

# Data warehouse y OLAP

---

Marta Zorrilla  
Universidad de Cantabria



## Tabla de contenido

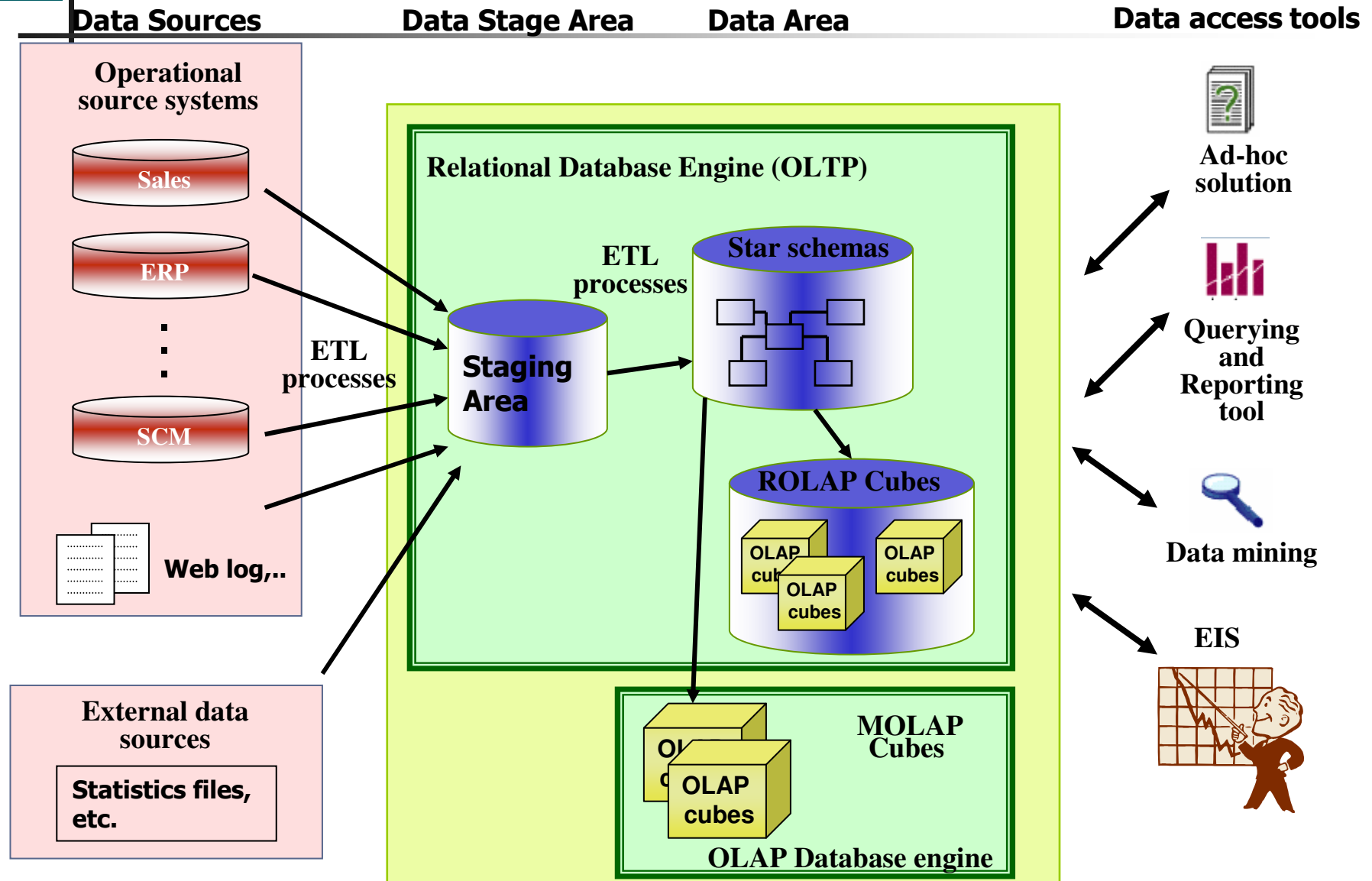
---

- Ciclo de vida de un sistema BI/DW
- Diseño multidimensional
  - Modelo dimensional básico
  - Data warehouse vs data marts
- Procesos ETL
- Cubos OLAP
- Soporte para BD dimensionales en SQL:1999  
Y SQL:2003



# Ciclo de vida BI/DW

# Componentes de un sistema BI/DW

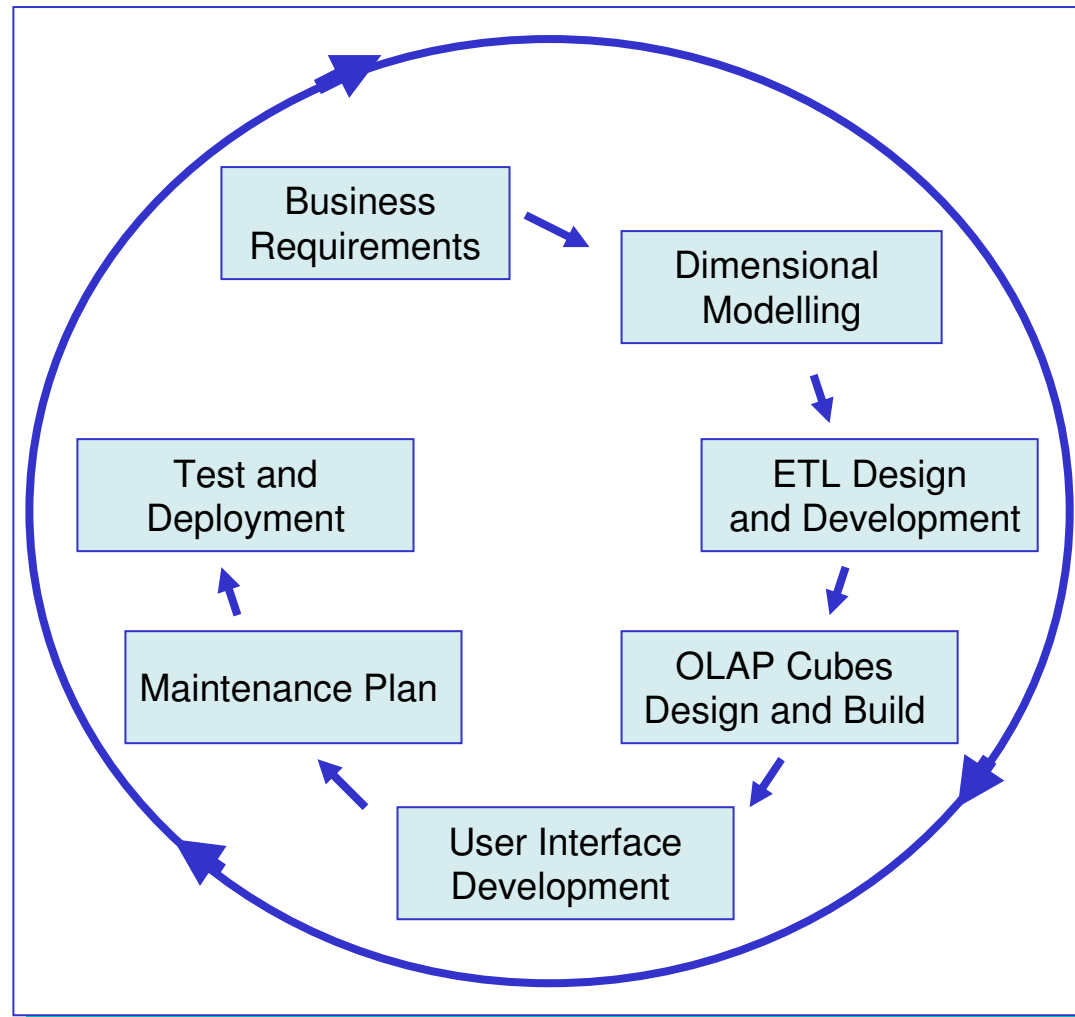


## ¿Qué es una solución BI/DW?

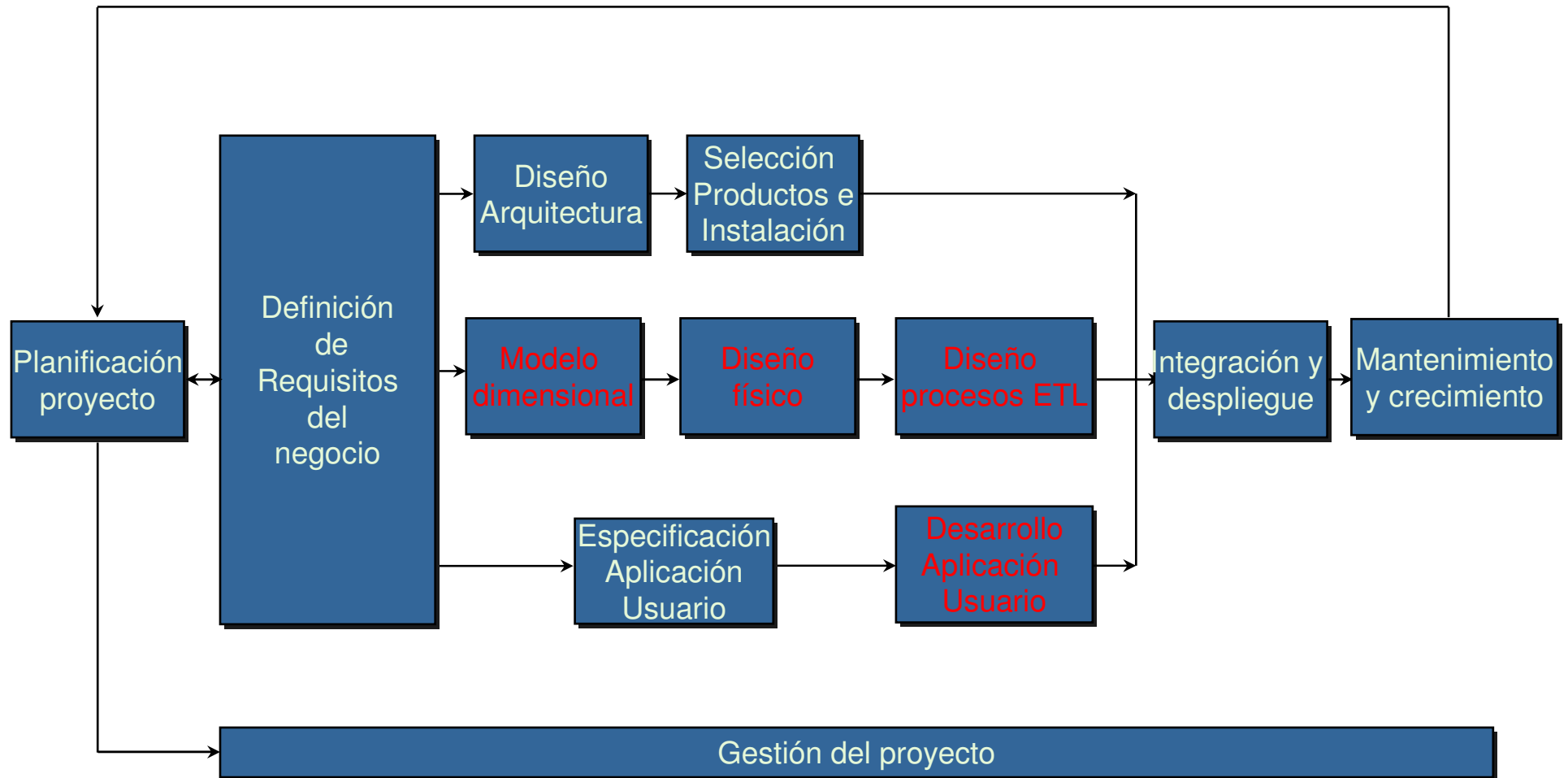
---

- Recoger los datos de la organización y transformarlos en información útil.
  - Estudiar la naturaleza del negocio
    - Indicadores, medidas, dimensiones (perspectivas de análisis)
  - Diseñar la estructura de datos dimensional
  - Recuperar los datos de los sistemas operacionales, transformarlos y cargarlos en la estructura de datos diseñada
  - Usar herramientas para el análisis de los datos y la toma de decisiones

