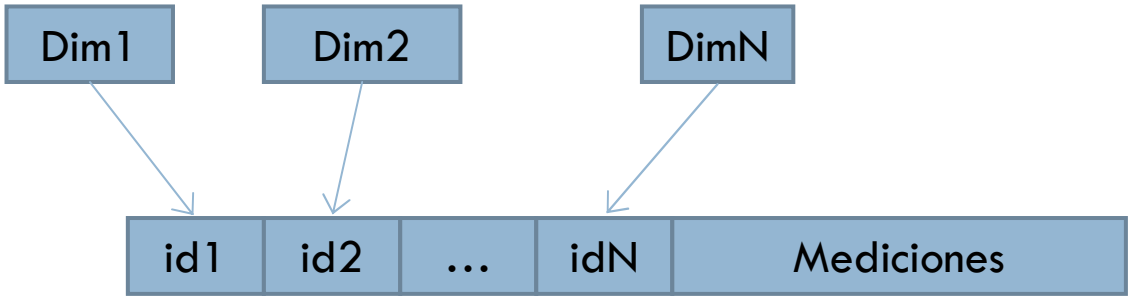
TEMA 3
PROCESAMIENTO DE CONSULTAS Y OPTIMIZACIÓN

Natalia Padilla Zea, Eladio Garví, José Samos

Contenido

3.1. Soporte de los sistemas relacionales

- PATRÓN DE CONSULTAS ESTÁNDAR
  - Partimos de un sistema ROLAP



  - A partir de él se obtienen informes mediante consultas
    - SELECT de las dimensiones
    - Operaciones sobre los hechos (p.e., SUM)
    - JOIN de los hechos y dimensiones
    - Puede haber filtros y agrupaciones

3.1. Soporte de los sistemas relacionales

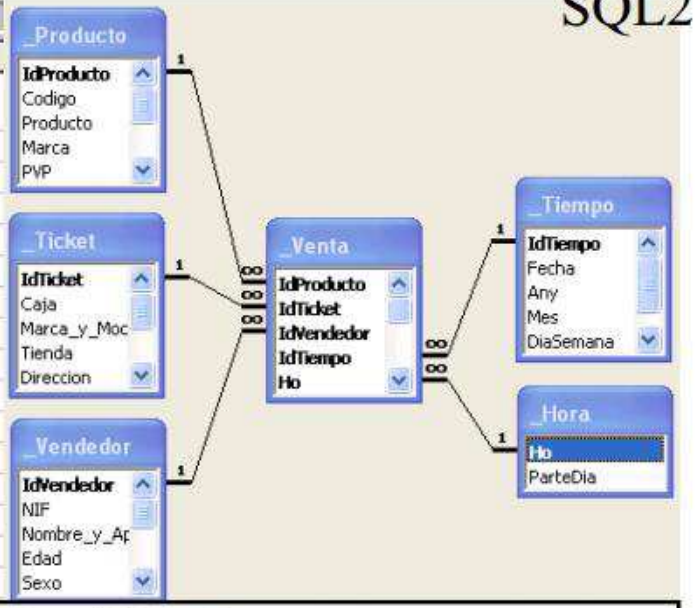
- PATRÓN DE CONSULTAS ESTÁNDAR
  - El patrón básico de consultas consistiría en:
    - Reducir las dimensiones (aplicar condiciones de selección)
    - Join entre Hechos y cada dimensión que interviene
    - Agrupar atributos de las dimensiones y agregar mediciones
    - Ordenar
  - Problema:
    - Si hay muchos Hechos, el JOIN de los Hechos con las dimensiones genera muchos registros también

3.1. Soporte de los sistemas relacionales

for Enero de 2002

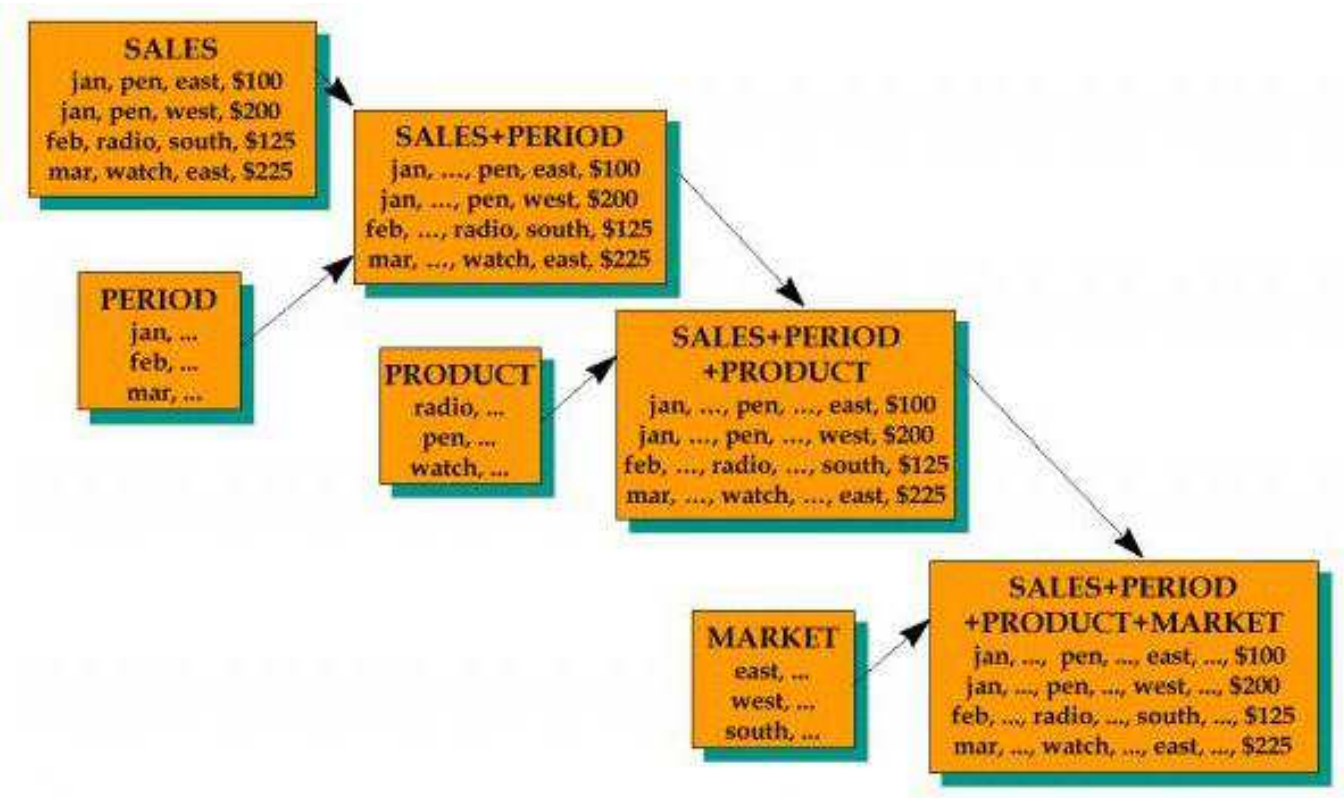
	A	B	C	D	E
1	Diasemana	Marca	Sum of Importe	Sum of Cantidad	
2	Domingo	Marca_1	€4.923,78	94,00	
3	Domingo	Marca_10	€4.460,63	88,00	
4	Domingo	Marca_11	€1.593,80	40,00	
5	Domingo	Marca_12	€3.259,46	44,00	
6	Domingo	Marca_13	€662,96	16,00	
7	Domingo	Marca_14	€4.432,68	70,00	
8	Domingo	Marca_15	€4.662,13	112,00	
9	Domingo	Marca_16	€4.390,17	114,00	
10	Domingo	Marca_17	€471,09	33,00	
11	Domingo	Marca_18	€2.152,34	23,00	
12	Domingo	Marca_19	€4.467,10	66,00	
13	Domingo	Marca_2	€6.188,64	165,00	
14	Domingo	Marca_20	€4.930,33	79,00	
15	Domingo	Marca_3	€3.919,94	73,00	
16	Domingo	Marca_4	€5.967,57	80,00	
17	Domingo				
18	Domingo				
19	Domingo				
20	Domingo				
21	Domingo				
22	Jueves				
23	Jueves				
24	Jueves				
25	Jueves				
26	Jueves				
27	Jueves				
28	Jueves				
29	Jueves				
30	Jueves	Marca_17	€6.976,84	107,00	

