

# 3 Diseño de cubos OLAP

## 3.1 Cubos de Datos

- 3.1.1 OLAP (On-Line Analytical Processing, Procesamiento Analítico en Línea).
- 3.1.2 Comparativa entre OLAP y OLTP (OnLine Transaction Processing, Procesamiento de Transacciones En Línea)

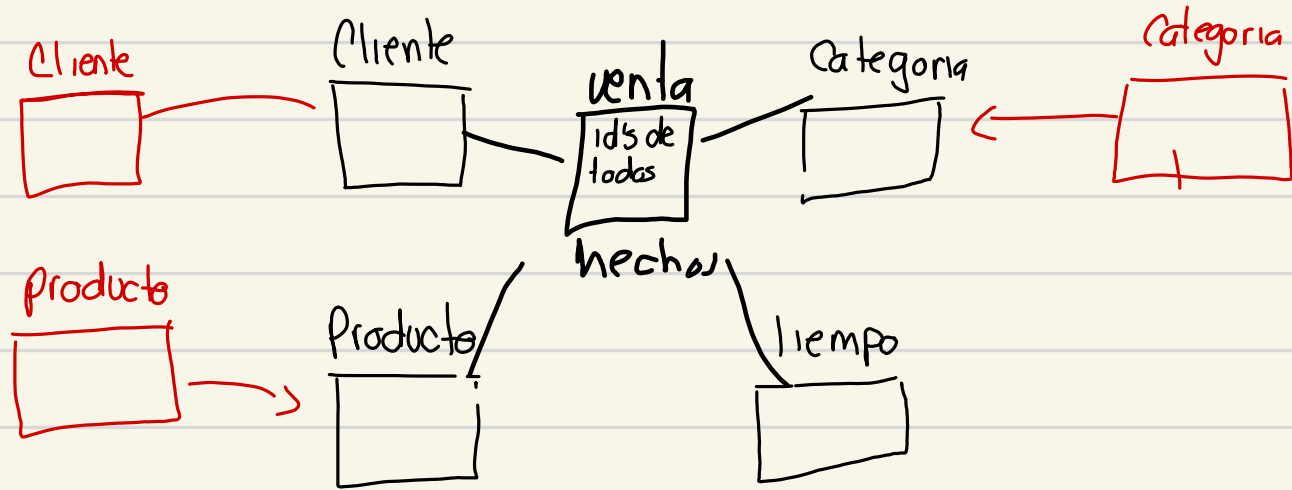
## 3.2 Modelado de Datos para almacenes de datos

- 3.2.1 Modelo Relacional
- 3.2.2 Modelo dimensional OLAP
- 3.2.3 Tipos de Esquemas (Estrellas, copo de nieve y otros)

## 3.3 Operadores de cubos de datos

## 3.4 Cubos multidimensionales

# Practica 8



## 3.1 Cubos de Datos

### 3.1.1 OLAP (On-Line Analytical Processing, Procesamiento Analítico en Línea).

- Es una solución utilizada en la Inteligencia de Negocios cuyo objetivo es agilizar la consulta de grandes cantidades de datos.
  - Para ello utiliza estructuras de datos diversas, normalmente multidimensionales (o Cubos OLAP), que contienen datos resumidos de Sistemas Transaccionales (OLTP).
  - La principal característica que potencia OLAP, es que es lo más rápido al ejecutar sentencias SQL de tipo SELECT, en contraposición con OLTP que es la mejor opción para operaciones de tipo INSERT, UPDATE Y DELETE

### 3.1.2 Comparativa entre OLAP y OLTP (OnLine Transaction Processing, Procesamiento de Transacciones En Línea)



Se hace referencia a las aplicaciones de software que están en el corazón de los sistemas operativos como procesamiento de transacciones en línea (on line transactional processing, OLTP).




Por otro lado, todo el conjunto de herramientas destinadas a realizar análisis de inteligencia de negocios y soporte los procesos de toma de decisiones se den por el nombre de procesamiento analítico en línea (on line analytical processing, OLAP).



Por lo tanto, podemos suponer que la función de un almacén de datos es proporcionar datos de entrada a las aplicaciones OLAP.

# Comparativa OLTP vs. OLAP

Característica	OLTP	OLAP
Volatilidad	datos dinámicos	datos estáticos
Puntualidad	sólo datos actuales	datos actuales e históricos
Dimensión de tiempo	implícito y actual	explícito y variante
Granularidad	datos detallados	datos agregados y consolidados
Actualización	continuo e irregular	periódico y regular
Actividades	Repetitivo	imprevisible
Flexibilidad	Bajo	Alto
Rendimiento	alto, pocos segundos por consulta	puede ser bajo para consultas complejas
Usuarios	Empleados	trabajadores del conocimiento
Funciones	Operacional	analítico
Finalidad del uso	Transacciones	consultas complejas y apoyo a la toma de decisiones
Prioridad	alto rendimiento	alta flexibilidad
Métricas	tasa de transacción	respuesta eficaz
Tamaño	megabytes a gigabytes	gigabytes a terabytes



## 3.2 Modelado de Datos para almacenes de datos

## 3.2.1 Modelo Relacional



# Extensiones al SQL para análisis de datos

- Para facilitar la ejecución de consultas OLAP y la agregación de datos, el estándar SQL-99 introdujo tres extensiones a la instrucción GROUP BY: el operador CUBE, ROLLUP y GROUPING SETS.
- El operador **CUBE** calcula una unión de GROUP BY en cada subconjunto de los tipos de atributos especificados.
- Su conjunto de resultados representa un cubo multidimensional basado en la tabla de origen.
  - La función agregada especificada en la lista de selecciones se aplica a estos grupos para generar valores de resumen para las filas superagregadas adicionales.
  - Se puede aplicar a todas las funciones agregadas, incluidas AVG, SUM, MAX, MIN y COUNT.
- Las consultas deben seguir el siguiente patrón:

```
SELECT c1, c2, c3, aggregate (c4)
FROM table_name
GROUP BY CUBE (c1, c2, c3);
```

# Ejemplo

PRODUCTO	CUARTO	REGIÓN	VENTAS
A	Q1	Europa	10
A	Q1	América	20
A	Q2	Europa	20
A	Q2	América	50
A	Q3	América	20
A	Q4	Europa	10
A	Q4	América	30
B	Q1	Europa	40
B	Q1	América	60
B	Q2	Europa	20
B	Q2	América	10
B	Q3	América	20
B	Q4	Europa	10
B	Q4	América	40

- Considere la siguiente tabla de ventas.
- Ahora podemos formular la siguiente consulta SQL:  

```
SELECT CUARTO, REGION, SUM(VENTAS)
FROM TABLA_VENTAS
GROUP BY CUBE (CUARTO, REGION)
```
- Esta consulta calcula la unión de  $2^2 = 4$  agrupaciones de la tabla de ventas:  
 $\{(cuarto, region), (cuarto), (region), ()\}$ ,
  - donde  $()$  denota una lista de grupos vacía que representa el agregado total en toda la tabla. En otras palabras, dado que cuarto tiene 4 valores y región 2, el multiconjunto resultante tendrá  $4 * 2 + 4 * 1 + 1 * 2 + 1$ , o 15 tuplas, como se puede ver en el resultado. Se han agregado valores NULL en las columnas de dimensión Cuarto y Región para indicar la agregación que tuvo lugar.