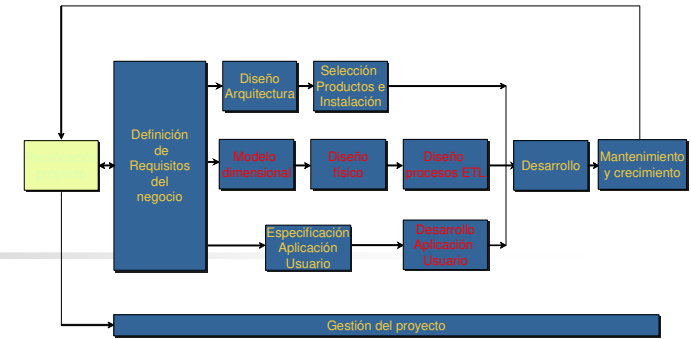
# Modelo de procesos



# Ciclo de vida de un BI/DW (Kimball)

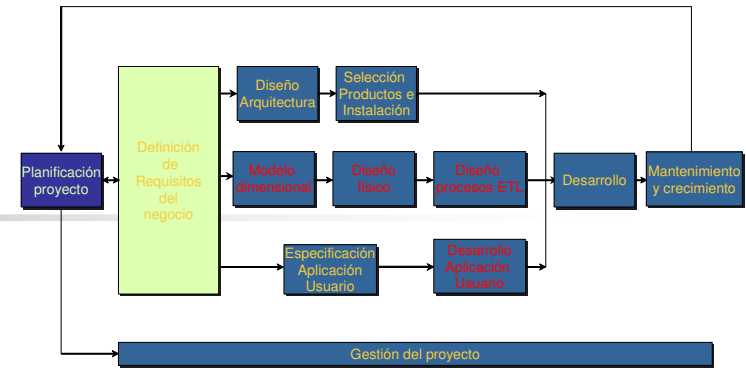


# Planificación del proyecto



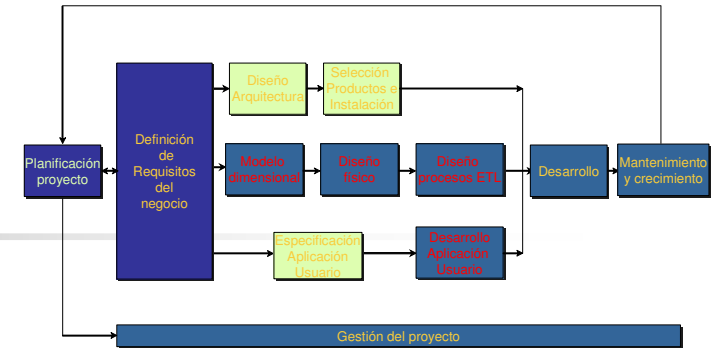
- Acometer la realización de un proyecto global ha conducido al fracaso. Mejor ir por áreas de negocio.
- A tener en cuenta:
  - Hacer partícipes a usuarios operacionales, técnicos IT y analistas.
    - Es importante hacer labor de mentalización antes de comenzar, creando expectativas realistas
    - Expertos y analistas de negocio aseguran la calidad del producto final
    - Usuarios aseguran la calidad de los datos
    - Técnicos dan soporte al sistema
  - Seleccionar una aplicación piloto con una alta probabilidad de éxito
    - Estudiar la viabilidad
    - Estudiar el origen de los datos
    - Estudiar el hardware y software disponible
    - Planificar el entrenamiento de los usuarios
  - Construir prototipos rápida y frecuentemente
  - Reportar activamente y publicar los casos exitosos
  - Herramientas para visualizar los datos fáciles de usar

# Definición de requisitos



- Lista de cuestiones a responder, informes a generar:
  - Determinar indicadores
    - Deben reflejar lo que se quiere medir y, por tanto, hay que tener una definición clara y concisa para su cálculo.
  - Y sus perspectivas de observación para distintos grupos de usuarios
    - Ventas por región > país > continente
    - Ventas por mes > trimestre > año

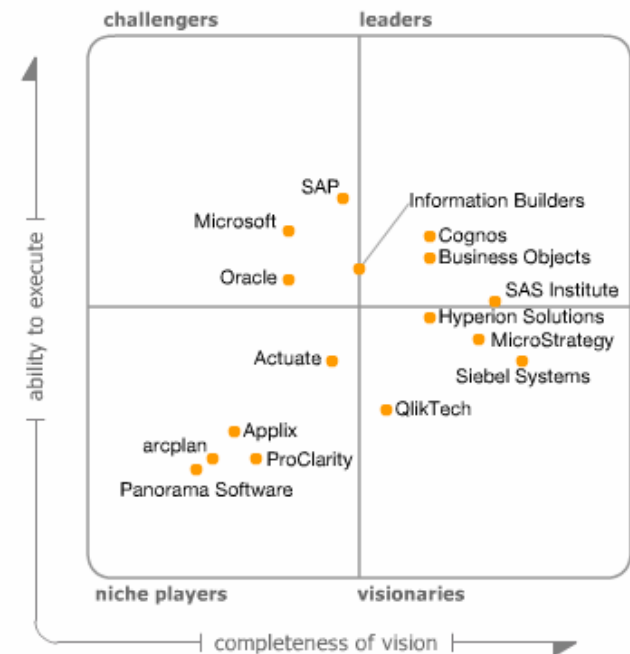
# Herramientas del mercado



- Fuentes de datos: sistemas operacionales corporativos y departamentales, fuentes externas, ficheros, etc.
- ETL: Ascential DataStage, Microsoft Data Transformation Services, DB2 Warehouse, ...
- SGBD Relacional: SQL Server, Oracle, Informix, DB2,...
- OLAP: Analysis Services, Cognos, Business Objects, Microstrategy, Excel, SAS, etc.
- Data mining: SAS, SPSS/Clementine, Analysis Services, Weka, etc.

Source: Gartner  
(December 2005)

## Magic Quadrant for BI Platforms 1H06



As of December 2005

## Evaluación de herramientas

---

- Existen varias herramientas que permiten implementar un Data Warehouse. Todas con parecidas prestaciones, su diferencia está en el precio por número de licencias y la plataforma de trabajo.
- Su elección dependerá, sobre todo, del software/hardware ya disponible.
- La suite comercial más económica: BI SQL Server 2005 de Microsoft.
- Herramientas EIS son bastantes caras. Excel Add-ins, la alternativa de cliente más sencilla.
  - PivotTable de Excel no suficiente
  - XLCubed, IntelligentApps, MIS AG: comerciales, muy potentes
  - PALO: open source, menos prestaciones
- BI platform - Open source ([www.sourceforge.net](http://www.sourceforge.net)):
  - BEE project (ETL, OLAP, MySQL).
  - Pentaho project: reporting, analysis, dashboard, data mining and workflow (firebird RDBMS, Weka DM, Mondrian OLAP, Enhydra ETL, JaWE workflow, BIRT reporting components)

# Modelo de datos dimensional

---

Marta Zorrilla  
Universidad de Cantabria



# Modelo de datos relacional (I)

## Cientes

Codigo_cliente	Nombre	Direccion	Codigo_empresa
A-234	Luis Aja	Alta, 5 Stder.	E-54
A-741	Ana Ros	Pez, 21 Madrid	E-33
A-562	José Maza	Ercilla, 3 Bilbao	E-54

Los datos se conciben agrupados en forma de tablas

Cada fila establece una relación entre un conjunto de valores

Operadores generan nuevas tablas

**SELECT** Nombre, Direccion **FROM** Cientes  
**WHERE** Codigo\_empresa = "E-54"

Nombre	Direccion
Luis Aja	Alta, 5 Stder.
José Maza	Ercilla, 3 Bilbao

# Modelo de datos relacional (II)

Criterio de diseño:  
**No repetir datos innecesariamente (Normalización)**

## Cientes

Codigo_cliente	Nombre	Direccion	Codigo_empresa
A-234	Luis Aja	Alta, 5 Stder.	E-54
A-741	Ana Ros	Pez, 21 Madrid	E-33
A-562	José Maza	Ercilla, 3 Bilbao	E-54

- Toda tabla tiene una columna o conjunto de columnas que permiten identificar cada una de sus filas; éstas componen la llamada **clave principal** de la tabla.

- Los valores de la clave principal no se pueden repetir.

## Facturas

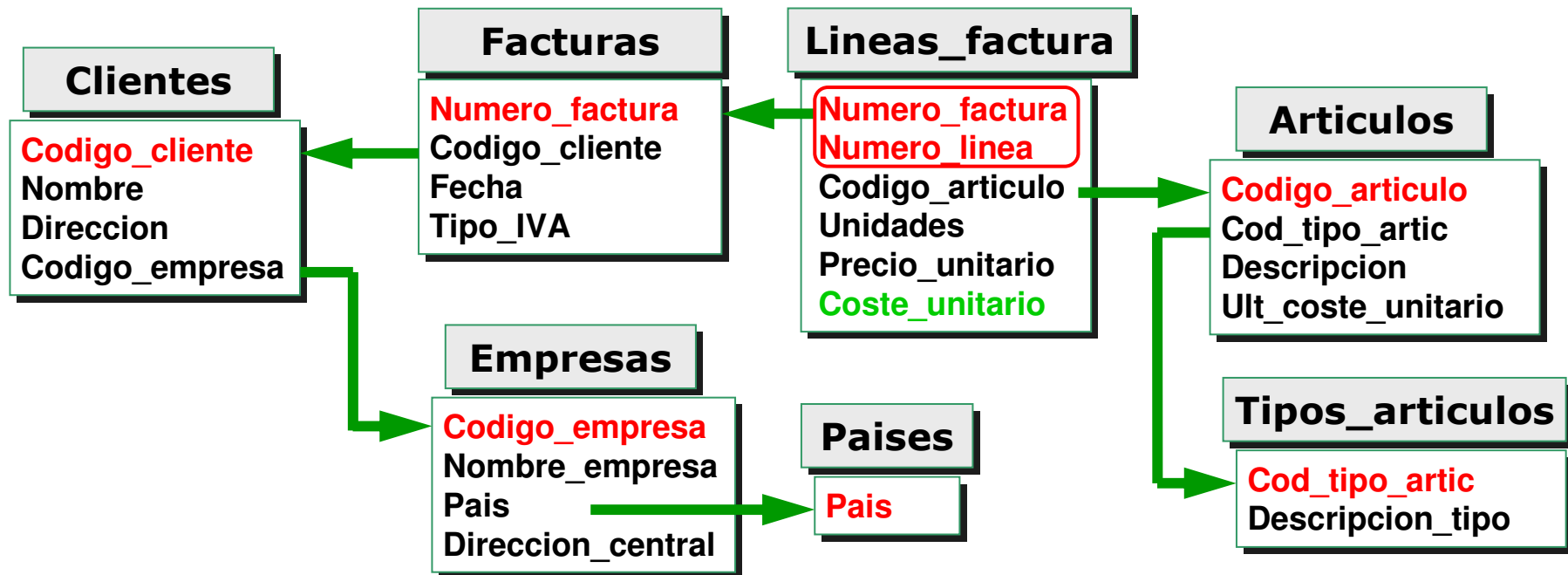
Codigo_cliente	Fecha	Tipo_IVA	Numero_factura
A-741	22-6-2004	16	3421
A-562	22-6-2004	16	3422
A-741	24-6-2004	16	3423

- Unas tablas se refieren a otras mediante vínculos de tipo jerárquico.