SQL2




```
select [Tiempo].[Diasemana], [Producto].[Marca],
        Sum( [Venta].[Importe]) as Col1, Sum( [Venta].[Cantidad]) as Col2
from [Venta], [Tiempo], [Producto]
where [Venta].[IDTIEMPO] = [Tiempo].[IDTIEMPO] and
        [Venta].[IDPRODUCTO] = [Producto].[IDPRODUCTO] and
        [Tiempo].[Mes] in ('Enero') and [Tiempo].[Any] in (2002)
group by [Tiempo].[Diasemana], [Producto].[Marca]
order by [Tiempo].[Diasemana], [Producto].[Marca]
```

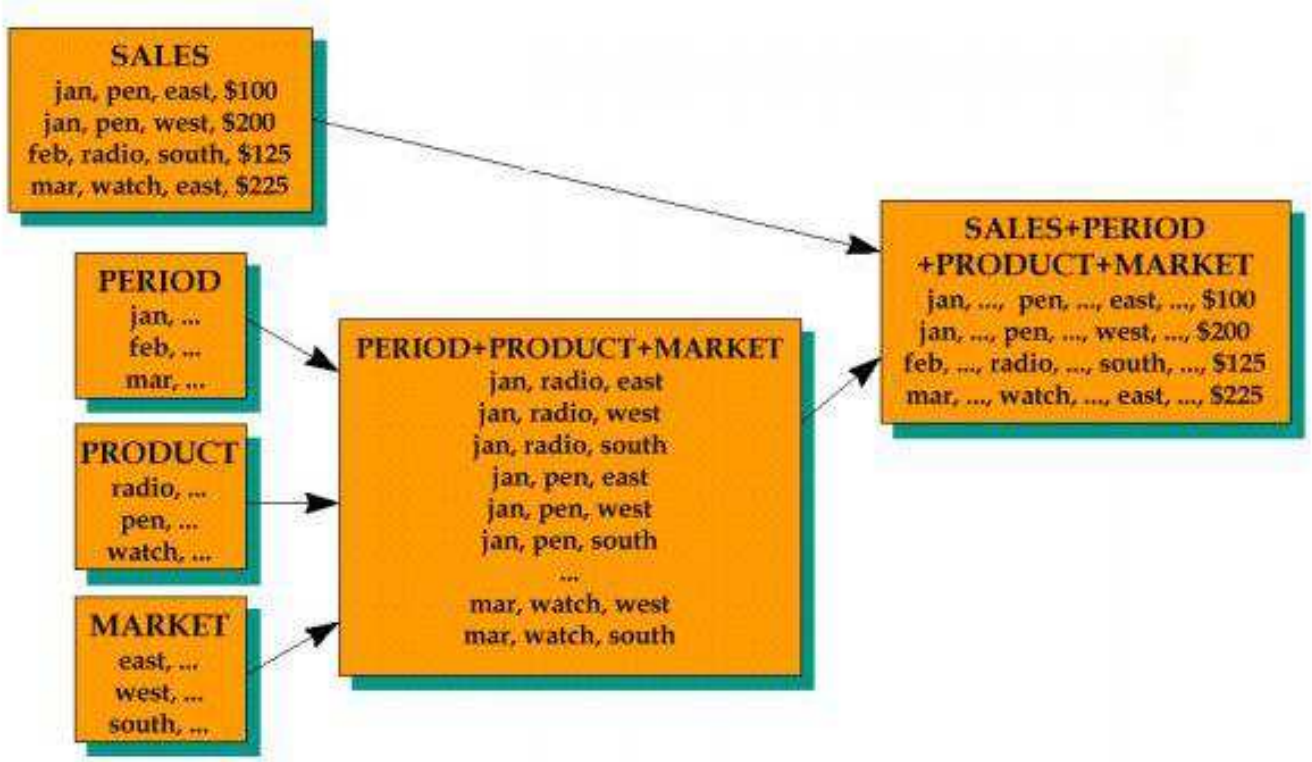
3.1. Soporte de los sistemas relacionales



3.1. Soporte de los sistemas relacionales

- PATRÓN DE CONSULTAS ESTÁNDAR
  - Alternativas:
    - Que los Hechos intervengan lo menos posible
    - Combinar las dimensiones reducidas y acceder una sola vez a los Hechos
      - Combinar las dimensiones puede o no ser factible, porque sean muy grandes

3.1. Soporte de los sistemas relacionales



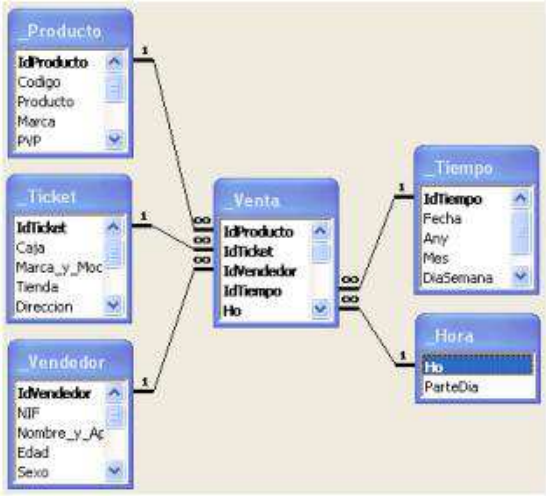
3.1. Soporte de los sistemas relacionales

- Soporte de SQL
  - Estándar de lenguaje de consulta
  - Varias versiones
  - SQL2:
    - De 1992: incluye la orden *select*
    - Para obtener informes con subtotales por N campos se necesitan N+1 subconsultas
    - El informe con todos los subtotales posibles requiere gran cantidad de select (2**N)
    - Las tablas intermedias ocupan mucho porque se hace el producto de cada dimensión con los hechos