CUARTO	REGIÓN	VENTAS
Q1	Europa	50
Q1	América	80
Q2	Europa	40
Q2	América	60
Q3	Europa	NULL
Q3	América	40
Q4	Europa	20
Q4	América	80
Q1	NULL	130
Q2	NULL	100
Q3	NULL	40
Q4	NULL	90
NULL	Europa	110
NULL	América	250
NULL	NULL	360

# Función grouping()

- La función **grouping()** devuelve 1 en caso de que se genere un valor NULL durante la agregación y 0 en caso contrario.
- Esto distingue los NULL generados y los posibles NULL reales derivados de los datos.
- Se puede aplicar en la consulta anterior:

```
SELECT CASE WHEN grouping(CUARTO) = 1  
THEN 'Total' ELSE CUARTO END AS CUARTO,  
CASE WHEN grouping(REGION) = 1 THEN  
'Total' ELSE REGION END AS REGION,  
SUM(VENTAS)  
FROM TABLA_VENTAS  
GROUP BY CUBE (CUARTO, REGION)
```

CUARTO	REGIÓN	VENTAS
Q1	Europa	50
Q1	América	80
Q2	Europa	40
Q2	América	60
Q3	Europa	NULL
Q3	América	40
Q4	Europa	20
Q4	América	80
Q1	Total	130
Q2	Total	100
Q3	Total	40
Q4	Total	90
Total	Europa	110
Total	América	250
Total	Total	360

# Operador ROLLUP

- El operador **ROLLUP** calcula la unión de atributos especificados, desde el más detallado hasta el total general.
- La diferencia clave entre el operador ROLLUP y CUBE es que el primero genera un conjunto de resultados que muestra los agregados para una jerarquía de valores de los atributos especificados, mientras que el segundo genera un conjunto de resultados que muestra los agregados para todas las combinaciones de valores de los atributos seleccionados. Por lo tanto, el orden en que se mencionan los atributos es importante para el ROLLUP, pero no para el operador CUBE. Considere la siguiente consulta:

```
SELECT CUARTO, REGION, SUM(VENTAS)
FROM TABLA_VENTAS
GROUP BY ROLLUP (CUARTO, REGION)
```

- Genera la unión de tres agrupaciones { (cuarto, region), (cuarto), () } donde () representa de nuevo la agregación completa. Por lo tanto, el multiconjunto resultante tendrá  $4 * 2 + 4 + 1$ , o 13 filas. Puede verse que la dimensión de región se enrolla primero seguida de la dimensión de cuarto.

CUARTO	REGIÓN	VENTAS
Q1	Europa	50
Q1	América	80
Q2	Europa	40
Q2	América	60
Q3	Europa	NULL
Q3	América	40
Q4	Europa	20
Q4	América	80
Q1	NULL	130
Q2	NULL	100
Q3	NULL	40
Q4	NULL	90
Total	Total	360

# Operador Grouping Sets

- El operador **GROUPING SETS** genera un conjunto de resultados equivalente al generado por un UNION ALL de múltiples cláusulas simples GROUP BY. Considere el siguiente ejemplo:

```
SELECT CUARTO, REGION, SUM(VENTAS)
FROM TABLA_VENTAS
GROUP BY GROUPING SETS ( (CUARTO), (REGION) )
```

- Esta consulta equivale a:

```
SELECT CUARTO, NULL, SUM(VENTAS)
FROM TABLA_VENTAS
GROUP BY CUARTO
UNION ALL
SELECT NULL, REGION, SUM(VENTAS)
FROM TABLA_VENTAS
GROUP BY REGION
```

CUARTO	REGIÓN	VENTAS
Q1	NULL	130
Q2	NULL	100
Q3	NULL	40
Q4	NULL	90
NULL	Europa	110
NULL	América	250

- Diferentes combinaciones de CUBE, ROLLUP y GROUPING SETS pueden generar conjuntos de resultados equivalentes. Considere la siguiente consulta:

```
SELECT CUARTO, REGION, SUM(VENTAS)
FROM TABLA_VENTAS
GROUP BY CUBE (CUARTO, REGION)
```

- Esta consulta equivale a:

```
SELECT CUARTO, REGION, SUM(VENTAS)
FROM TABLA_VENTAS
GROUP BY GROUPING SETS ((CUARTO, REGION), (CUARTO), (REGION), ( ))
```

- Asimismo, la siguiente consulta:

```
SELECT CUARTO, REGION, SUM(VENTAS)  
FROM TABLA_VENTAS  
GROUP BY ROLLUP (CUARTO, REGION)
```

- es idéntico a:

```
SELECT CUARTO, REGION, SUM(VENTAS)  
FROM TABLA_VENTAS  
GROUP BY GROUPING SETS ((CUARTO, REGION), (CUARTO), ( ))
```

- Dada la cantidad de datos que se agregarán y recuperarán, las consultas OLAP SQL pueden llevar mucho tiempo. Una forma de acelerar el rendimiento es convertir algunas de estas consultas OLAP en vistas materializadas.

# Función Rank()

- Es una función analítica que calcula el rango de un valor en un conjunto de valores.
- La función devuelve el mismo rango para las filas con los mismos valores. Agrega el número de filas coincidentes al rango para calcular el siguiente rango. Por lo tanto, los rangos pueden no ser números consecutivos.
- La función es útil para consultas top-N y bottom-N.
- La sintaxis de la función:

```
RANK() OVER ([ query_partition_clause ]  
order_by_clause)
```



# Ejemplo de funcionamiento

Dato
A
A
B
C
C
C
D

- Considere el contenido de la tabla izquierda.
- Al ejecutar la siguiente consulta, se obtendrán los resultados de la derecha:

```
SELECT col, RANK() OVER (ORDER BY col)  
my_rank FROM rank_demo;
```

- Observe que en los resultados las dos primeras filas recibieron el mismo rango 1. La tercera fila obtuvo el rango 3 porque la segunda fila ya recibió el rango 1. Las siguientes tres filas recibieron el mismo rango 4 y la última fila obtuvo el rango 7.

COL	MY_RANK
A	1
A	1
B	3
C	4
C	4
C	4
D	7

- Considerando la tabla siguiente, se puede ejecutar la siguiente consulta para obtener el listado de los productos de mayor costo incluyendo su rango:

```
SELECT product_name,  
list_price, RANK()  
OVER(ORDER BY list_price  
DESC) FROM products;
```

## PRODUCTS

\* PRODUCT\_ID  
PRODUCT\_NAME  
DESCRIPTION  
STANDARD\_COST  
LIST\_PRICE  
CATEGORY\_ID

PRODUCT_NAME	LIST_PRICE	RANK()OVER(ORDERBYLIST_PRICEDESC)
Intel SSDPECME040T401	8867.99	1
PNY VCQP6000-PB	5499.99	2
PNY VCQM6000-24GB-PB	4139	3
Intel Xeon E5-2699 V3 (OEM/Tray)	3410.46	4
PNY VCQM6000-PB	3254.99	5
ATI FirePro W9000	3192.97	6
ATI FirePro S9150	3177.44	7
AMD FirePro W9100	2998.89	8
EVGA 12G-P4-3992-KR	2799.99	9
Intel Xeon E5-2697 V3	2774.98	10
AMD 100-505989	2699.99	11
Intel Xeon E5-2698 V3 (OEM/Tray)	2660.72	12
Intel Xeon E5-2697 V4	2554.99	13
Intel Xeon E5-2685 V3 (OEM/Tray)	2501.69	14
Intel Xeon E5-2695 V3 (OEM/Tray)	2431.95	15