- Este vínculo de referencia entre dos tablas se establece mediante columnas con idéntico tipo de dato.

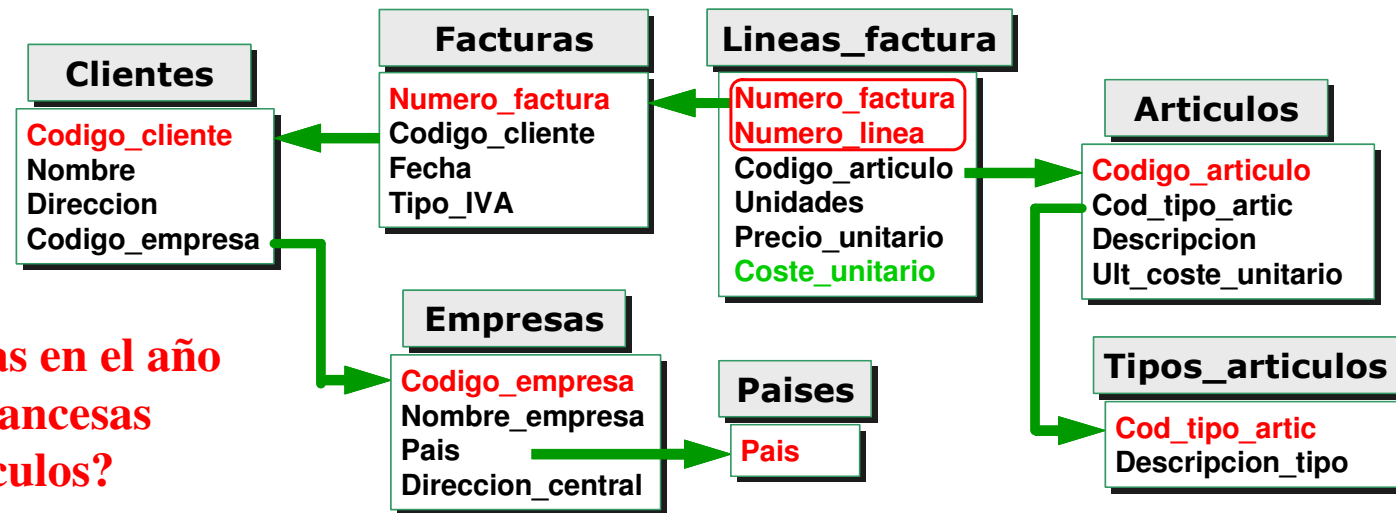
- La referencia de una fila de una tabla a otra de la otra tabla se produce cuando ambas tienen el mismo valor.

## Modelo de datos relacional (Facturación)



¿Beneficio de ventas en el año 2004 a empresas francesas según tipos de artículos?

# Consultas al modelo relacional de facturación

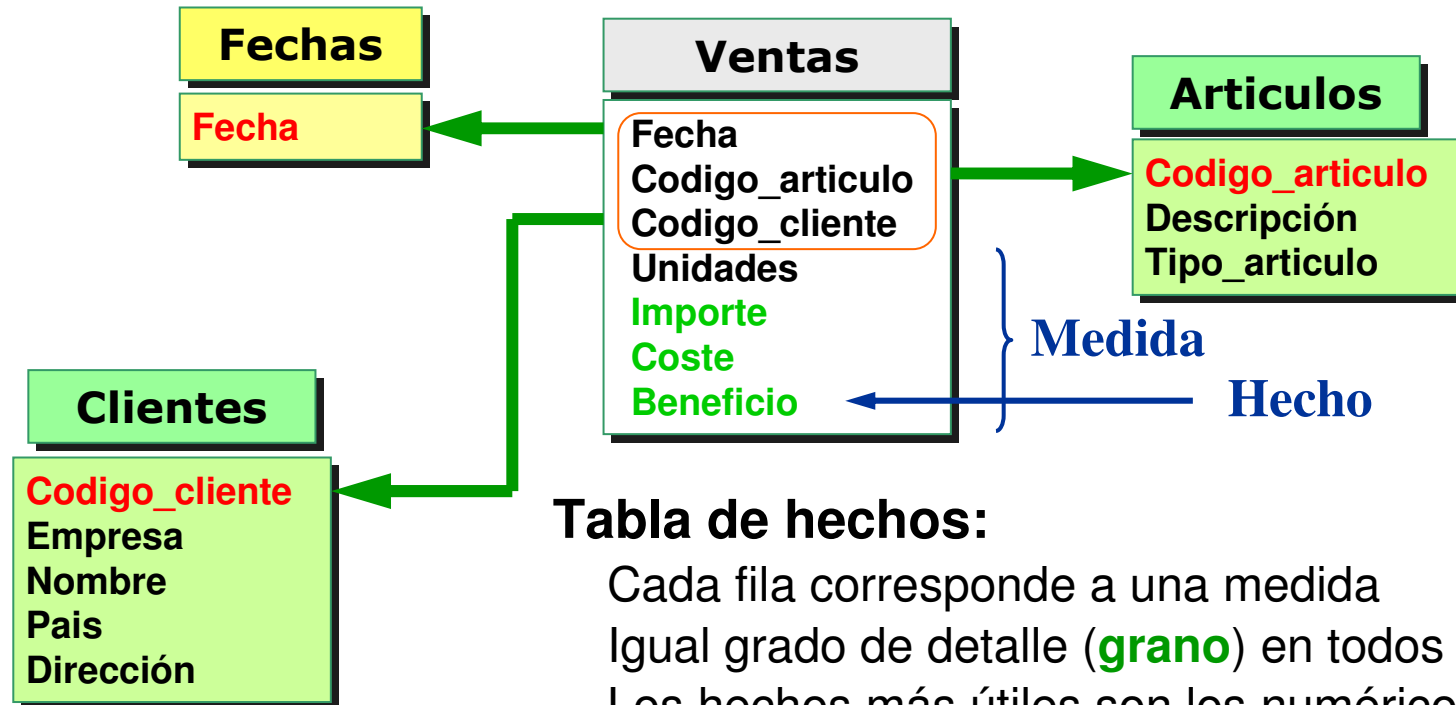


¿Beneficio de ventas en el año  
2004 a empresas francesas  
según tipos de artículos?

```

SELECT SUM(Unidades*(Precio_unitario - Coste_unitario)), Cod_tipo_artic
FROM Empresas INNER JOIN
  (Clientes INNER JOIN
    (Facturas INNER JOIN
      (Lineas_factura INNER JOIN Articulos
        ON Lineas_factura.Codigo_articulo=Articulos.Codigo_articulo)
      ON Facturas.Numero_factura=Lineas_factura.Numero_factura)
    ON Clientes.Codigo_cliente=Factura.Codigo_cliente)
  ON Empresas.Codigo_empresa=Clientes.Código_empresa
WHERE Fecha BETWEEN '1/1/2004' AND '31/12/2004' AND Pais='Francia'
GROUP BY Cod_tipo_artic
  
```

# Esquema de hechos y dimensiones



## Tabla de hechos:

Cada fila corresponde a una medida  
Igual grado de detalle (**grano**) en todos los hechos  
Los hechos más útiles son los numéricos y aditivos

## Tablas de dimensión:

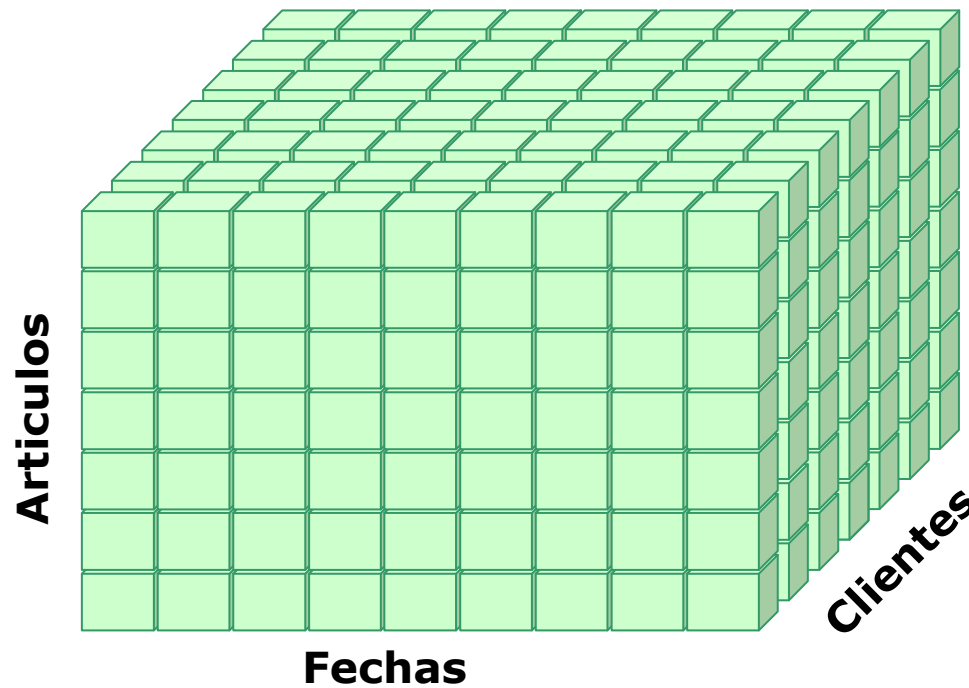
Contienen descriptores textuales  
Son los puntos de entrada en la tabla de hechos

