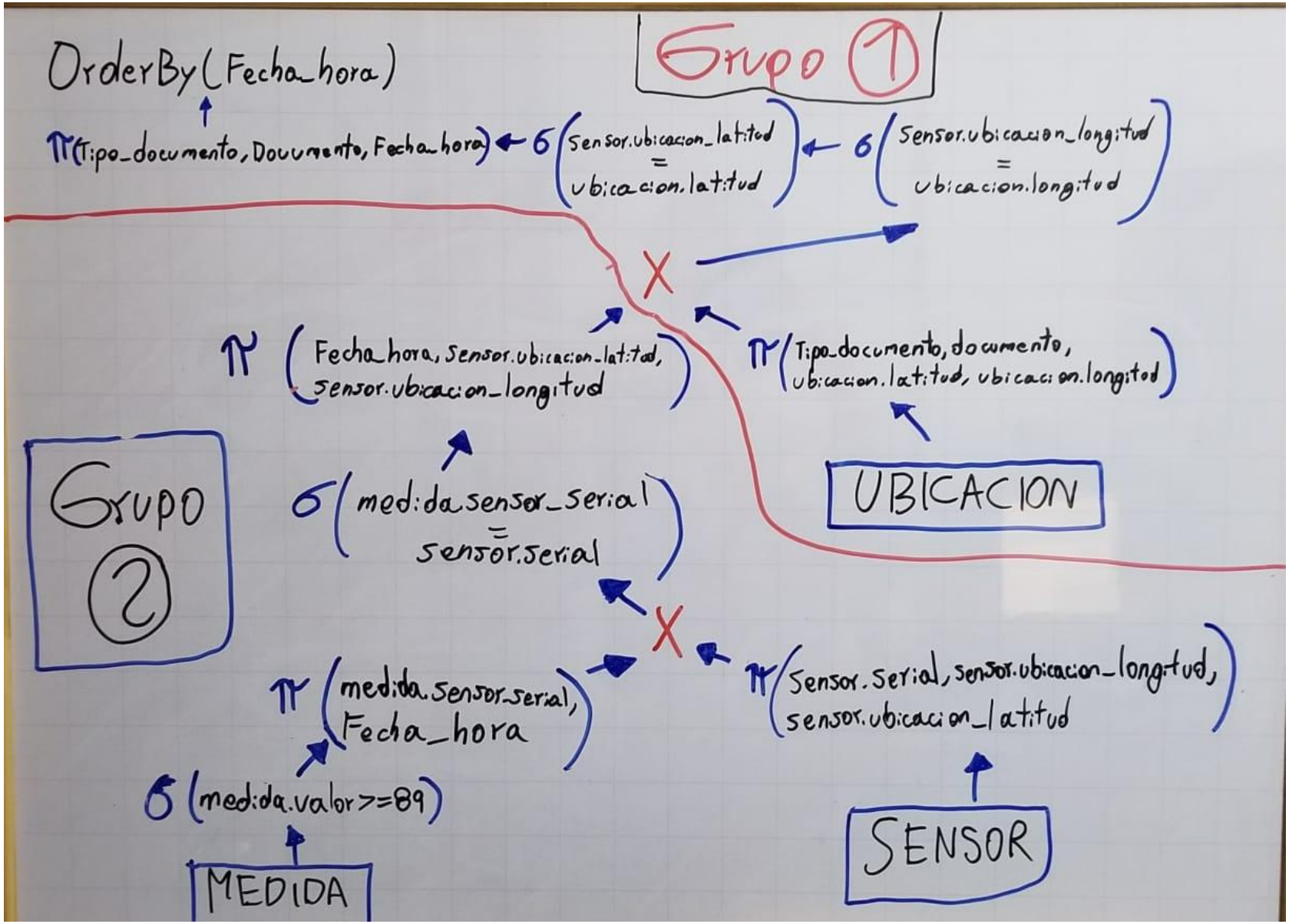
- Aplicación de la regla 10 para los productos *(conmutando proyección con un producto cartesiano)*



- Aplicación de la Regla 5.2 *(Conmutando selecciones y proyecciones)* y la Regla 10 *(conmutando proyección con un producto cartesiano)* para mover las proyecciones tan abajo como fuese posible.



- Partición de los Nodos del Árbol en grupos según las operaciones Binarias presentes en la consulta (en este caso sólo X). Podemos observar que todas las operaciones dentro de cada grupo son unitarias.



REFERENCIA

[1] Ullman, Jeffrey D. Database and Knowledge-Base System. Vol II. Cap. 11. Pp: 633-673. Computer Science Press. 1989.
Traducido por: John Freddy Duitama Muñoz. Profesor U. de A. Facultad de Ingeniería.
Optimización de Consultas. pp-24