- Para obtener los 10 productos más caros, se puede usar la siguiente declaración:

```
WITH cte_products AS
(SELECT product_name, list_price, RANK()
OVER(ORDER BY list_price DESC)
price_rank FROM products)
SELECT product_name, list_price,
price_rank FROM cte_products WHERE
price_rank <= 10;
```

- Se usa la expresión WITH para mantener los resultados de la primera consulta aplicados en la segunda

PRODUCT_NAME	LIST_PRICE	PRICE_RANK
Intel SSDPECME040T401	8867.99	1
PNY VCQP6000-PB	5499.99	2
PNY VCQM6000-24GB-PB	4139	3
Intel Xeon E5-2699 V3 (OEM/Tray)	3410.46	4
PNY VCQM6000-PB	3254.99	5
ATI FirePro W9000	3192.97	6
ATI FirePro S9150	3177.44	7
AMD FirePro W9100	2998.89	8
EVGA 12G-P4-3992-KR	2799.99	9
Intel Xeon E5-2697 V3	2774.98	10

# RANK() PARTITION BY

- La función RANK() PARTITION BY permite crear particiones de una columna, de la siguiente manera:
  - Primero, la cláusula divide los productos en múltiples particiones por categoría.
  - Luego, la cláusula ordena las filas de cada partición por precio de lista en orden descendente.
  - Finalmente, la función calcula el rango para cada fila en cada partición. Reinicializa el rango para cada partición.
- En el ejemplo siguiente se devuelven los 3 productos más caros de cada categoría:

```
WITH cte_products AS ( SELECT product name,  
list_price, category_id, RANK() OVER (PARTITION  
BY category_id ORDER BY list_price DESC)  
price_rank FROM products )
```

```
SELECT product name, list_price, category_id,  
price_rank FROM cte_products WHERE price_rank <=  
3;
```

PRODUCT_NAME	LIST_PRICE	CATEGORY_ID	PRICE_RANK
Intel Xeon E5-2699 V3 (OEM/Tray)	3410.46	1	1
Intel Xeon E5-2697 V3	2774.98	1	2
Intel Xeon E5-2698 V3 (OEM/Tray)	2660.72	1	3
PNY VQP6000-PB	5499.99	2	1
PNY VQM6000-24GB-PB	4139	2	2
PNY VQM6000-PB	3254.99	2	3
Supermicro X10SDV-8C-TLN4F	948.99	4	1
Intel DP35DPM	789.79	4	2
Asus X99-E-10G WS	649	4	3
Intel SSDPECME040T401	8867.99	5	1
Crucial	1620.99	5	2
G.Skill TridentZ RGB	1504.99	5	3

## 3.2.2 Modelo dimensional OLAP

Estructura basada en  
tablas

Tiempos de resp  
rápidos

El diseño de DW y DM se basa en un paradigma multidimensional para la representación de datos que proporciona al menos dos ventajas principales:

- en el lado funcional, puede garantizar tiempos de respuesta rápidos incluso a consultas complejas,
- mientras que en el lado lógico las dimensiones coinciden naturalmente con los criterios seguidos por los trabajadores del conocimiento para realizar sus análisis.

La representación multidimensional se basa en un esquema en estrella que contiene dos tipos de tablas de datos: tablas de dimensiones y tablas de hechos.

# Tablas de hechos

Las tablas de hechos suelen hacer referencia a transacciones, y contienen dos tipos de datos:

- Los vínculos -establecidos como pares de llaves primarias (PK) / llaves foráneas (FK) del modelo relacional- a cada tabla de dimensiones que hacen referencia a los datos de la tabla de hechos;
- Los valores numéricos de los atributos que caracterizan a las transacciones y son las medidas que serán empleadas para las operaciones OLAP.

Por ejemplo, una tabla de hechos puede contener transacciones de ventas y hacer referencia a varias tablas de dimensiones, como clientes, puntos de venta, productos, proveedores, tiempo.

- Las medidas de interés correspondientes son atributos como la cantidad de artículos vendidos, el precio unitario y el descuento.
- La tabla de hechos permite a los analistas evaluar las tendencias de las ventas a lo largo del tiempo, ya sea totales, o referidas a un grupo de clientes.

# Tablas de dimensiones

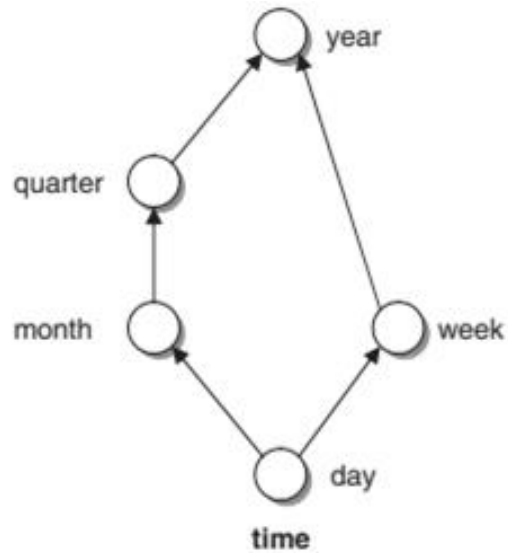
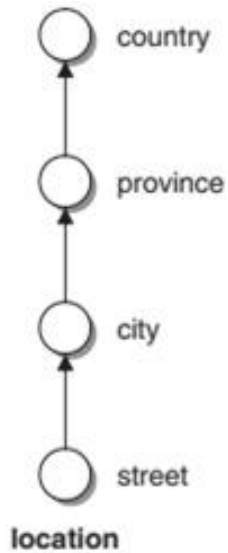
Las tablas de dimensiones están asociadas a las entidades en torno a las cuales giran los procesos de una organización y corresponden a las entidades principales contenidas en el DW.

- En la mayoría de los casos, derivan directamente de las tablas maestras almacenadas en sistemas OLTP, como clientes, productos, ventas, ubicaciones y tiempo.

Cada tabla de dimensiones se estructura interna según las relaciones jerárquicas (si están presentes).

- Por ejemplo, la dimensión temporal se basa normalmente en la jerarquías {día, semana, año}. Del mismo modo, la dimensión de ubicación puede organizarse jerárquicamente como {calle, código postal, ciudad, provincia, región, país, área}.

Las dimensiones predeterminan las principales trayectorias a lo largo de las cuales se desarrollará el análisis OLAP.

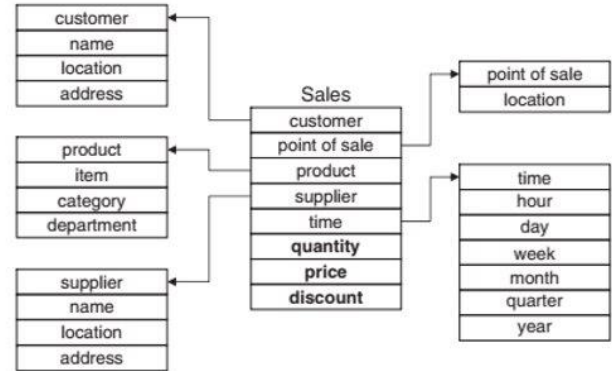


## Jerarquías en un modelo multidimensional

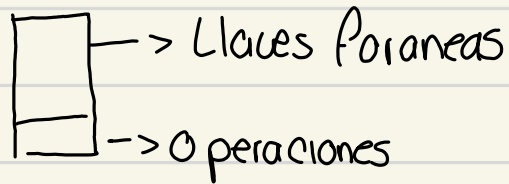


### 3.2.3 Tipos de Esquemas (Estrellas, copo de nieve y otros)

- En el esquema estrella, la tabla de hechos se coloca en el centro del esquema y se enlaza a las tablas de dimensiones a través de las referencias PK/FK.