**Transformación de datos**

**Desnormalización**

**Aún no es el modelo dimensional**

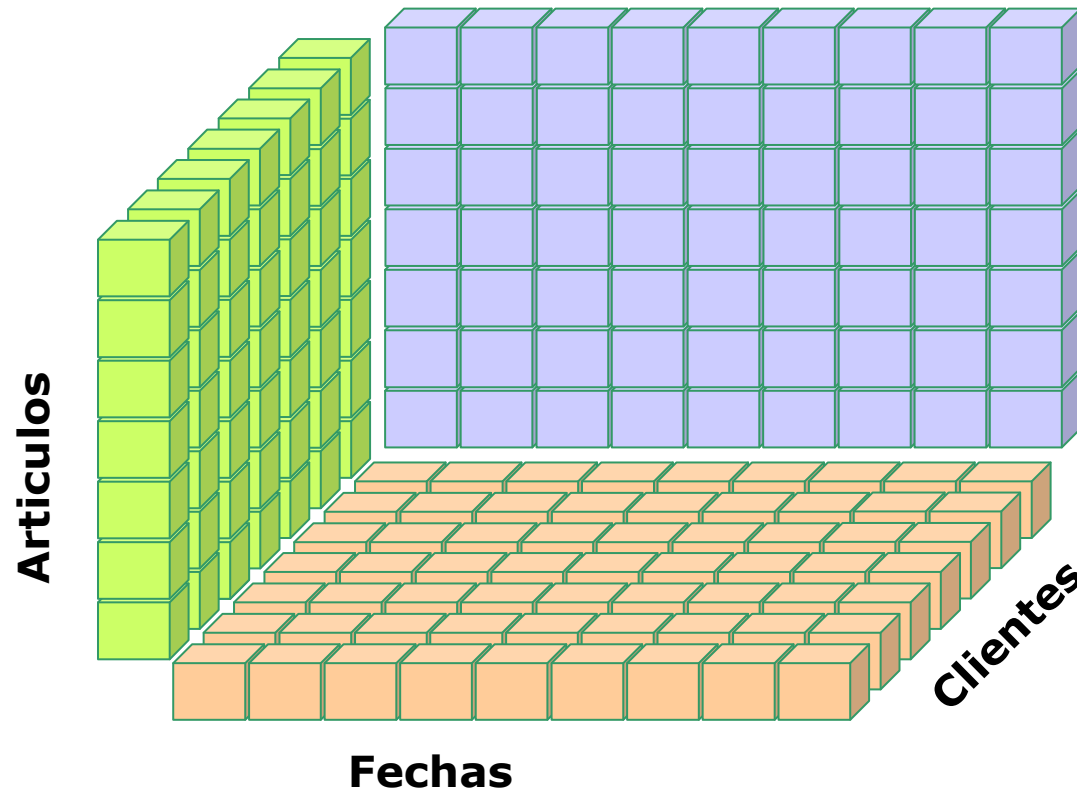
# Cubos e hipercubos



Una estructura de datos como la anterior admite una representación espacial en tres dimensiones.

Cada cubo elemental representa una ocurrencia (fila) en la tabla de hechos.

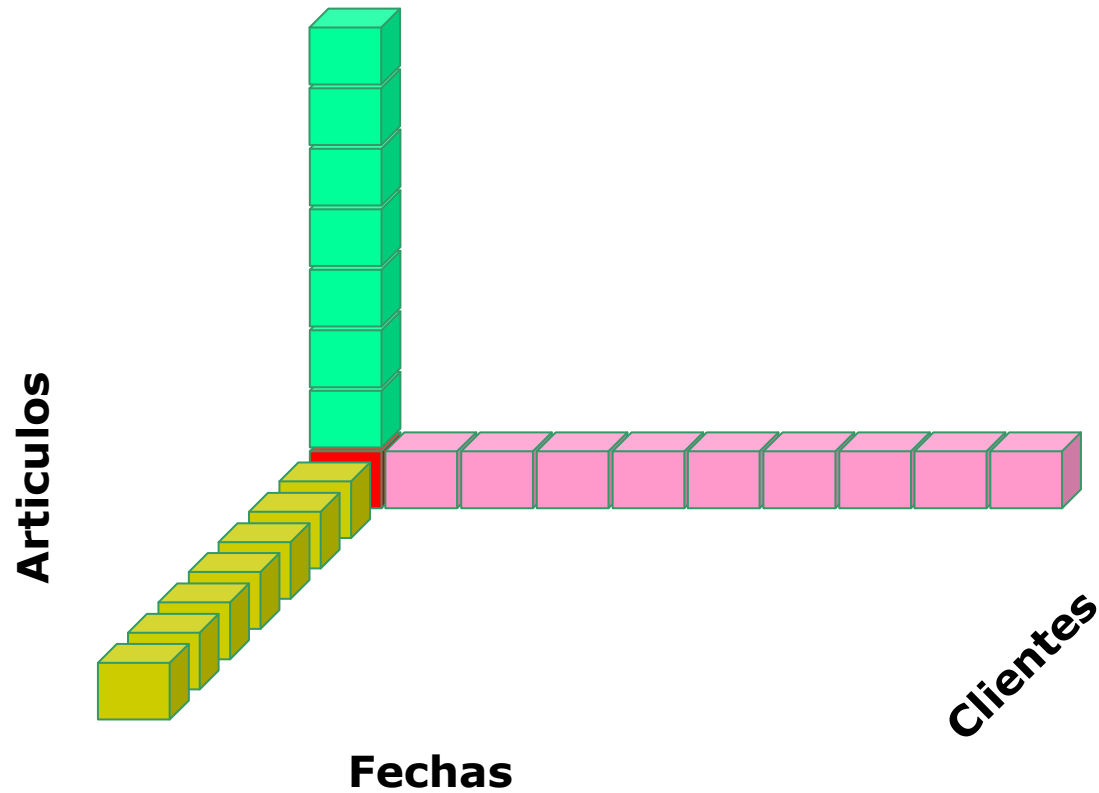
## Acumulados según una dimensión



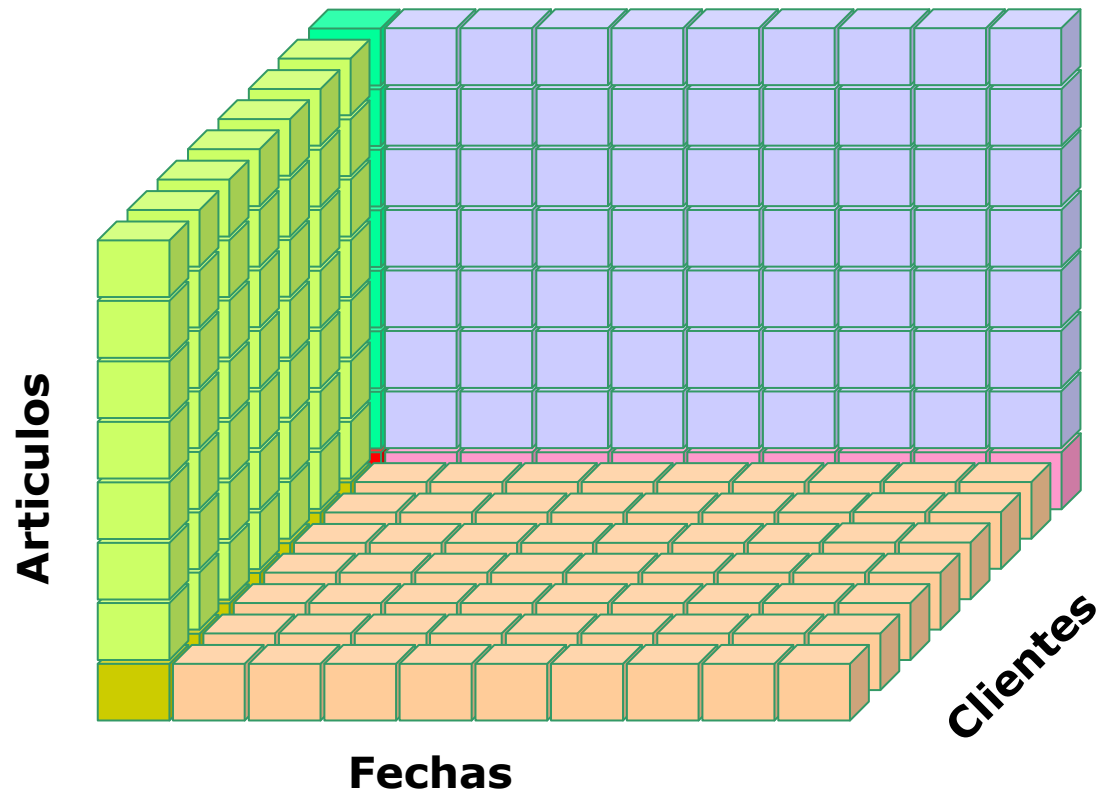
Las medidas (como el beneficio) tiene la propiedad de ser aditivas. Es decir, tiene sentido la suma según todas las dimensiones (beneficios en una fecha, o con un artículo o con relación a un cliente).

Además de almacenar los valores elementales de las medidas, se pueden también guardar los acumulados según las dimensiones.

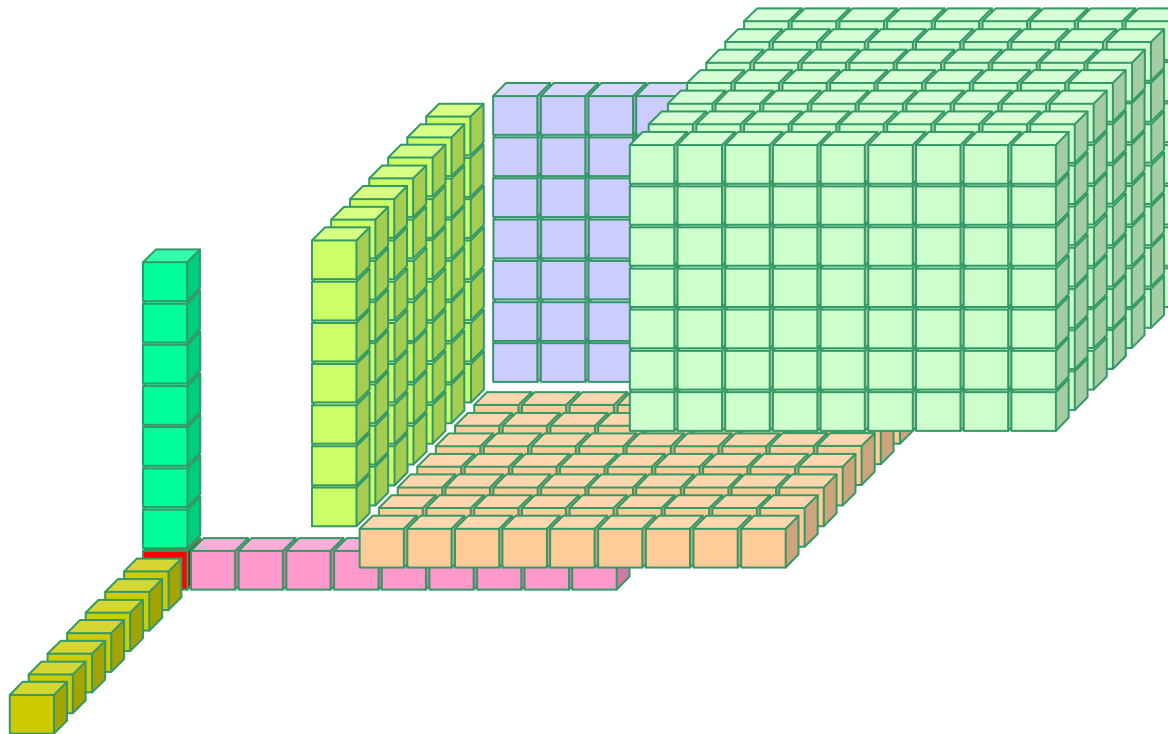
# Acumulados según dos y las tres dimensiones