3.1. Soporte de los sistemas relacionales

- Soporte de SQL
  - SQL3:
    - De 1999: incluye modificadores multidimensionales
    - Modificador ROLLUP: genera subtotales con una única consulta
      - En el informe incluye NULL para indicar la suma
    - Modificador CUBE: genera el informe con todos los subtotales posibles
    - Optimiza las consultas a nivel interno

3.1. Soporte de los sistemas relacionales




```
SELECT Time, Region, Department, SUM(Profit)
 FROM Sales
 GROUP BY Time, Region, Department
UNION  ALL
 SELECT Time, Region, '' , SUM(Profit)
 FROM Sales
 GROUP BY Time, Region
UNION ALL
 SELECT Time, '', '', SUM(Profits)
 FROM Sales
 GROUP BY Time
UNION ALL
 SELECT '', '', '', SUM(Profits)
 FROM Sales;
```

SQL3

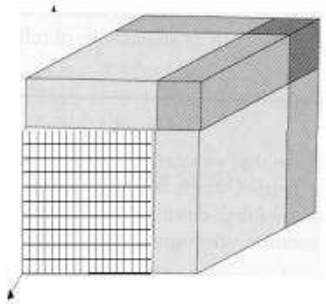
Time	Region	Department	Profit
1996	Central	VideoRental	75,000
1996	Central	VideoSales	74,000
1996	Central	[NULL]	149,000
1996	East	VideoRental	89,000
1996	East	VideoSales	115,000
1996	East	[NULL]	204,000
1996	West	VideoRental	87,000
1996	West	VideoSales	86,000
1996	West	[NULL]	173,000
1996	[NULL]	[NULL]	526,000
1997	Central	VideoRental	82,000
1997	Central	VideoSales	85,000
1997	Central	[NULL]	167,000
1997	East	VideoRental	101,000
1997	East	VideoSales	137,000
1997	East	[NULL]	238,000
1997	West	VideoRental	96,000
1997	West	VideoSales	97,000
1997	West	[NULL]	193,000
1997	[NULL]	[NULL]	598,000
[NULL]	[NULL]	[NULL]	1,124,000

```
SELECT Time, Region, Department,
 sum(Profit) AS Profit FROM sales
 GROUP BY ROLLUP(Time, Region, Dept)
```

3.1. Soporte de los sistemas relacionales

SQL3

Time	Region	Department	Profit
1996	Central	VideoRental	75,000
1996	Central	VideoSales	74,000
1996	Central	[NULL]	149,000
1996	East	VideoRental	89,000
1996	East	VideoSales	115,000
1996	East	[NULL]	204,000
1996	West	VideoRental	87,000
1996	West	VideoSales	86,000
1996	West	[NULL]	173,000
1996	[NULL]	VideoRental	251,000
1996	[NULL]	VideoSales	275,000
1996	[NULL]	[NULL]	526,000
1997	Central	VideoRental	82,000
1997	Central	VideoSales	85,000
1997	Central	[NULL]	167,000
1997	East	VideoRental	101,000
1997	East	VideoSales	137,000
1997	East	[NULL]	238,000
1997	West	VideoRental	96,000
1997	West	VideoSales	97,000
1997	West	[NULL]	193,000
1997	[NULL]	VideoRental	279,000
1997	[NULL]	VideoSales	319,000
1997	[NULL]	[NULL]	598,000
[NULL]	Central	VideoRental	157,000
[NULL]	Central	VideoSales	159,000
[NULL]	Central	[NULL]	316,000
[NULL]	East	VideoRental	190,000
[NULL]	East	VideoSales	252,000
[NULL]	East	[NULL]	442,000
[NULL]	West	VideoRental	183,000
[NULL]	West	VideoSales	183,000
[NULL]	West	[NULL]	366,000
[NULL]	[NULL]	VideoRental	530,000
[NULL]	[NULL]	VideoSales	594,000
[NULL]	[NULL]	[NULL]	1,124,000



2^n SELECTs

```
SELECT Time, Region, Department,
   sum(Profit) AS Profit FROM sales
   GROUP BY CUBE  (Time, Region, Dept)
```

3.1. Soporte de los sistemas relacionales

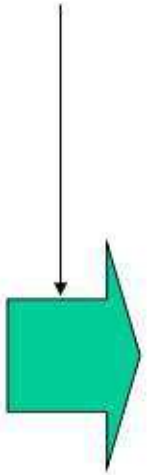
- Soporte de SQL
  - SQL3:
    - Modificador DECODE (GROUPING): devuelve si se trata de una agrupación (diciendo el tipo) o el valor NULL
    - Facilita la presentación de los informes
  - Los importancia ganada por los SMD ha provocado cambios en el estándar SQL
  - Es preferible usar herramientas SQL3 para optimizar

3.1. Soporte de los sistemas relacionales

SQL3

```
SELECT
  DECODE(GROUPING(Time), 1, 'All Times', Time) as Time,
  DECODE(GROUPING(region), 1, 'All Regions', 0, null)  as
  Region, SUM(Profit) AS Profit from Sales
  GROUP BY CUBE(Time, Region)
```

Time	Region	Profit
1996	East	200,000
1996	[NULL]	200,000
[NULL]	East	200,000
[NULL]	[NULL]	190,000
[NULL]	[NULL]	190,000
[NULL]	[NULL]	190,000
[NULL]	[NULL]	390,000



Time	Region	Profit
1996	East	200,000
1996	All Regions	200,000
All Times	East	200,000
[NULL]	[NULL]	190,000
[NULL]	All Regions	190,000
All Times	[NULL]	190,000
All Times	All Regions	390,000

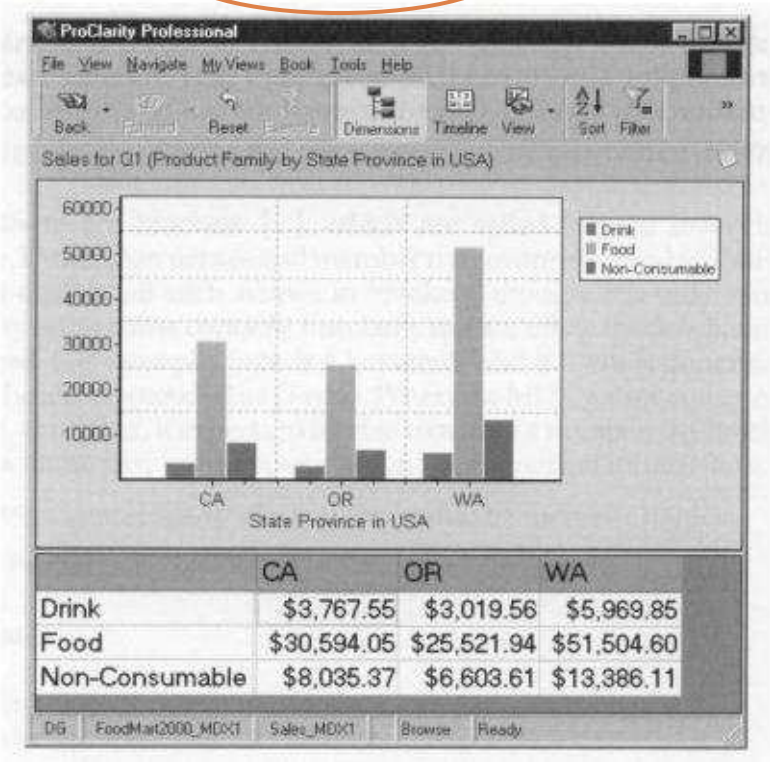
3.2. Estándares de consulta e intercambio