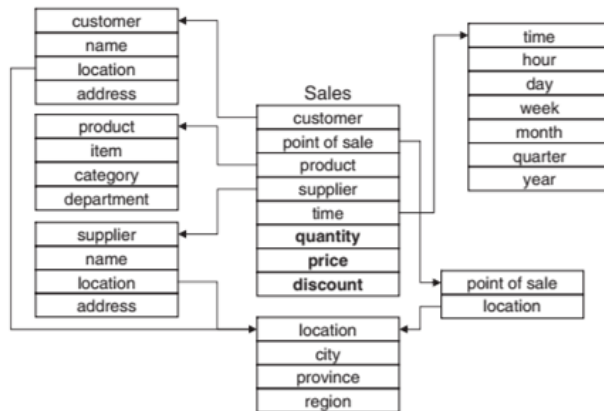
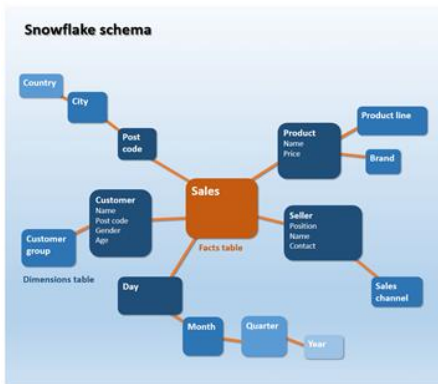


La tab de hechos se coloca a la mitad y al rededor las dim



# Esquema copo de nieve

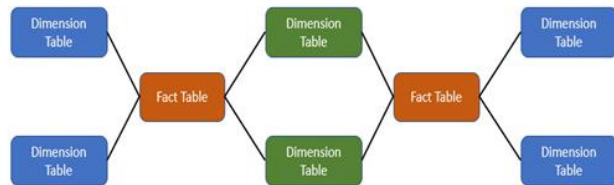
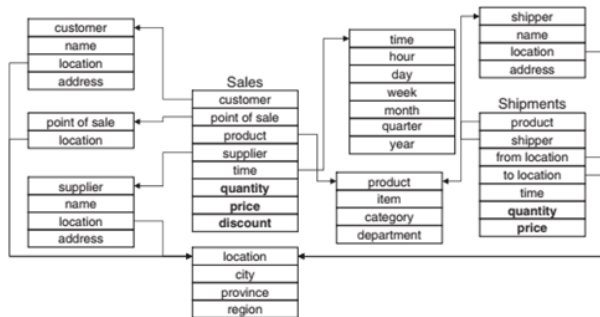
- En el esquema copo de nieve, las tablas de dimensiones se conectan a su vez a otras tablas de dimensiones, con el fin de reducir el uso de memoria



Tiene 1 tab de hechos, pero las tab de dim hace referencia a otra para evitar redundancias

# Esquema galaxia

- El esquema galaxia incluye varias tablas de hechos, interconectadas con tablas de dimensiones, vinculadas a su vez con otras tablas de dimensiones



## 3.3 Operadores de cubos de datos

# Jerarquías de conceptos y operaciones OLAP

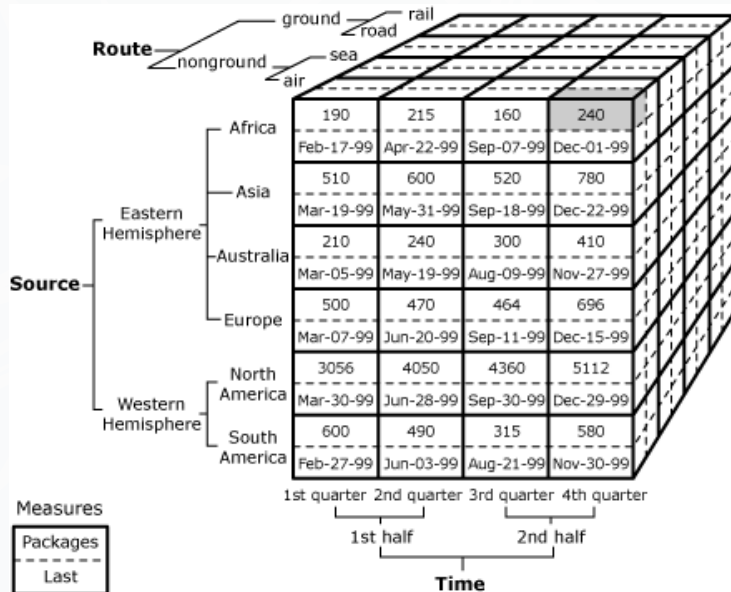
Una jerarquía de conceptos define un conjunto de mapeos de un nivel inferior de conceptos a un nivel superior.

- Por ejemplo, la dimensión {localización} puede originar una jerarquía totalmente ordenada, que se desarrolla a lo largo de la relación {dirección, municipio, provincia, país}

Los análisis OLAP se pueden basar en jerarquías de conceptos para consolidar los datos y crear vistas lógicas a lo largo de las dimensiones de un DW.

- Los tipos de jerarquía específicos pueden estar predefinidos en la plataforma de software utilizada para la creación y administración de un almacén de datos. Para otras jerarquías es necesario que los analistas definan explicativamente las relaciones entre los conceptos.

Las jerarquías de conceptos también se usan para realizar varias operaciones de visualización que tratan con cubos de datos en un DW.



# Cubo de datos



# Drill-down

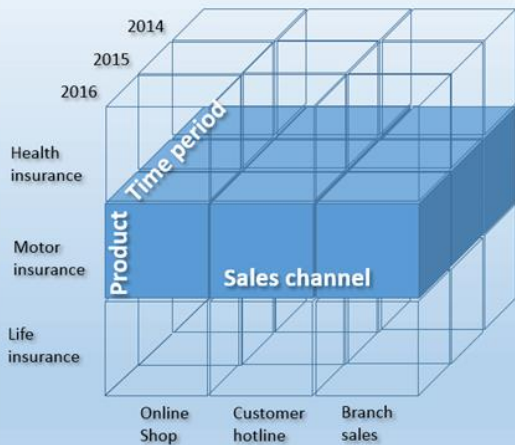
Permite la navegación a través de un cubo de datos desde información agregada y consolidada hasta información más detallada.

Una operación de desglose puede llevarse a cabo de dos maneras:

- Bajar a un nivel inferior a lo largo de una jerarquía de una sola dimensión.
  - Por ejemplo, en el caso de la dimensión {ubicación}, es posible pasar del nivel {provincia} al nivel {ciudad} y desagregar las medidas temáticas de interés sobre todos los registros en los que la ciudad pertenece a la misma provincia.
- Agregar una dimensión.
  - Por ejemplo, la introducción de la dimensión {time} lleva a desagregar las medidas de interés en todos los períodos de tiempo existentes en un cubo de datos.