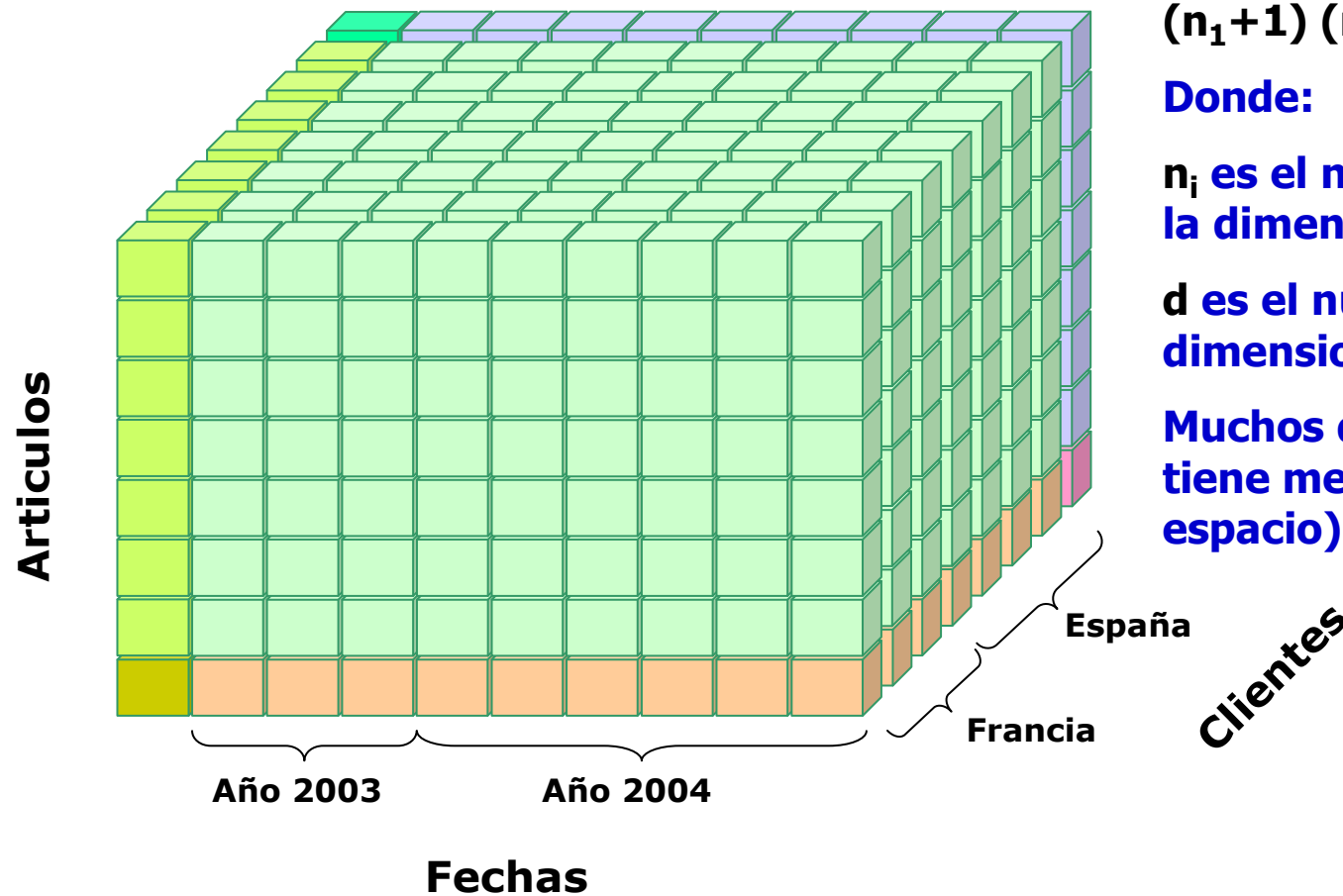
# Acumulados según todas las dimensiones



# Unión del cubo y sus acumulados



# El cubo de medidas con todos sus acumulados



**Tamaño máximo del cubo:**

$$(n_1+1) (n_2+1) \dots (n_d+1)$$

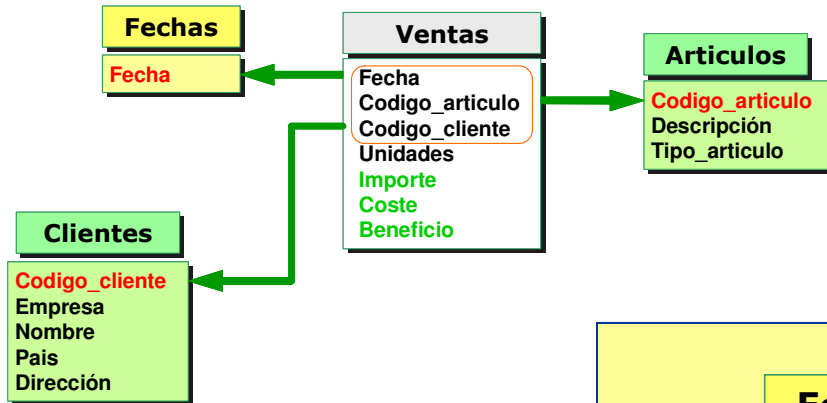
**Donde:**

$n_i$  es el número de filas de la dimensión  $i$  y

$d$  es el número de dimensiones

Muchos de los cubos no tiene medida  $\rightarrow$  (no ocupan espacio)

# Modelo de datos dimensional



## Tablas de dimensión:

Claves de gestión

Atributos [criterios de agregación]

Claves simples (autonómicas)

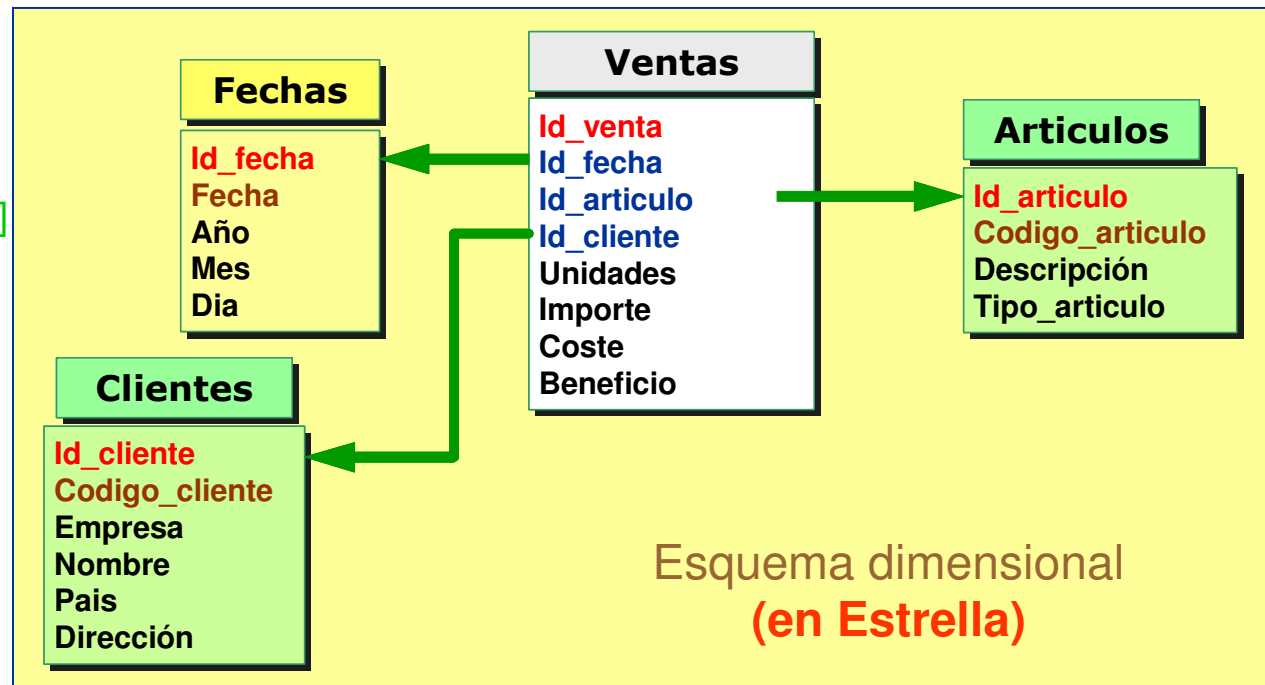
## Tabla de hechos:

Atributos (hechos) [aditividad]

Claves de referencia

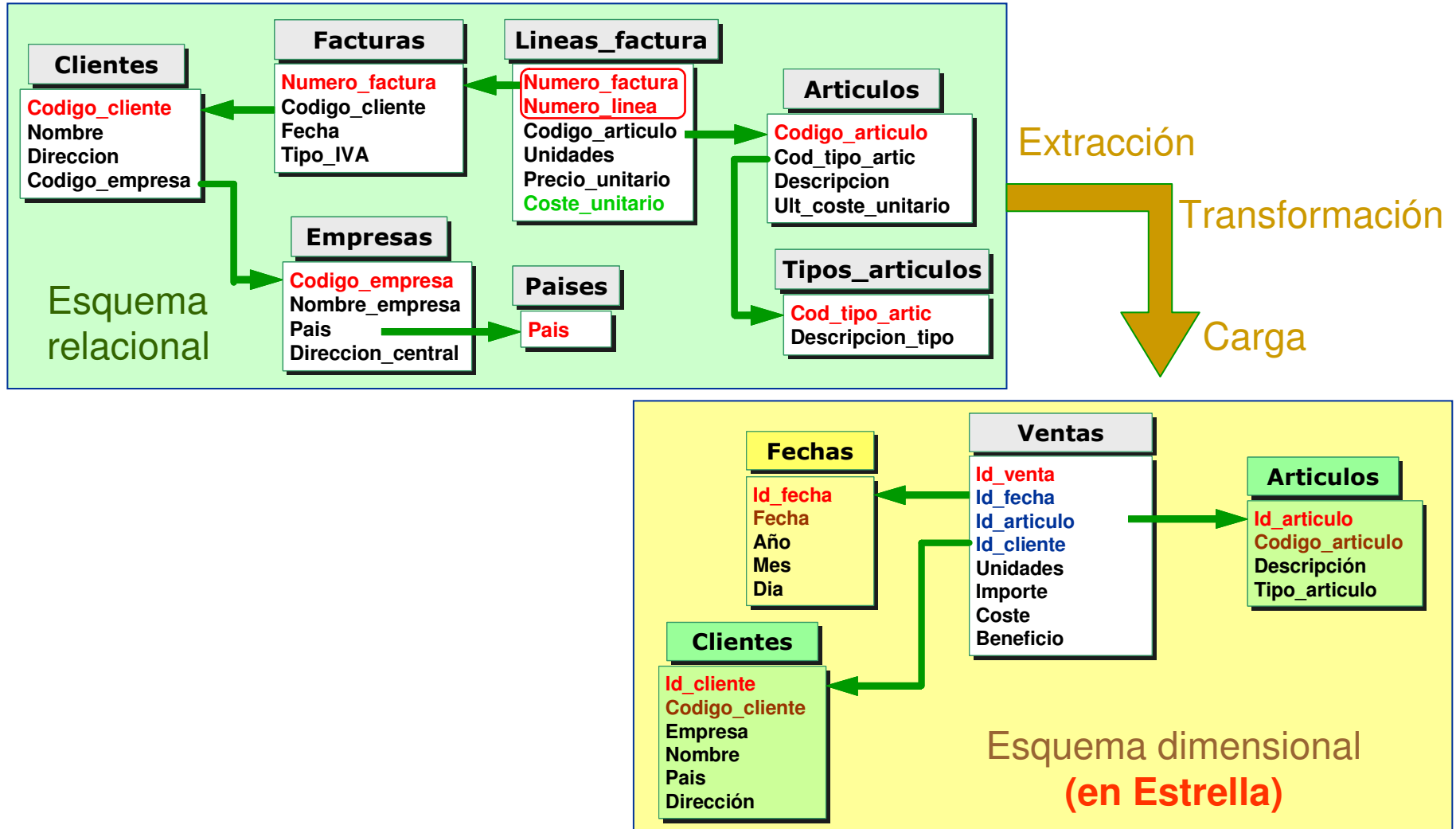
Clave simple (autonómica)