- SQL2 y SQL3 se basan en el modelo de datos relacional
- En SMD el estándar de facto es MDX:
  - Definido por Microsoft para Analysis Services
  - Se usa en la mayoría de los SMD, como Oracle y Mondrian (software libre)
  - Similar a SQL

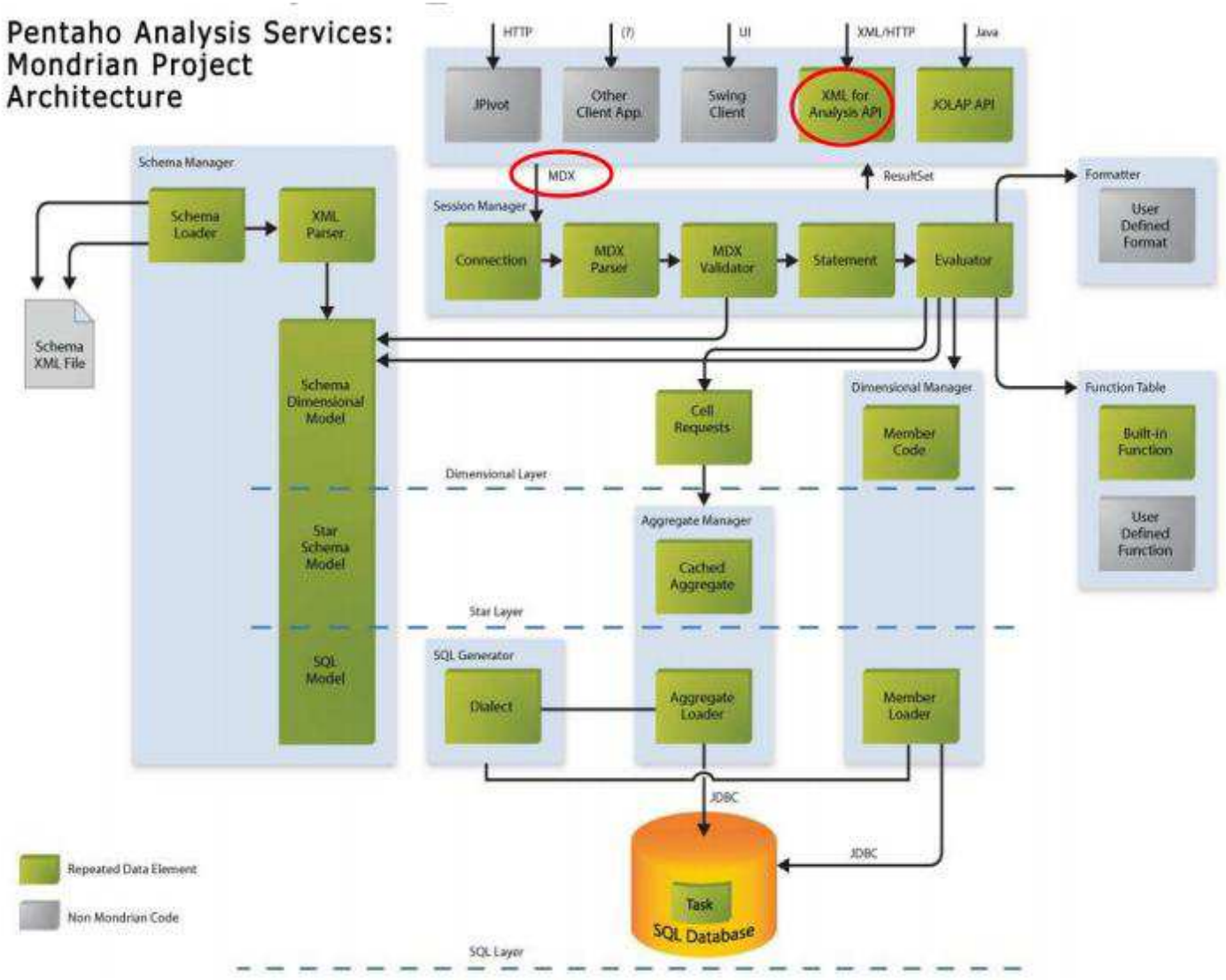
3.2. Estándares de consulta e intercambio


```
SELECT
{[Customers].[All Customers].[USA].Children} ON COLUMNS ,
{[Product].[All Products].Children} ON ROWS
FROM [Sales_MDX1]
WHERE ([Measures].[Sales], [Time].[1998].[Q1])
```

¿Qué operación representa?

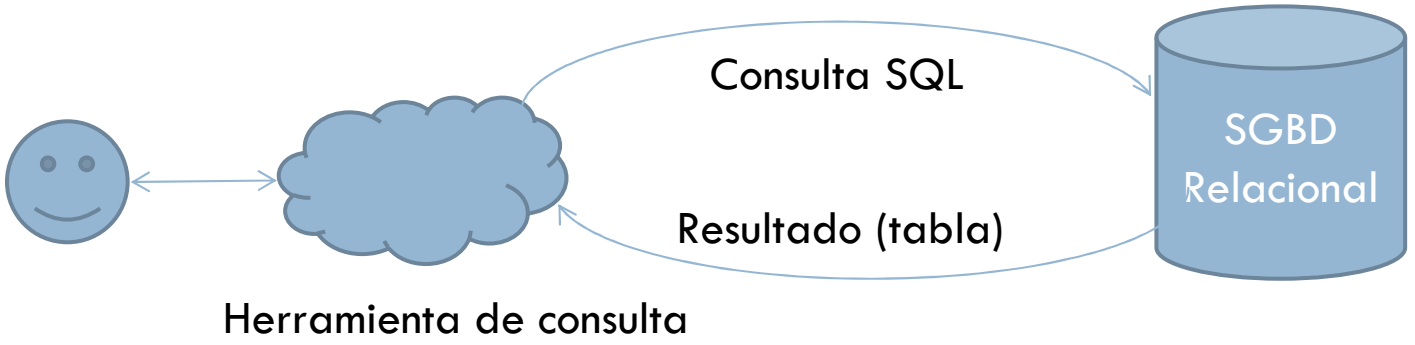


3.2. Estándares de consulta e intercambio

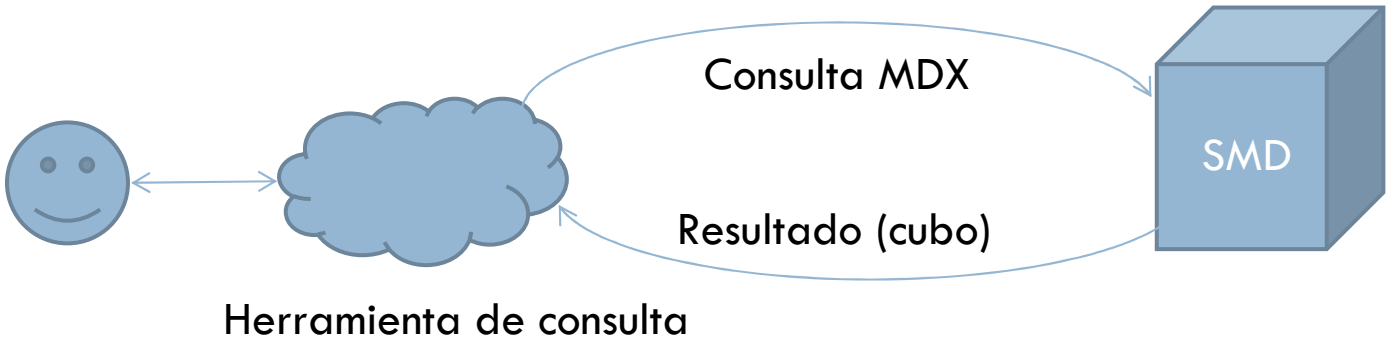


3.2. Estándares de consulta e intercambio

- Con SQL teníamos:



- Con MDX:

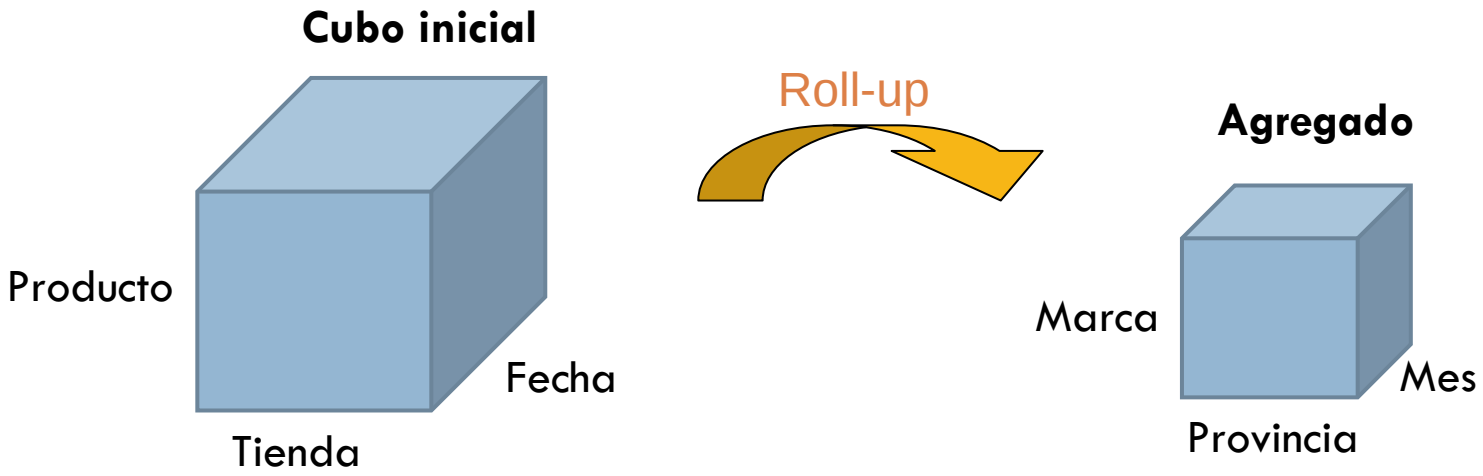


3.2. Estándares de consulta e intercambio

- El cubo que devuelve la consulta MDX se expresa en XML for Analysis (XMLA)
- XML for Analysis es un estándar para representar cubos
- XMLA permite “hablar” a las aplicaciones cliente con sistemas multidimensionales
- Utiliza Simple Object Acces Protocol (SOAP) para conectar a los clientes con los servidores OLAP
- Su lenguaje de consulta es MDX

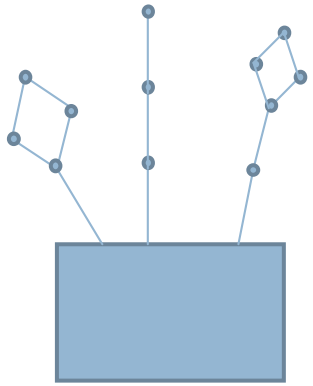
3.3. Optimización y ajuste a nivel lógico

- Agregados
  - Cubos obtenidos a partir del cubo base mediante Roll-up para responder a consultas más rápidamente
- Ejemplo:
  - Consultas frecuentes por mes, provincia y marca



3.3. Optimización y ajuste a nivel lógico

- Ventaja: mejora el tiempo de respuesta
- Inconvenientes:
  - Si cambia algo en el cubo base hay que trasladar los cambios a los agregados
  - Ocupa más espacio
  - Determinar el número de agregados a definir:

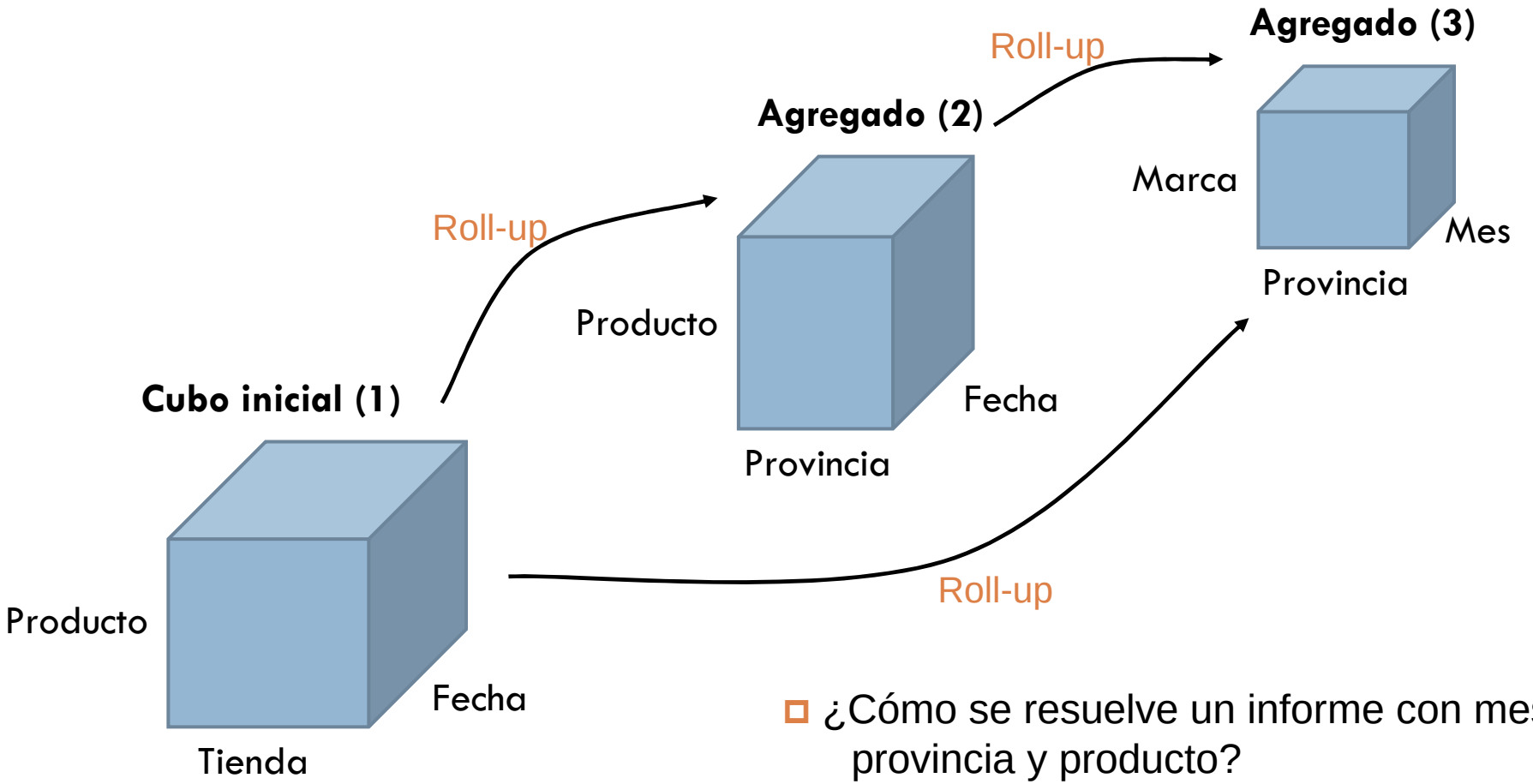


Nº máximo de agregados = (4x3x5)-1 (cubo base)

¿Se definen todos? Se usa algún criterio
(P.e., máxima ocupación total en Megabytes o mejorar el tiempo de respuesta en un porcentaje determinado)

3.3. Optimización y ajuste a nivel lógico

- Relaciones entre agregados (Ejemplo)



- ¿Cómo se resuelve un informe con mes, provincia y producto?

3.3. Optimización y ajuste a nivel lógico

- Uso de los agregados
  - El sistema debe ser amigable: transparente al usuario
  - Habrá un "navegador de agregados" que modificará la consulta para realizarla sobre el cubo más adecuado
  - Se anota las veces que se accede a los agregados
  - Los agregados se construyen de mayor a menor tamaño
  - Los agregados se recorren desde el más pequeño para ver si responden a la consulta

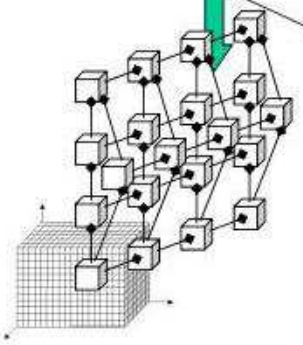
3.3. Optimización y ajuste a nivel lógico

- Ejemplo de uso de los agregados

```
select category_description, sum(sales_dollars)
from base_sales_fact, product, store, time
where base_sales_fact.product_key = product.product_key
and base_sales_fact.store_key = store.store_key
and base_sales_fact.time_key = time.time_key
and store.city = 'Cincinnati'
and time.day = 'January 1, 1996'
group by category_description.
```

Metadatos de agregados y estadísticas de uso

Navegador de Agregados



```
select category_description, sum(sales—dollars)
from category_sales_fact, category_product, store, time
where category_sales_fact.product_key = category_product.product_
   key
and category_sales_fact.store_key = store.store_key
and category_sales_fact.time_key = time.time_key
and store.city = 'Cincinnati'
and time.day = 'January 1, 1996'
group by category_description.
```

3.3. Optimización y ajuste a nivel lógico

- Fallo en la dispersión de agregados ¿nº de instancias?

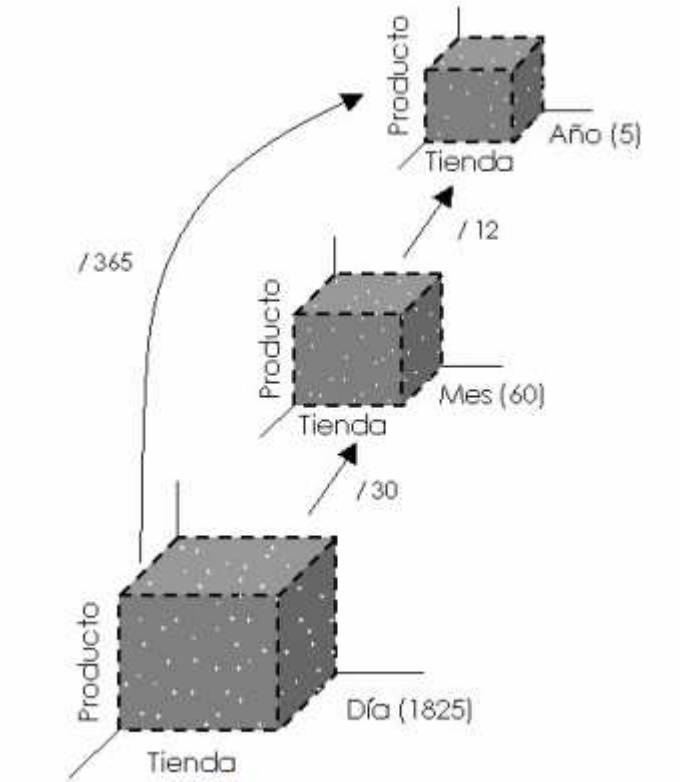


Figura 2.35: Fallo en la dispersión de los agregados

Dado el cubo base de la figura 2.35, supongamos que éste tiene $150 \cdot 10^6$ de registros. Entonces podríamos pensar que el agregado a nivel de mes tendrá $\frac{150 \cdot 10^6}{30}$ registros y, del mismo modo, el agregado a nivel de año, $\frac{150 \cdot 10^6}{365}$ registros.

Sin embargo esta suposición es errónea, ya que en los agregados existe menos dispersión (los huecos son más pequeños) que en el cubo base.

De este modo, si el número de registros del cubo base se calcula como

$$20 \text{ tiendas} \times (60{,}000 \text{ productos} \times 7\,\%) \times 1825 \text{ días} \approx 150 \cdot 10^6$$

entonces el número de registros del agregado a nivel de mes sería

$$20 \text{ tiendas} \times (60{,}000 \text{ productos} \times 80\,\%) \times 60 \text{ meses} = 57{,}6 \cdot 10^6 \quad \left(\neq \frac{150 \cdot 10^6}{30} = 5 \cdot 10^6\right)$$

y el número de registros del agregado a nivel de año, como

$$20 \text{ tiendas} \times (60{,}000 \text{ productos} \times 100\,\%) \times 5 \text{ años} = 6 \cdot 10^6 \quad \left(\neq \frac{150 \cdot 10^6}{365} \approx 411 \cdot 10^3\right)$$

El número de registros en los agregados no se reduce en la misma proporción que lo hacen los ejes de coordenadas.