# Drill-down

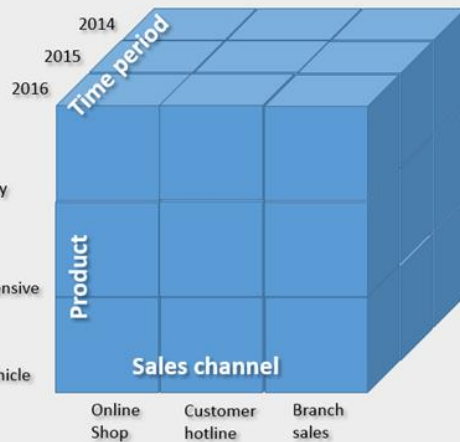
## Drill Down



Motor  
Insurance  
- third party

Motor  
insurance  
- comprehensive

Motor  
insurance  
- second vehicle



# Roll-up

Consiste en una agregación de datos en el cubo, que se puede obtener alternativamente en las dos vías siguientes.

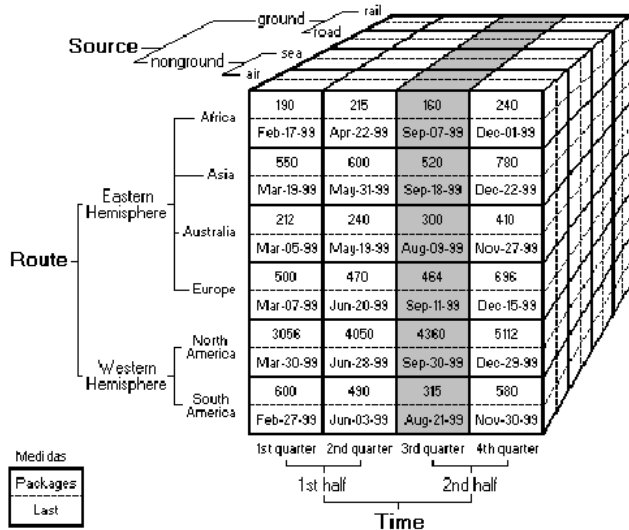
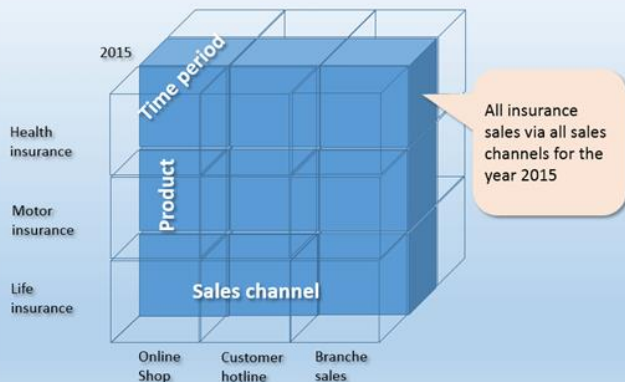
- Avanzar hacia arriba a un nivel superior a lo largo de una única dimensión definida en una jerarquía de conceptos.
  - Por ejemplo, para la dimensión {ubicación} es posible pasar del nivel {ciudad} al nivel {provincia} y consolidar las medidas de interés a través de una suma condicionada grupo por todos los registros en los que la ciudad pertenece a la misma provincia.
- Reducción en una dimensión.
  - Por ejemplo, la eliminación de la dimensión {time} conduce a medidas consolidadas a través de la suma durante todos los períodos de tiempo existentes en el cubo de datos.

Es la operación opuesta a drill-down.

# Slice

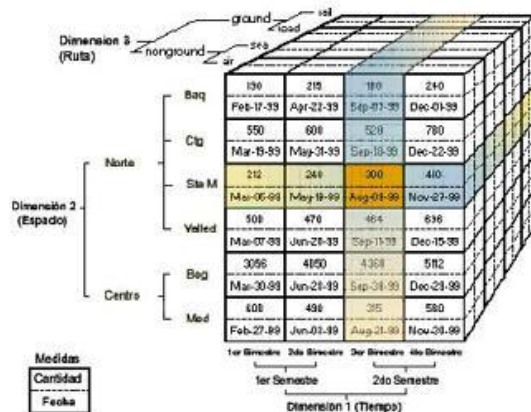
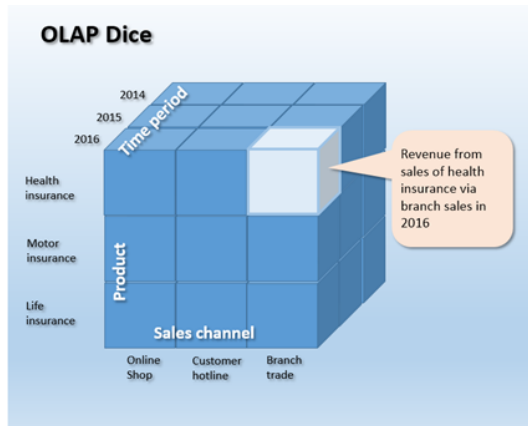
El valor de un atributo se selecciona y se fija a lo largo de una dimensión.

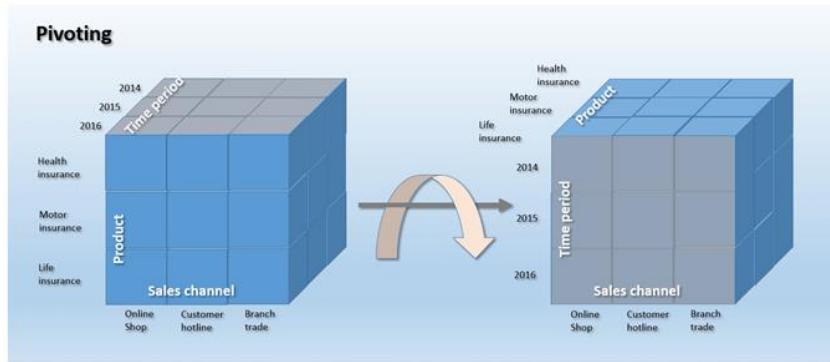
## Slicing



# Dice

- Obtiene un cubo en un subespacio seleccionando varias dimensiones simultáneamente.





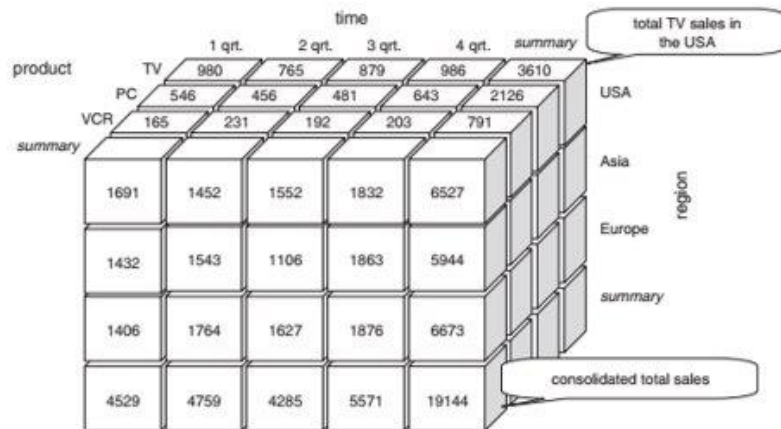
## Pivot

- Produce una rotación de los ejes, intercambiando algunas dimensiones para obtener una vista diferente de un cubo de datos.

## 3.4 Cubos multidimensionales

Una tabla de hechos conectada con n tablas de dimensiones puede representarse mediante un cubo de datos n-dimensional, donde cada eje corresponde a una dimensión.

Los cubos multidimensionales son una extensión natural de las hojas de cálculo populares de dos dimensiones, que se pueden interpretar como cubos bidimensionales.