**Claves propias sin significado → Independencia ante cambios de claves de producción**



Esquema dimensional  
(en Estrella)

# Del modelo relacional al dimensional



# Fases en el diseño dimensional

## 1.- Seleccionar los procesos a modelar:

En función de las preguntas estratégicas a responder

## 2.- Decidir el grano:

Preferible información del máximo nivel de detalle

Los datos guardados ya no pueden observarse a un nivel de grano más fino

Normalmente las consultas no pretenden ver el nivel individual

El nivel del grano condiciona la flexibilidad de las consultas admisibles

## 3.- Escoger las dimensiones:

El grano determina la dimensionalidad

Es preciso ajustar las dimensiones al grano

## 4.- Determinar los hechos que deben considerarse:

Buscar la aditividad de los hechos a observar

Porcentajes y proporciones → deben guardarse numerador y denominador

El precio unitario no es aditivo → guardar importe = precio x unidades

## Aditividad de las medidas

---

**Aditividad = tiene sentido sumar según cualquier dimensión**

**Siempre que se pueda, las medidas que se elijan deben ser aditivas**

**Los datos de actividad (como ventas) son generalmente aditivos**

**Los valores de intensidad no suelen ser aditivos**

Ejemplos: existencias de inventario, partidas del presupuesto, ....

Suelen ser aditivos por todas las dimensiones menos por las de tiempo

**Hay otras medidas que no son aditivas según ninguna dimensión**

Ejemplos: temperaturas, precios unitarios, ....

**Las medidas no aditivas pueden agregarse calculando valores medios**

**Hay casos en que interesa valorar la ocurrencia o no de un suceso, su aditividad puede conseguirse mediante los valores 1 ó 0**

Ejemplos: comunicación (SI ó NO), supervisión (SI ó NO)

# Las tablas de dimensiones y sus atributos

## **Siempre debe haber al menos una dimensión temporal**

El grano más adecuado suele ser la fecha (diario)

Cuando también se quiera registrar el instante del día, es mejor otra dimensión

Hay atributos de fecha que no pueden extraerse mediante funciones SQL

Ejemplos: día laborable, vacaciones,...

La dimensión de fecha podría tener más de 20 atributos

## **Con los atributos de las dimensiones se definen las agregaciones**

Ejemplo: Saber las unidades vendidas este año de un artículo en fin de semana

## **Es normal que una dimensión tenga 50 o más atributos descriptivos**

## **Generalmente, una estrella no tiene más de 15 tablas de dimensión**

Un exceso de dimensiones (más de 25) denota que varias no son independientes

En este caso, deben combinarse en dimensiones más simples

## Extensibilidad del esquema

---

### **Nuevos atributos de dimensión:**

- Dan lugar a nuevas columnas en la tabla de dimensión
- Si los nuevos atributos sólo están disponibles a partir de una fecha, en las anteriores deben figurar como no disponibles

### **Nuevas dimensiones:**

- Hay que añadir una nueva clave de referencia en la tabla de hechos
- Y cargar los nuevos valores de la tabla de hechos

### **Nuevas medidas:**

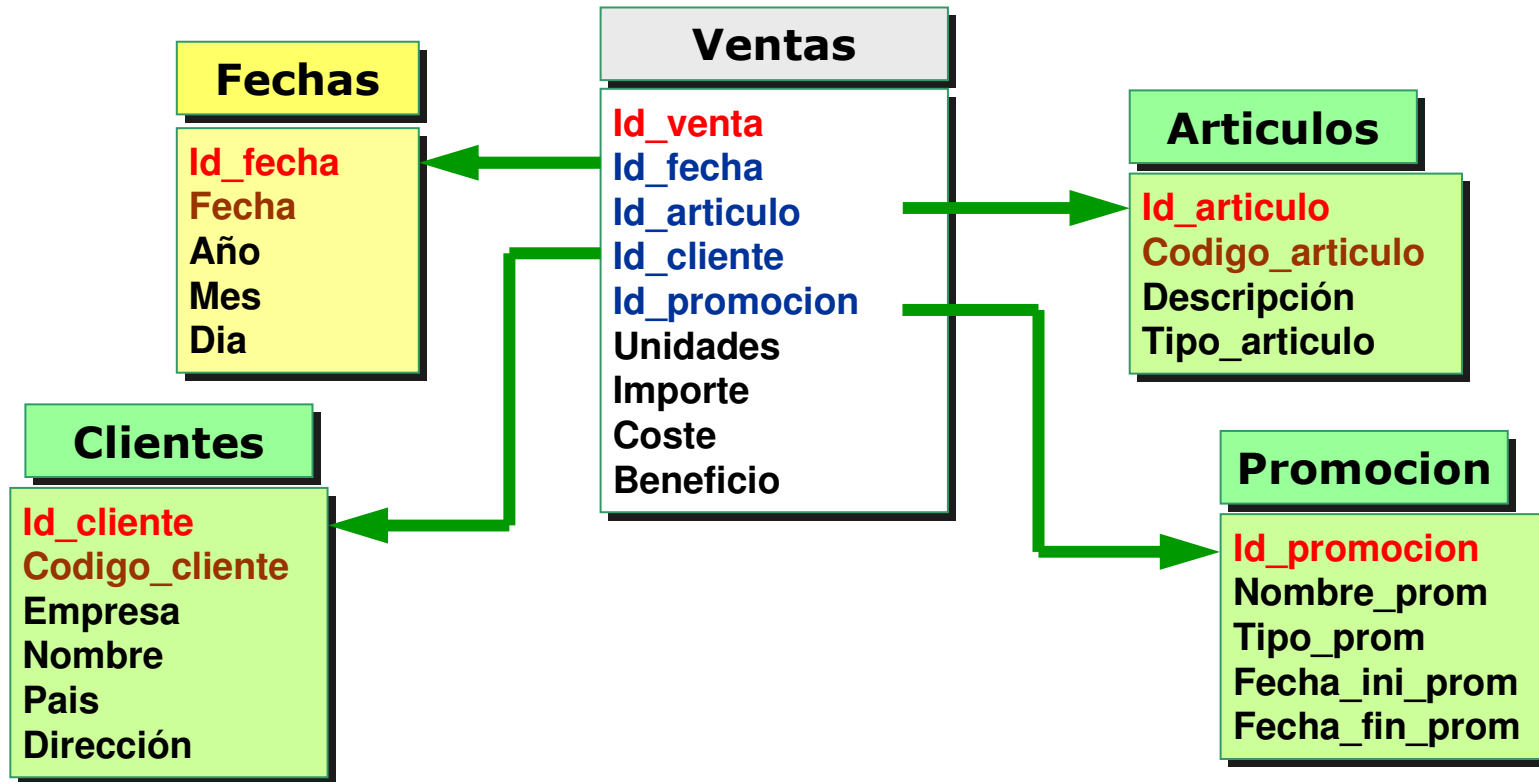
- Nuevas columnas en la tabla de hechos
- Hay que rellenar de valor las filas anteriores al cambio

### **Dimensiones existentes más granulares:**

Hay que eliminar la tabla de hechos y reconstruirla

La tabla de dimensión puede no necesitar su supresión

## Nueva dimensión de promoción



Hay que evitar claves nulas en la tabla de hechos

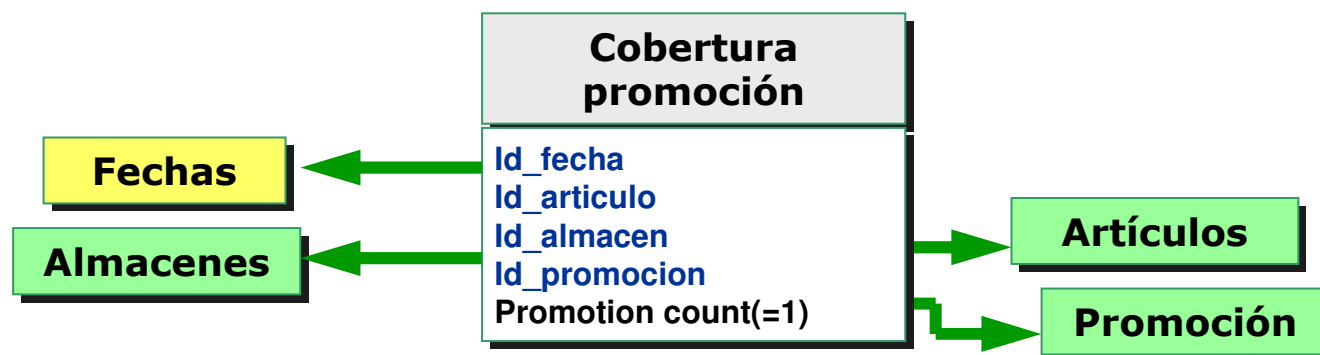
Para ello, deberá haber una fila en la dimensión que indique que no es aplicable