Referencia: Félix Gómez Mármol. Sistemas de Información. Universidad de Granada

3.4. Optimización y ajuste a nivel físico

- Índice en mapa de bits

  - Características:
    - Desarrollado en los años 70
    - Buenos para consultas
    - Menos eficientes para modificaciones
    - Vuelven en los 90 para sistemas OLAP

3.4. Optimización y ajuste a nivel físico

- Índice en mapa de bits
  - Proceso de construcción
    - Una columna para cada valor distinto del campo a indexar
    - Una fila por cada registro de la tabla a indexar
    - En cada celda se pone 1 si el registro tiene el valor de esa columna y 0 si no lo tiene
  - Se pueden hacer tantos índices como queramos
  - Permite responder algunas consultas mediante operaciones lógicas

3.4. Optimización y ajuste a nivel físico

- Ejemplo de índice en mapa de bits

Tabla: Empleado

Id_registro	Nombre	Edad
rE4	Juan	20
rE18	María	20
rE19	José	21
rE34	Carmen	20
rE35	José	20
rE36	Luis	25
rE5	María	21
rE41	Andrés	26

CREATE BITMAP INDEX Nombres
ON Empleado(Nombre)

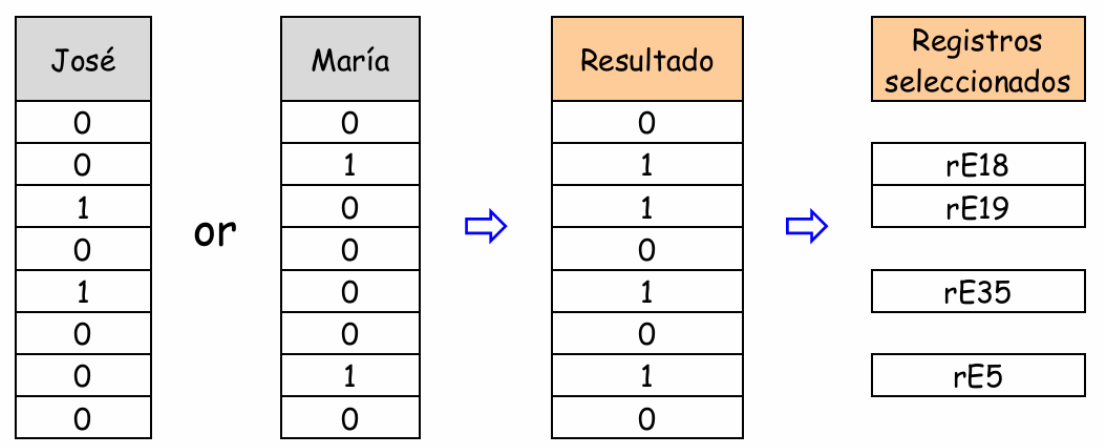
Nombres					
Juan	María	José	Carmen	Luis	Andrés
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	0	0	1	0
0	1	0	0	0	0
0	0	0	0	0	1

CREATE BITMAP INDEX Edades
ON Empleado(Edad)

Edades			
20	21	25	26
1	0	0	0
1	0	0	0
0	1	0	0
1	0	0	0
1	0	0	0
0	0	1	0
0	1	0	0
0	0	0	1

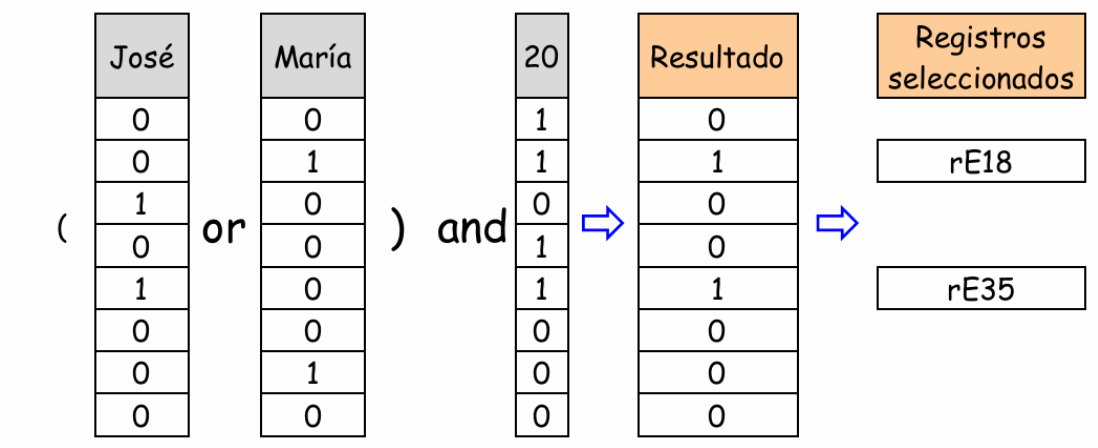
3.4. Optimización y ajuste a nivel físico

- Ejemplo de índice en mapa de bits
  - ¿Cuántas personas se llaman José o María?



3.4. Optimización y ajuste a nivel físico

- Ejemplo de índice en mapa de bits
  - ¿Cuántas personas se llaman José o María y tienen 20 años?



3.4. Optimización y ajuste a nivel físico

- Índice de Join (Join Index)
  - Definición 1: materialización de un Join (entre tablas) mediante un índice
  - Definición 2: índice definido sobre una tabla usando campos de otra tabla a la que se accede mediante Join
  - Pueden ser de varios tipos: mapa de bits, árboles…
  - Uso en sistemas OLAP: se materializa el Join entre las dimensiones y los hechos utilizando las llaves generadas correspondientes a las dimensiones afectadas (en particular puede ser una única dimensión)

3.4. Optimización y ajuste a nivel físico

- Índice de Join
  - Proceso general
    - Reducir cada una de las dimensiones
      - Se aplican las operaciones OR y AND que se necesiten sobre mapas de bits de la dimensión para seleccionar los registros que intervienen de esa dimensión (p.e., María o José, y 20 años)
      - Las llaves generadas de los registros obtenidos en el paso anterior se tratan como columnas del índice de Join entre la dimensión y los hechos
      - Las columnas anteriores se unen usando OR para dar lugar a una única columna de índice de Join para la dimensión
    - Combinar las columnas de índice de Join de las dimensiones haciendo AND entre ellas
    - Acceder a los hechos

3.4. Optimización y ajuste a nivel físico

Ejemplo de índice de Join

- Seleccionar las ventas realizadas en tiendas de Granada y Almería por empleados que se llamen José o María y que tengan 20 años

Tabla: Empleado (con llave generada)

rId_Empleado	Id_Empleado	Nombre	Edad
rE4	1	Juan	20
rE18	2	María	20
rE19	3	José	21
rE34	4	Carmen	20
rE35	5	José	20
rE36	6	Luis	25
rE5	7	María	21
rE41	8	Andrés	26

Tabla: Tienda (con llave generada)

rId_Tienda	Id_Tienda	Nombre	Provincia
rT2	1	Pancho	Granada
rT13	2	Cuca	Jaén
rT17	3	Piluca	Granada
rT4	4	Aladino	Almería
rT27	5	Soleá	Málaga

Tabla: Venta (utilizando las llaves generadas)

rId_Venta	Id_Empleado	Id_Tienda	Id_Producto	Id_Tiempo	Cantidad
rV2	1	1	23	1	1
rV32	2	1	4	1	2
rV21	3	2	67	1	4
rV23	1	1	2	1	3
rV56	1	1	43	1	2
rV6	2	1	78	1	2
rV12	3	2	29	1	1
rV7	2	1	81	2	1
rV65	2	1	84	2	1
rV18	4	2	64	2	3
rV37	4	2	53	2	1
rV60	5	3	23	2	2
rV45	1	1	37	2	5
rV31	6	4	67	2	3
rV48	7	5	78	2	1
rV9	2	1	76	2	2
rV55	8	3	53	2	1