## Multidimensionalidad

---

Un cubo de datos de cuatro dimensiones no se puede representar gráficamente.

---

Sin embargo, se pueden obtener cuatro vistas lógicas compuestas por cubos tridimensionales, llamados cuboides, dentro del cubo de cuatro dimensiones, fijando los valores de unidimensionalidad.

---

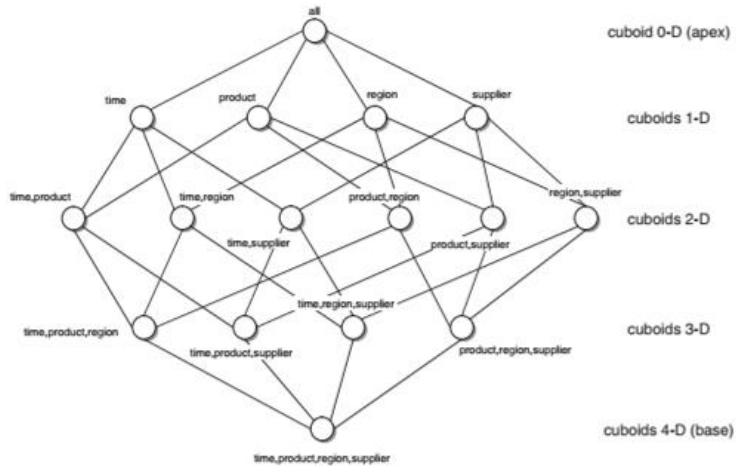
Más generalmente, a partir de una tabla de hechos vinculada a  $n$  tablas de dimensiones, es posible obtener una red de cuboides, cada uno de ellos correspondiente a un nivel diferente de consolidación a lo largo de una o más dimensiones.

---

Este tipo de agregación es equivalente en SQL a una suma de consulta derivada de una condición de agrupamiento.

# Lattice multidimensional

- El cuboide asociado a los datos atómicos, que por lo tanto no implica ningún tipo de consolidación, se denomina cuboide base.
- En el otro extremo, el cuboide ápice se define como el cuboide correspondiente a la consolidación a lo largo de todas las dimensiones, por lo tanto asociado con el total general de la medida de interés.



# Arquitectura MOLAP

---

Almacenan y proporcionan acceso a datos y metadatos en formato multidimensional

---

Especialmente preparadas para entregar datos en estructuras adecuadas al proceso analítico

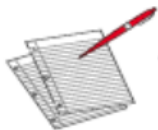
---

No llegan a compararse con ROLAP en potencia y capacidad de almacenamiento cuando hay muchas dimensiones

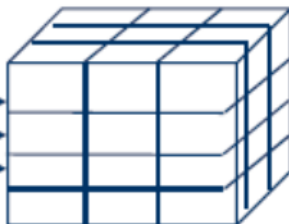
---

Carga más difícil

Sistemas de OLTP

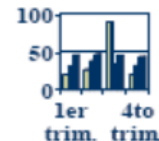
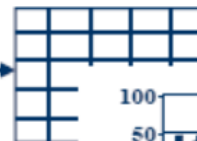


Almacén de datos Multidimensional



Nivel Base de Datos y Lógica de Aplicación

Interfaz OLAP



Nivel Presentación

# Arquitectura ROLAP

---

Se integra naturalmente con la tecnología y estándares existentes.

---

Soporta consultas ad-hoc con eficiencia

---

Actualización más difícil (carga en tablas relacionales)

---

La eficiencia puede ser mejorada empleando técnicas como la codificación y la compresión

---

Superior en paralelismo, concurrencia y distribución

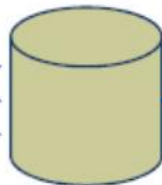
---

Puede tomar ventaja de la tecnología relacional paralela

Sistemas de OLTP



Almacén de datos Relacional



Nivel Base de Datos

Motor ROLAP

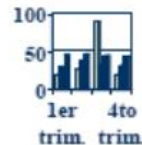
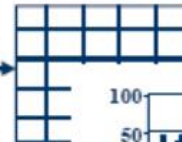


Nivel Lógica de Aplicación

Interfaz OLAP



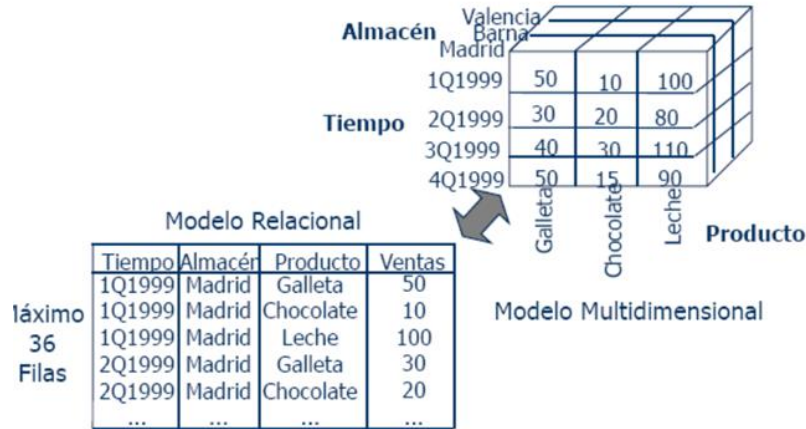
1er trim.  
2do trim.  
3er trim.  
4to trim.



Este  
Oeste  
Norte

Nivel Presentación

# MOLAP vs ROLAP



MOLAP	ROLAP
2 Niveles	3 Niveles
10 Dimensiones	Más Dimensiones
Volúmenes Medios (10 GB)	Grandes Volúmenes de datos
Mayor necesidad de procesos por lotes (batch)	Mayores costes en cada consulta, se requiere recalcular
Rendimiento crítico	Alta volatilidad de datos
Baja escalabilidad	Alta Escalabilidad
Implantación directa y clara del modelo multidimensional	Implantación simulada, se requiere capa intermedia

# MultiDimensional eXpressions

- ❓ MDX es un lenguaje de consultas OLAP creado en 1997 por Microsoft. No es un estándar pero diversos fabricantes lo han adoptado como el estándar de facto.
- ❓ La principal diferencia del mundo OLAP respecto al mundo relacional radica en que las estructuras dimensionales están jerarquizadas y se representan en forma de árbol y por lo tanto existen relaciones entre los diferentes miembros de las dimensiones.
- ❓ Tiene similitudes con el lenguaje SQL, incluyendo funciones y fórmulas especiales orientadas al análisis de estructuras jerarquizadas que presentan relaciones entre los diferentes miembros de las dimensiones.



# Sintaxis de MDX

- ❓ La sintaxis de MDX es compleja; la mejor manera de ejemplificarla es a través de un caso determinado. Imaginemos un cubo de ventas con las siguientes dimensiones:
  - Temporal de las ventas con niveles de año y mes.
  - Productos vendidos con niveles de familia de productos y productos en sí.
  - Medidas: importe de las ventas y unidades vendidas.
- ❓ Para obtener, por ejemplo, el importe de las ventas para el año 2018 para la familia de productos lácteos, la consulta sería:

```
SELECT {[medidas].[importe ventas]} on columns,  
{[tiempo].[2018]} on rows  
FROM [cubo_ventas]  
WHERE ([Familia].[lácteos])
```

# Consultas MDX

❓ Es posible observar que la estructura general de una consulta es de la forma

`SELECT... FROM... WHERE...:`

- En el *select* se especifica el conjunto de elementos que se van a visualizar y debe especificarse si se devuelve en columnas o filas.
- En el *from*, el cubo de donde se extrae la información.
- En el *where*, las condiciones de filtrado.
- { } permite crear listas de elementos en las selecciones.
- [ ] encapsulan elementos de las dimensiones y niveles.