Ejemplo: ventas sin promoción

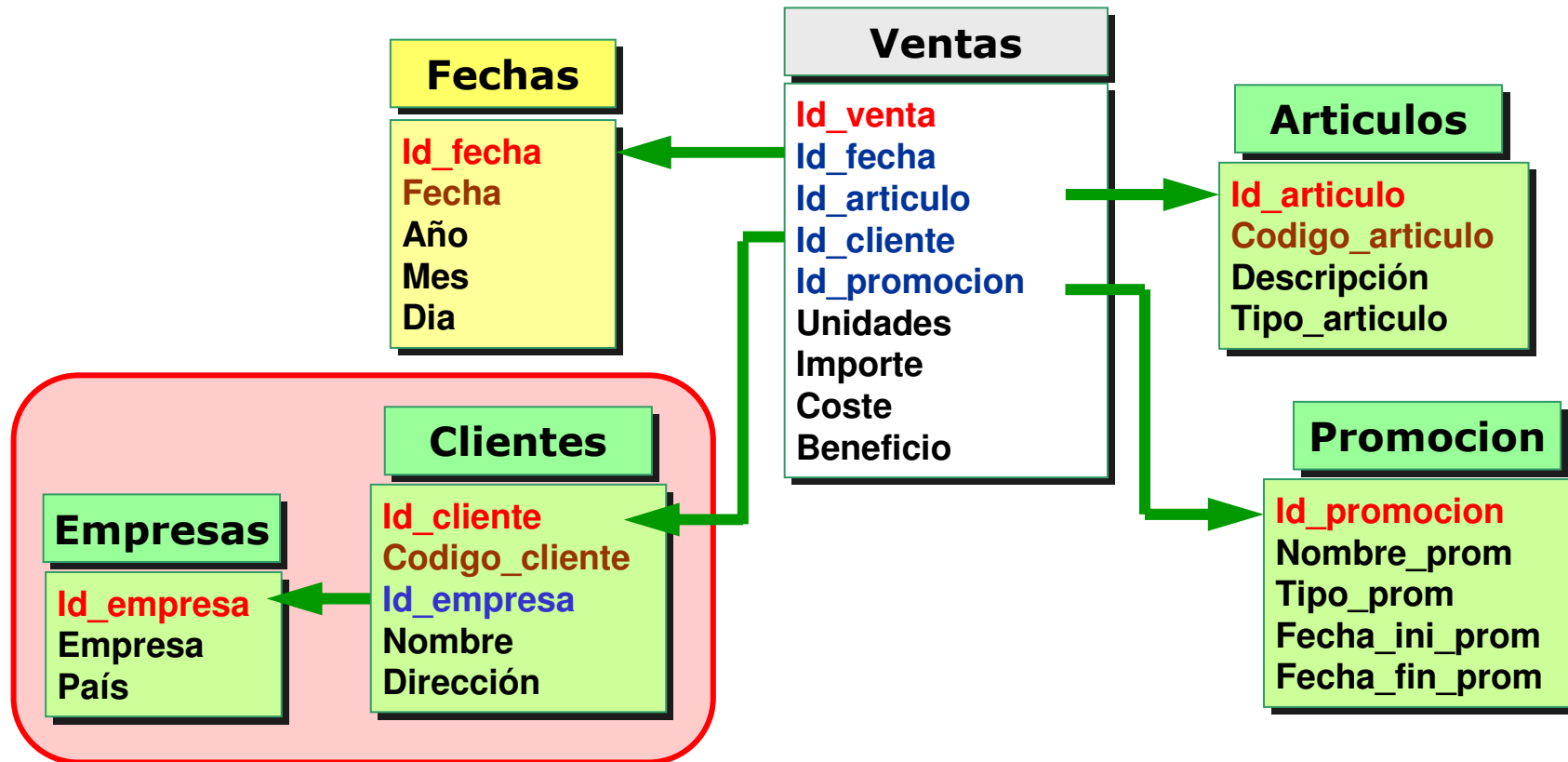
## Tabla de hechos sin hechos

- Tablas de hechos transaccionales no tienen filas para los eventos que no han ocurrido (Ejemplo: productos no vendidos)
- Por una parte.
  - Es bueno: toma ventaja de la escasez de datos
    - Menos datos a almacenar si los eventos son poco frecuentes
  - Es malo: no hay registro de lo que no ocurre
    - Ejemplo: ¿qué productos en promoción no se vendieron?
- Tabla de hechos “Factless”
  - No tiene columnas de hechos numéricas
  - Se utilizan para capturar relaciones entre dimensiones
  - Incluyen una columna de hechos ficticia con valor 1 (siempre)

Ej.: ¿Qué productos estuvieron en promoción en qué almacenes y en qué días?



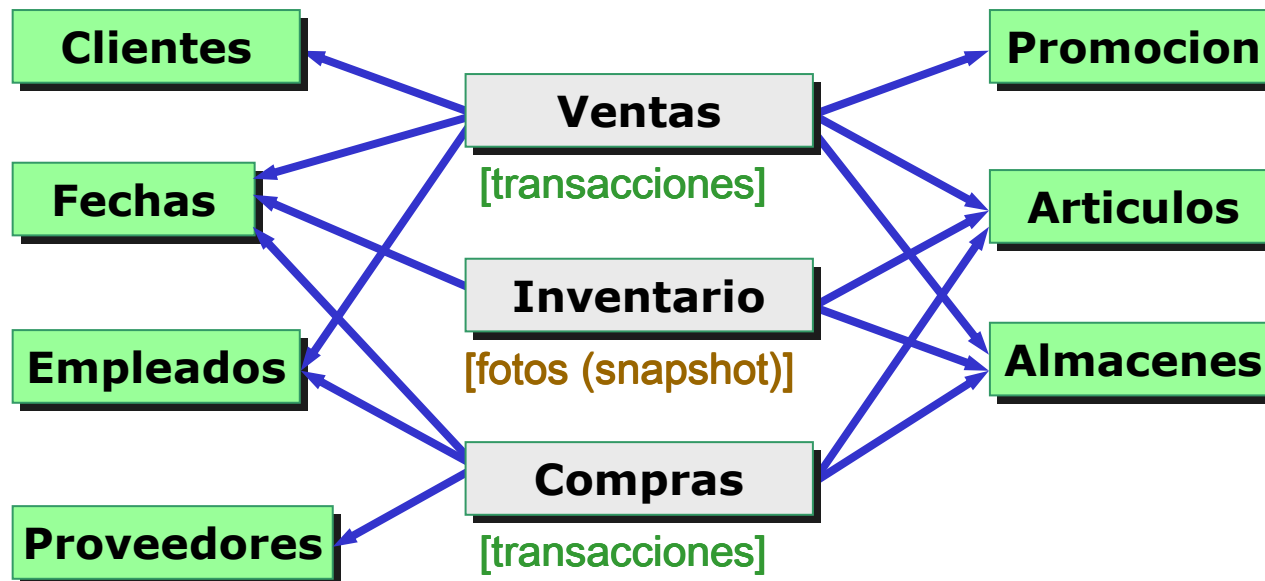
# Normalización de dimensiones (Snowflaking)



Algunas dimensiones se jerarquizan en varias tablas (normalización). Esquema en copo de nieve (**snowflake**)

El nombre de los atributos y valores debería ser único en las dimensiones jerarquizadas

# Tablas de hechos compartiendo dimensiones



## Transaccional vs. Snapshot

- Transaccional
  - Cada fila representa un evento
  - La información se encuentra a nivel más detallado
  
- Snapshot
  - Cada fila representa un instante en el tiempo
  - Generalmente los snapshots se toman a intervalos predefinidos
    - Ejemplos: diarios, semanales
  - Suministran una visión acumulativa
  - Se utilizan para procesos continuos y medidas de intensidad
    - Ejemplos:
      - Balance bancario
      - Inventario
      - Temperaturas de una habitación

## Gestión de inventario (semiaditivo)



$$\text{Margen de retorno de inventario} = \frac{\text{Uds\_vendidas} * (\text{precio\_ult\_venta} - \text{precio\_coste})}{\text{Uds\_entradas} * \text{precio\_ult\_venta}}$$

A decorative L-shaped line consisting of a vertical line on the left and a horizontal line extending to the right, both in a dark grey color. The vertical line is positioned to the left of the text, and the horizontal line is positioned below the text.

## Data warehouse vs data marts

- Data mart:
  - subconjunto de la información de un data warehouse, generalmente de un solo proceso de negocio, que se dirige a un determinado departamento/grupo de usuarios
    - Ejemplos: data mart sobre ventas para dpto comercial y dpto de marketing
- Normalmente contiene la información de un diagrama en estrella por lo que se suelen utilizar como sinónimos, aunque conceptualmente son diferentes
- El grano de un data mart puede ser más grueso que en el data warehouse o del mismo detalle

## Data marts (justificación)

---

- Frecuentemente hay grupos de usuarios que sólo acceden a un subconjunto concreto de los datos
- Descomponer el data warehouse en diferentes data marts suele mejorar el rendimiento de las consultas al reducir el volumen de datos que se recorren para responder
- Los data marts se utilizan para:
  - Segmentar la información en diferentes plataformas hardware (posible portabilidad)
  - Facilitar el acceso de las herramientas de consulta
  - Dividir los datos para controlar mejor los accesos
  - Mejorar los tiempos de respuesta
- Los data marts son necesarios cuando las herramientas de acceso de los usuarios tiene sus propias estructuras internas

## Data marts (otras consideraciones)

---

- Para asegurar la consistencia, los data marts deben ser cargados a partir del data warehouse y no desde las fuentes de datos
- El uso de data marts suele suponer costes adicionales en hardware, software y accesos a la red
- Los procesos de carga de los data marts pueden requerir un tiempo adicional importante
- Los data marts pueden ser necesarios en control de accesos:
  - Los SGBD tradicionales sólo permiten restringir acceso a tablas, no a filas
  - Con un data mart se pueden separar físicamente porciones completas de datos

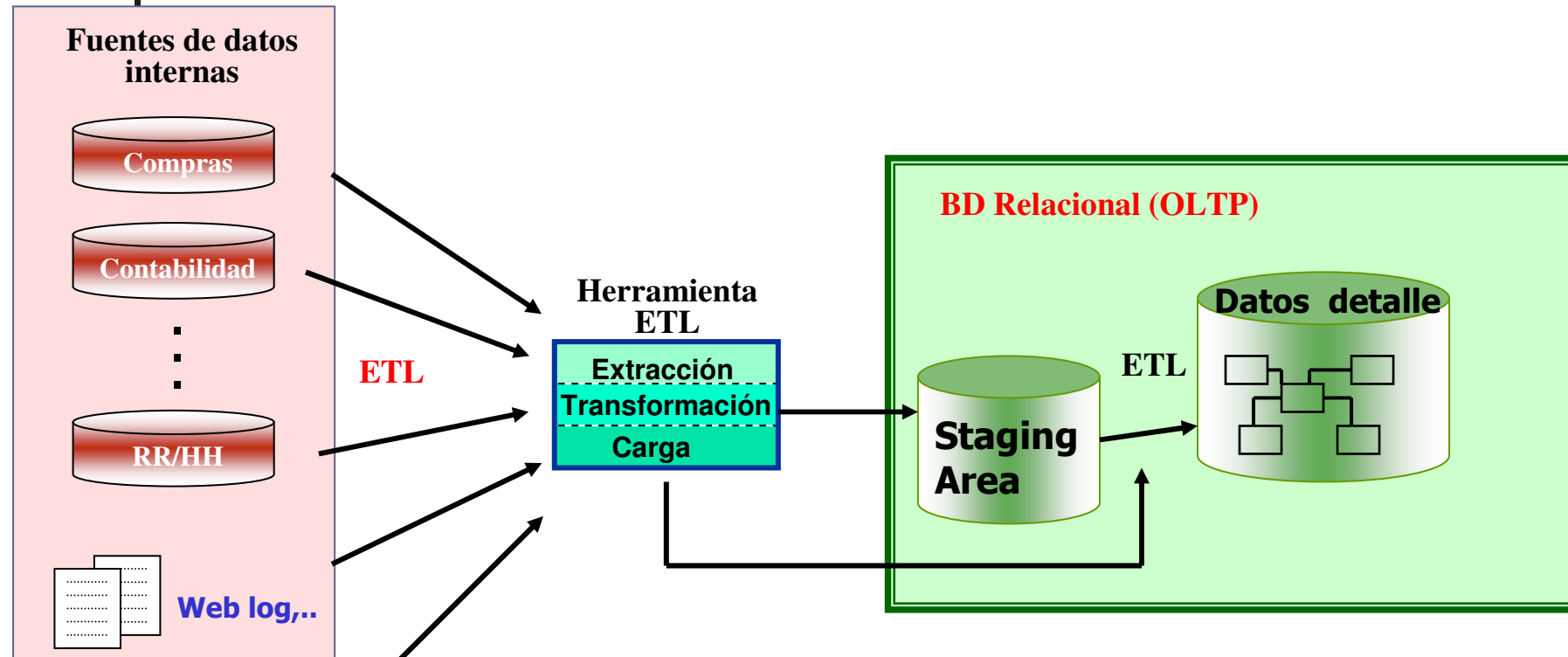
# Procesos de extracción, transformación y carga