3.4. Optimización y ajuste a nivel físico

- Ejemplo de índice de Join
  - Los empleados se seleccionan de forma análoga al ejemplo de uso de mapa de bits
  - A continuación se seleccionan las tiendas

```
CREATE BITMAP INDEX Provincias
ON Tienda(Provincia)
```

Granada	Jaén	Almería	Málaga
1	0	0	0
0	1	0	0
1	0	0	0
0	0	1	0
0	0	0	1

Granada
1
0
1
0
0

or

Almería
0
0
0
1
0

⇨

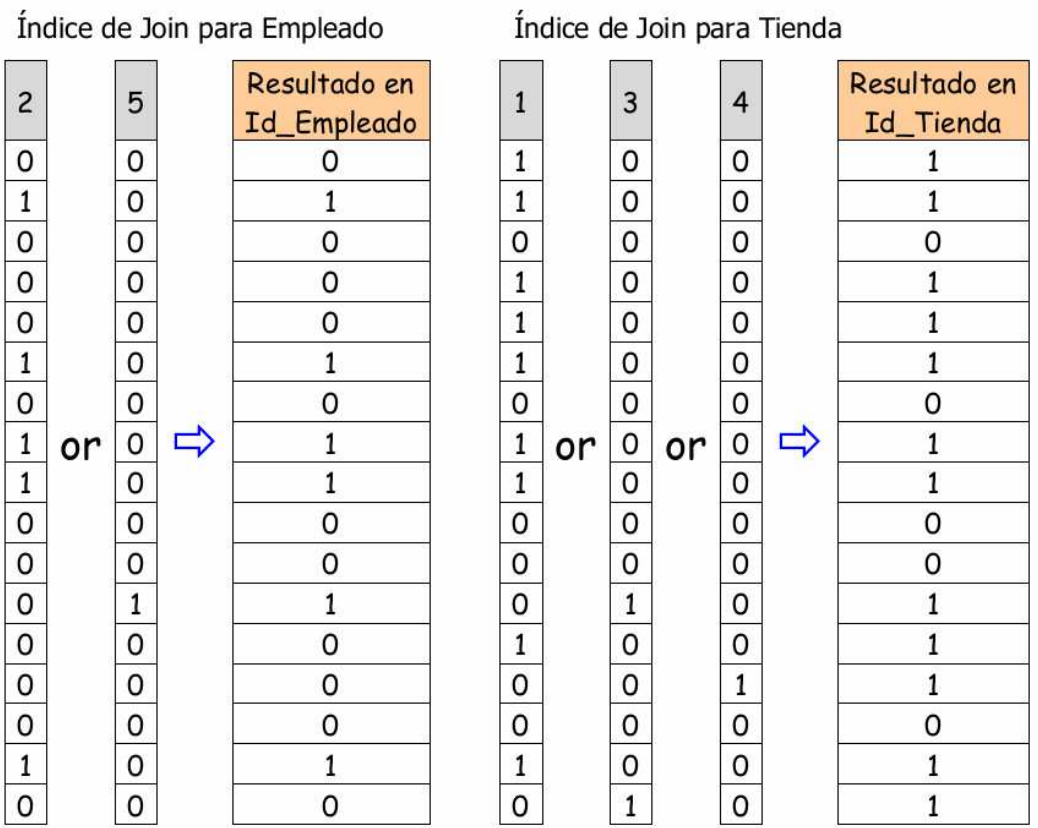
Resultado
1
0
1
1
0

⇨

Registros seleccionados
rT2
rT17
rT4

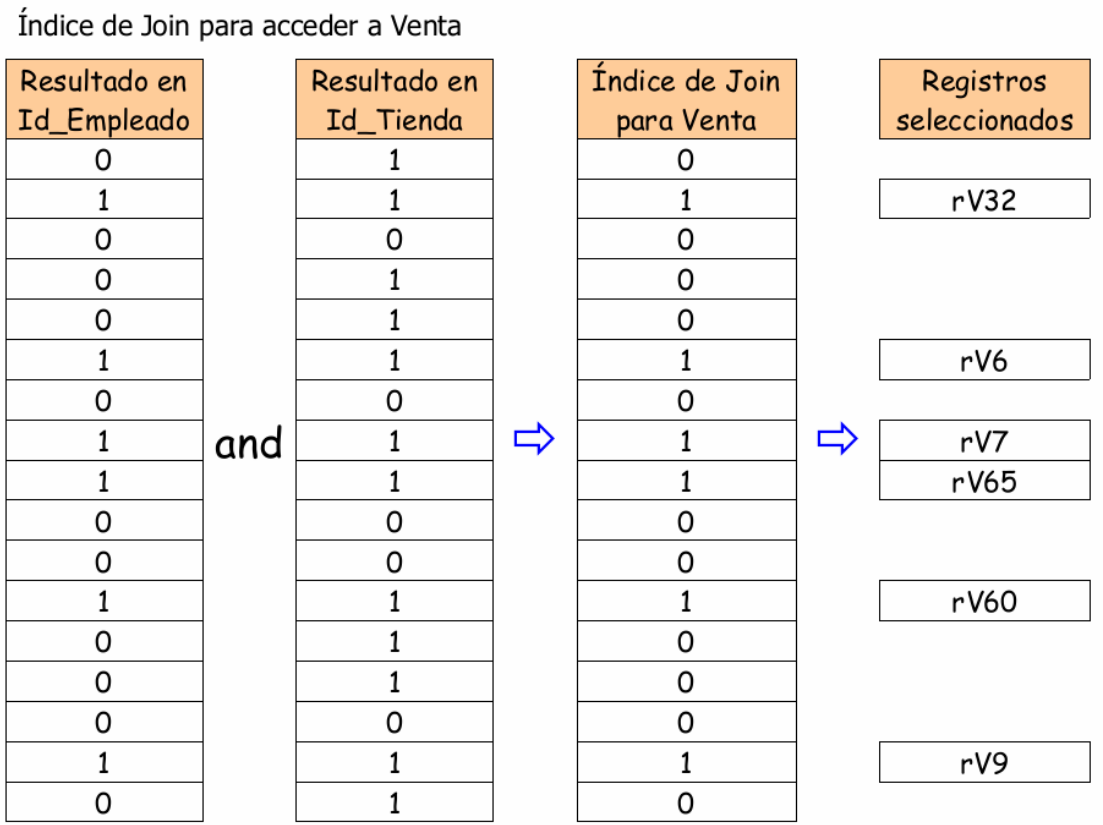
3.4. Optimización y ajuste a nivel físico

- Ejemplo de índice de Join
  - Se crean las columnas del índice de Join



3.4. Optimización y ajuste a nivel físico

- Ejemplo de índice de Join
  - Se combinan las columnas del índice de Join y se accede a los hechos



3.4. Optimización y ajuste a nivel físico

- Índice de Join
  - Planteamiento seguido para la consulta