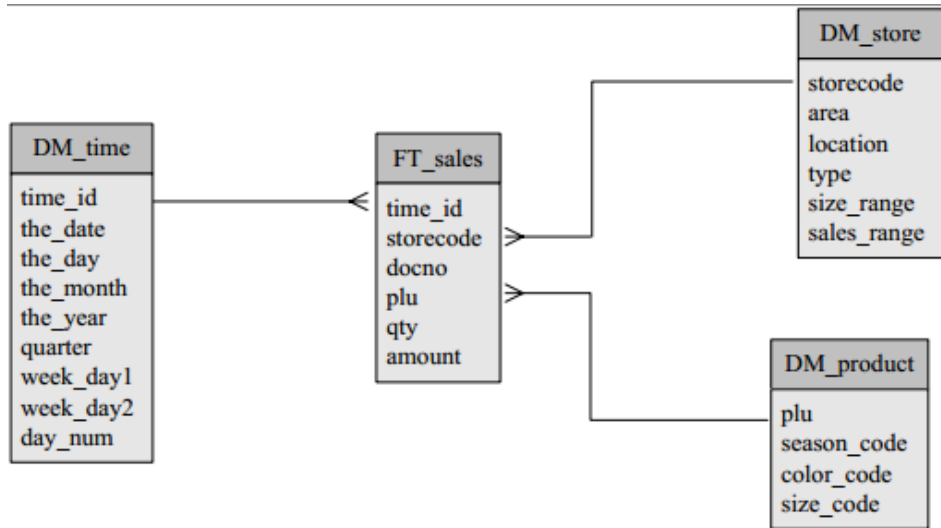
# Ejemplo

- ❏ Obtener las ventas totales en los países de Francia, Alemania y Reino Unido:

```
SELECT Measures.[Internet Sales Amount]
on COLUMNS,
{{[Customer].[Country].[France],
[Customer].[Country].[Germany],
[Customer].[Country].[United Kingdom]}}
on ROWS
FROM [Adventure Works]
```

Country	Internet Sales Amount
France	120,000
Germany	999,999
United Kingdom	55,000

## Esquema multidimensional



## Ejemplo roll-up

### MDX

```
SELECT [SALES].[AMOUNT] ON  
COLUMNS,  
[store].[Houston] ON ROWS  
FROM SALES
```

### SQL

```
select sum(amount), area  
from SALES  
where (area='Houston') group by area
```

# Ejemplo drill-down

## MDX

```
SELECT [SALES].[AMOUNT] ON COLUMNS,  
[time].[2003].[Q4].[Dec].[31],  
[time].[2003].[Q4].[Dec].[30],... ...,  
[time].[2003].[Q4].[Dec].[2],  
[time].[2003].[Q4].[Dec].[1] ON ROWS FROM  
SALES
```

## SQL

```
select sum(amount), the_date  
from SALES  
where (the_date='2003-Dec-31')  
or (the_date='2003-Dec-30')  
or... ..or (the_date='2003-Dec-2')  
or (the_date='2003-Dec-1') group by  
the_date
```

## Ejemplo slice

### MDX

```
SELECT [SALES].[AMOUNT] ON COLUMNS,  
[store].[Houston].[292] ON ROWS FROM  
SALES
```

### SQL

```
select sum(amount), storecode  
from SALES  
where (storecode='292') group by  
storecode
```

## Ejemplo dice

### MDX

```
SELECT [SALES].[AMOUNT] ON COLUMNS,  
[store].[HK],[store].[NT],  
[store].[Houston] ON ROWS  
FROM SALES WHERE  
[time].[2003].[Q4].[Dec].[24]
```

### SQL

```
select sum(amount), area  
from SALES  
where ( (area='HK') or (area='NT') or  
(area='Houston'))  
and (the_date='2003-Dec-24')  
group by area
```



# Principales funciones y elementos MDX

- ❓ El objetivo de MDX está orientado a temas de comparaciones y relaciones jerárquicas entre elementos
- ❓ Una de las funcionalidades que distinguen al MDX es el acceder a los elementos utilizando estructura de árbol. Así para un determinado nivel de una dimensión tenemos comandos como:
  - **[Familia].[lácteos].CurrentMember** : Permite acceder al miembro actual
  - **[Familia].[lácteos].Children** : Permite acceder a los hijos de la familia de los lácteos
  - **[Familia].[lácteos].prevMember**: Permite acceder al miembro anterior de la dimensión

# Funciones de MDX

- MDX incluye múltiples funciones para realizar consultas a través de la jerarquía existente en el esquema OLAP. Podemos destacar entre ellas:
- *CurrentMember*: permite acceder al miembro actual.
  - *Children*: permite acceder a todos los hijos de una jerarquía.
  - *prevMember*: permite acceder al miembro anterior de la dimensión.
  - *CrossJoin(conjunto\_a,conjunto\_b)*: obtiene el producto cartesiano de dos conjuntos de datos.
  - *BottomCount(conjunto\_datos,N)*: obtiene un número determinado de elementos de un conjunto, empezando por abajo, opcionalmente ordenado.
  - *BottomSum(conjunto\_datos,N,S)*: obtiene de un conjunto ordenado los N elementos cuyo total es como mínimo el especificado (S).

# Miembros calculados en MDX

- ❓ Una de las funcionalidades más potentes que ofrece el lenguaje MDX es la posibilidad de realizar cálculos complejos tanto dinámicos (en función de los datos que se están analizando en ese momento) como estáticos.
- ❓ Los cubos multidimensionales trabajan con medidas (del inglés measures) y con miembros calculados (calculated members).
  - Las medidas son las métricas de la tabla de hechos a las que se aplica una función de agregación (count, distinct count, sum, max, avg, etc.).
- ❓ Un miembro calculado es una métrica que tiene como valor el resultado de la aplicación de una fórmula que puede utilizar todos los elementos disponibles en un cubo, así como otras funciones de MDX disponibles.

# Funciones de MDX

- ❓ *Except(conjunto\_a,conjunto\_b)*: obtiene la diferencia entre dos conjuntos.
- ❓ AVG COUNT VARIANCE VARIANCE y todas las funciones trigonométricas (seno, coseno, tangente, etc.).
- ❓ *PeriodsToDate*: devuelve un conjunto de miembros del mismo nivel que un miembro determinado, empezando por el primer miembro del mismo nivel y acabando con el miembro en cuestión, de acuerdo con la restricción del nivel especificado en la dimensión de tiempo.
- ❓ *WTD(<Miembro>)*: devuelve los miembros de la misma semana del miembro especificado.
- ❓ *MTD(<Miembro>)*: devuelve los miembros del mismo mes del miembro especificado.
- ❓ *QTY(<Miembro>)*: devuelve los miembros del mismo trimestre del miembro especificado.
- ❓ *YTD(<Miembro>)*: devuelve los miembros del mismo año del miembro especificado.
- ❓ *ParallelPeriod*: devuelve un miembro de un periodo anterior en la misma posición relativa que el miembro especificado.

# Consideraciones del Modelo MD

## Ventajas

- Modelo simple, expresado en terminología de negocio y usuario
- Permite realizar consultas muy complejas que van a combinar varias dimensiones de información

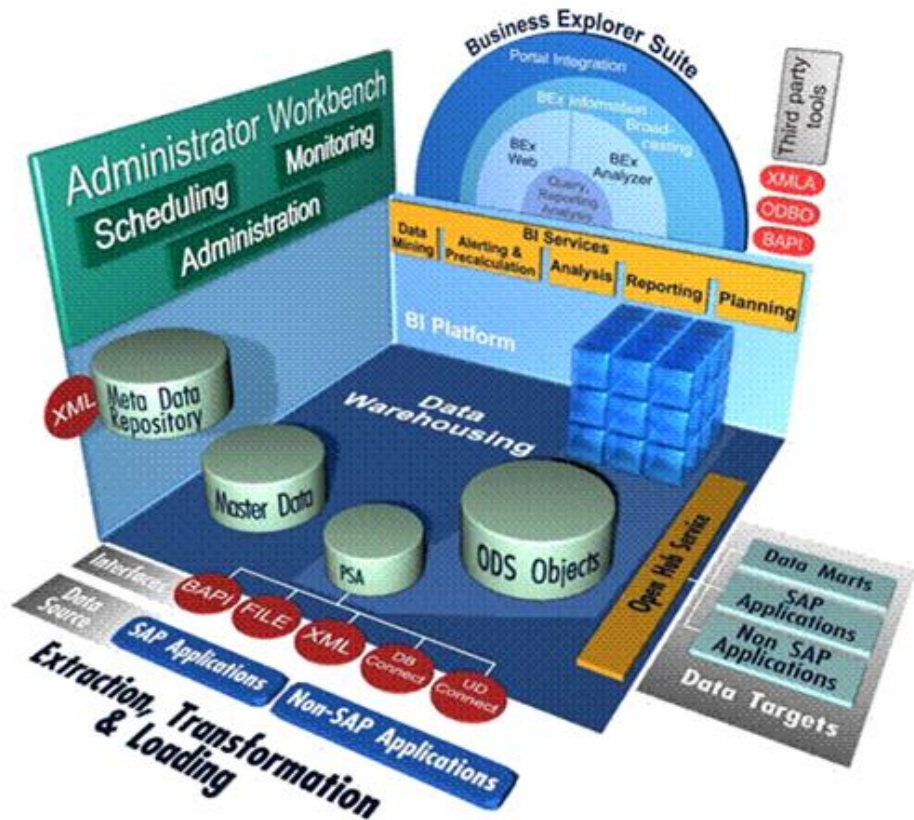
## Inconvenientes

- El espacio de almacenamiento crece (densidad)
- Deben relacionarse las dimensiones adecuadas
- El nivel de detalle de los análisis debe ser el adecuado
- El modelo podrá evolucionar con el tiempo, pero los datos deben mantenerse
- El diseño no es tan fácil como puede parecer, y, aunque puede modificarse, muchas decisiones tomadas son críticas, ya que no podremos prescindir de los datos cargados

# Magic Quadrant for Analytics and Business Intelligence Platforms



SAP



# Oracle

**Oracle BI Publisher**  
(Reporting / Publishing)



**Oracle Interactive Dashboards**



**Oracle Answers**  
(Ad Hoc Analysis)



**Oracle BI Server**

- Simplified business view and unified BI metadata
- Calculation and integration engine
- Federated, optimized data access

**Oracle Database**



**Oracle Warehouse Builder**  
(ETL)

**Data Sources**



OLTP Systems



Data Marts



Apps



Files  
Excel  
XML

## Oracle BI Applications

An Enterprise Performance Management System

Reporting — Scorecarding — Analytic Apps — Planning

Modeling

Planning

Budgeting

Financial Management

Consolidation

Scorecards

Sales & Marketing

Service & Contact Center

Human Resources

Order Management & Fulfillment

Supply Chain

Verticals

Common Enterprise Information Model

Oracle Business Intelligence Foundation



OLTP & ODS Systems



Data Warehouse  
Data Mart



Essbase



SAP, Oracle  
PeopleSoft, Siebel,  
Custom Apps



Files  
Excel  
XML



Business Process



# Cuadro de Mando

Se entiende por cuadro de mando (o dashboard) al sistema que informa de la evolución de los parámetros fundamentales de negocio de una organización o de un área del mismo.

Un cuadro de mando está formado principalmente por diversos elementos combinados:

- Tabla: tiene forma de matriz y permite presentar una gran cantidad de información. La tabla puede ser estática, dinámica, o incluso un análisis
- OLAP. Se persigue con este elemento presentar información de forma estructurada al usuario final.
- Métricas: valores que recogen el proceso de una actividad o los resultados de la misma. Estas medidas proceden del resultado de la actividad de negocio. Como ya sabemos, existen diferentes tipos de métricas. En un cuadro de mando, se suelen usar KPI.
- Listas: comúnmente formadas por KPI. En caso de que el cuadro de mando sólo esté formado por este tipo de elemento, se denomina scorecard.

# Diseño de cuadros de mando

Tanto los informes como los resultados OLAP son herramientas que proporcionan información a los usuarios finales. La gran cantidad de información que normalmente incluyen estas herramientas puede hacerlas inadecuadas para usuarios que necesiten tomar decisiones de forma rápida a partir de ellas.

El cuadro de mando proviene del concepto francés tableau du bord, y permite mostrar información consolidada a alto nivel. Se enfoca en:

- Presentar una cantidad reducida de aspectos de negocio.
- Uso mayoritario de elementos gráficos.
- Inclusión de elementos interactivos para potenciar el análisis en profundidad y la comprensión de la información consultada.

Los cuadros de mando son una herramienta muy popular dado que permiten entender muy rápidamente la situación de negocio y son muy atractivos visualmente.

# Elementos de un cuadro de mando

- **Gráficos:** persigue el objetivo de mostrar información con un alto impacto visual que sirva para obtener información agregada o sumariada con mucha más rapidez que a través de tablas. El gráfico puede estar formado por la superposición de diferentes tipos de visualización
- **Mapas:** este elemento permite mostrar información geolocalizada. No toda la información es susceptible de estar en este tipo de formato. Se combina con otros elementos para presentar el detalle de la información.
- **Alertas visuales y automáticas:** consisten en alertas que informan del cambio de estado de información. Pueden estar formadas por elementos gráficos como fechas o colores resultados, y deben estar automatizadas en función de reglas de negocio encapsuladas en el cuadro de mando.
- **Menús de navegación:** facilitan al usuario final realizar operaciones con los elementos del cuadro de mando.

# Ventajas

- La implantación de un cuadro de mando integral proporciona los siguientes beneficios:
  - Define y clarifica la estrategia.
  - Suministra una imagen del futuro mostrando el camino que conduce a él.
  - Comunica la estrategia a toda la organización.
  - Permite alinear los objetivos personales con los departamentales.
  - Facilita la vinculación entre el corto y el largo plazo.
  - Permite formular con claridad y sencillez las variables más importantes objeto de control.
  - Constituye un instrumento de gestión.
  - Facilita el consenso en toda la empresa gracias a su capacidad de explicitar un modelo de negocio y traducirlo en indicadores.
  - Se puede utilizar para comunicar los planes de la empresa, aunar los esfuerzos en una sola dirección y evitar la dispersión.
  - Permite detectar de forma automática desviaciones en el plan estratégico u operativo, e incluso indagar en los datos operativos de la compañía hasta descubrir la causa original que dio lugar a esas desviaciones.