---

Marta Zorrilla  
Universidad de Cantabria



# Objetivo



- Extracción: lectura de datos de las diferentes fuentes de datos
- Transformación:
  - Limpiar y transformar los datos añadiéndoles contexto y significado
  - Crear agregaciones y tablas resumen
- Carga: inserción/actualización de las tablas de dimensiones y de hechos.

# Tareas principales de preprocesado

---

- Limpieza
  - Detectar y eliminar errores, rellenar atributos vacíos, resolver inconsistencias
  
- Integración
  - Identificar la mejor fuente de datos para cada registro
  
- Transformación
  - Transformar los datos al contexto del DW: Estandarizar códigos y formatos de representación, definir dominios, usar conversiones y combinaciones para generar nuevos campos, corregir datos ...
  
- Carga de datos
  - Definir procesos de carga y sincronización

## ¿Por qué se han de limpiar los datos?

- Los datos en el mundo real (**data dirty** )
  - **incompletos**: atributos sin valor, falta de atributos interesantes para el contexto o el valor del atributo se tiene agregado
    - ej., ocupación=“ ”, sexo a partir del nombre
  - **con “ruido”**: contienen errores o outliers
    - ej., salario=“-10”
  - **inconsistentes**: contienen discrepancias
    - ej., edad=“42” fecha\_nacimiento=“03/07/1997”
    - ej., era rango “1,2,3”, y ahora “A, B, C”
    - ej., discrepancia entre registros duplicados (ej. Info de personas)

## ¿Por qué nos encontramos “Data Dirty”?

---

- Incompletos porque
  - No es necesario el dato cuando se registra
  - Consideraciones diferentes cuando el dato es registrado y cuando es analizado
  - Problemas humanos/hardware/software
- Incorrectos debido a
  - Error humano o del programa al introducir los datos. Modelos de datos poco robustos.
  - Errores en la transmisión de datos
- Inconsistentes porque
  - Proviene de diferentes fuentes de datos
  - Modelo de datos no normalizados (incumplen las FN)

## La importancia de un buen preprocesado

- Datos sin calidad → resultados de análisis no fiables e inexactos
  - Registros duplicados o valores no asignados llevan a obtener estadísticas incorrectas
- Data warehouse →  
**integración consistente de datos con calidad**
- ETL supone el 70% de la carga de trabajo del desarrollo de un data warehouse

## Características de calidad

---

- Un dato debe ser:
  - Preciso
  - Completo
  - Consistente
  - Creíble
  - Con valor añadido
  - Interpretable
  - Accesible
  - Riguroso en el tiempo

# Limpieza de datos (cleaning)

---

- **Importancia**
  - “Data cleaning is one of the three biggest problems in data warehousing”—Ralph Kimball
  - “Data cleaning is the number one problem in data warehousing”—DCI survey
  
- **Tareas**
  - Identificar y corregir outliers y errores (tipográficos, de dominio,...)
  - Corregir datos inconsistentes (fecha de nacimiento > hoy)
  - Rellenar valores perdidos (missing data)
  
- Realizar la corrección, si es posible, en cada fuente de datos de origen. Si no en la tarea de transformación.

## Missing Data (Datos perdidos)

---

- Los datos puede que no estén siempre disponibles debido a:
  - No se registra la historia o los cambios de los datos
  - Mal funcionamiento del equipo
  - Los datos nunca fueron rellenados o no existía en aquel momento
  - O se eliminaron por ser inconsistentes con otra información registrada
- ¿qué soluciones?
  - Etiquetarle con una constante global “desconocido”
  - Asignarle la media del conjunto total o de los de su clase
  - A veces, éstos deben ser inferidos por medio de una fórmula Bayesiana o un árbol de decisión

## Integración de datos

- El problema de la redundancia:
  - Aparece cuando se integran múltiples bases de datos
    - *Identificación de objetos*: el mismo atributo u objeto puede tener diferentes nombres en cada base de datos, e incluso, el dominio puede variar
    - *Datos derivados*: un atributo puede aparecer como dato derivado en otra tabla ( ej. Renta anual)
- Identificar la fuente de datos más fiable para cada dato

Una integración cuidadosa ayuda a evitar/reducir las redundancias y las inconsistencias

# Transformación

---

- Estandarizar códigos y formatos de representación
  - Pasar la información EBCDIC a ASCII o Unicode
  - Convertir números cardinales a ordinales
  - Separar fecha y hora
  - Poner textos descriptivos
  - Unificar códigos ( hombre-mujer, varón-hembra, 0-1).
  - Unificar estándares: unidades de medida, de tiempo, moneda, etc.
  
- Corregir (si no se pudo hacer en el origen)
  - errores tipográficos
  - datos que no tienen sentido (fecha de nacimiento > hoy)
  - resolver conflictos de dominio
  - aclarar datos ambiguos
  - asignar valor a datos nulos (missing data)

## Transformación (y 2)

---

- Política de resolución de claves (cuando proceden datos de varias fuentes)
  - Uso de metadata
- Eliminar datos/registros duplicados
  - Situaciones complejas → análisis de correlaciones o apoyarse en funciones “fuzzy” (fuzzy lookup, fuzzy grouping)
- Usar conversiones y combinaciones para generar nuevos campos
  - Calculados: importe neto, beneficio,...
  - Agregados: cuentas, promedios, etc...
  - Derivados: la nota numérica y textual

- Inicial
  - No suele plantear problemas.
  - Puede requerir bastante tiempo.
  - Realizar en horas de baja carga de los sistemas.
  
- Sincronización o actualización incremental
  - Es compleja de diseñar. Debe ser eficiente computacionalmente → “Staging Area”
  
  - ¿cómo reconocer los cambios?
    - Utilizar la información de última modificación (fecha o campo rowversion) si los datos en el entorno operacional disponen de ella. No muy frecuente.
    - Mantener una imagen del “antes” y otra del “después” de la extracción de la información, comparar ambas y detectar los cambios. Esta es una aproximación compleja y requiere muchos recursos.
    - Utilizar tablas de auditoría: dimension table staging y fact table staging.
  
  - Se ha de prestar atención a la variable temporal si los datos proceden de distintas fuentes (p. ej. datos de bolsa Madrid y Nueva York).



# Cubos OLAP

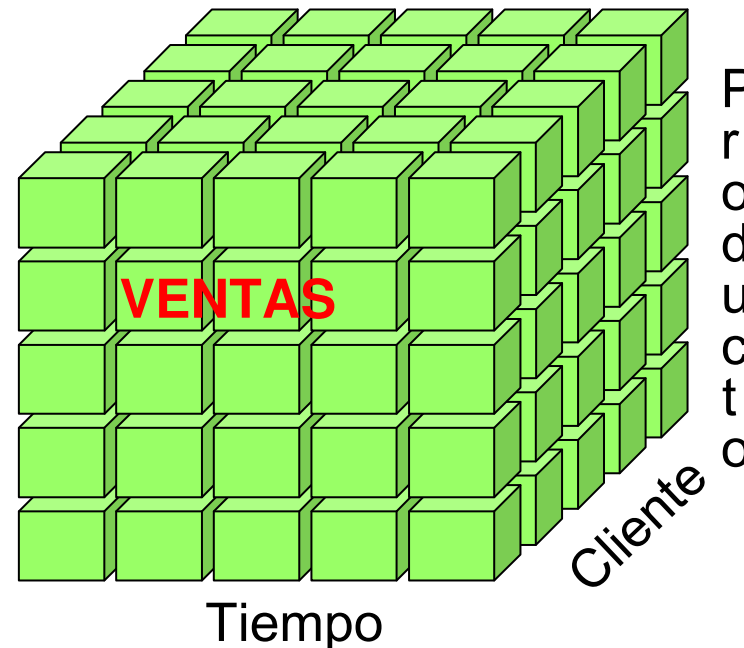
## Definición cubos OLAP

Estructura de almacenamiento que permite realizar diferentes combinaciones de datos para visualizar los resultados de una organización (indicadores) hasta un determinado grado de detalle, permitiendo navegar por sus dimensiones y analizar sus datos desde distintos puntos de vista.

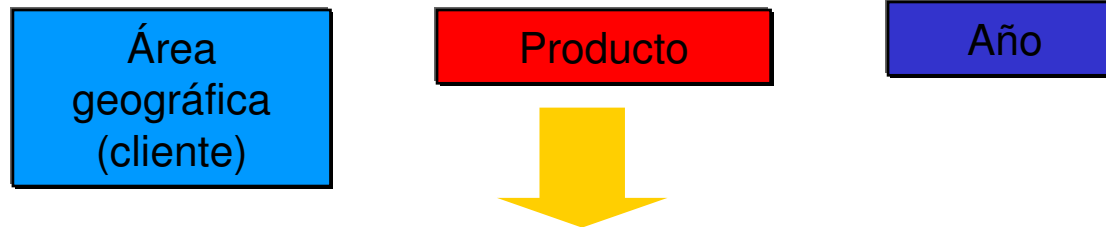
## Introducción a los cubos

- Un cubo es un subconjunto de datos de un DW que es almacenado en una estructura multidimensional. El cubo contiene los valores agregados de todos los niveles de todas las dimensiones.
- Presenta los datos de interés para el usuario.

- Importe total y unidades vendidas durante este año de los productos del departamento Bebidas, por trimestre y por categoría
- ¿Cómo varió la venta de los productos cárnicos durante el tercer trimestre del 2002 en relación al segundo trimestre?
- Beneficio neto por producto, área geográfica y año



# Informe OLAP



Oracle Discoverer - [Vidstr31.dis]

File Edit View Sheet Format Tools Graph Window Help

Min Max % + - x ÷ = ≠ < ≤ > ≥ ∇

Arial 10 B U

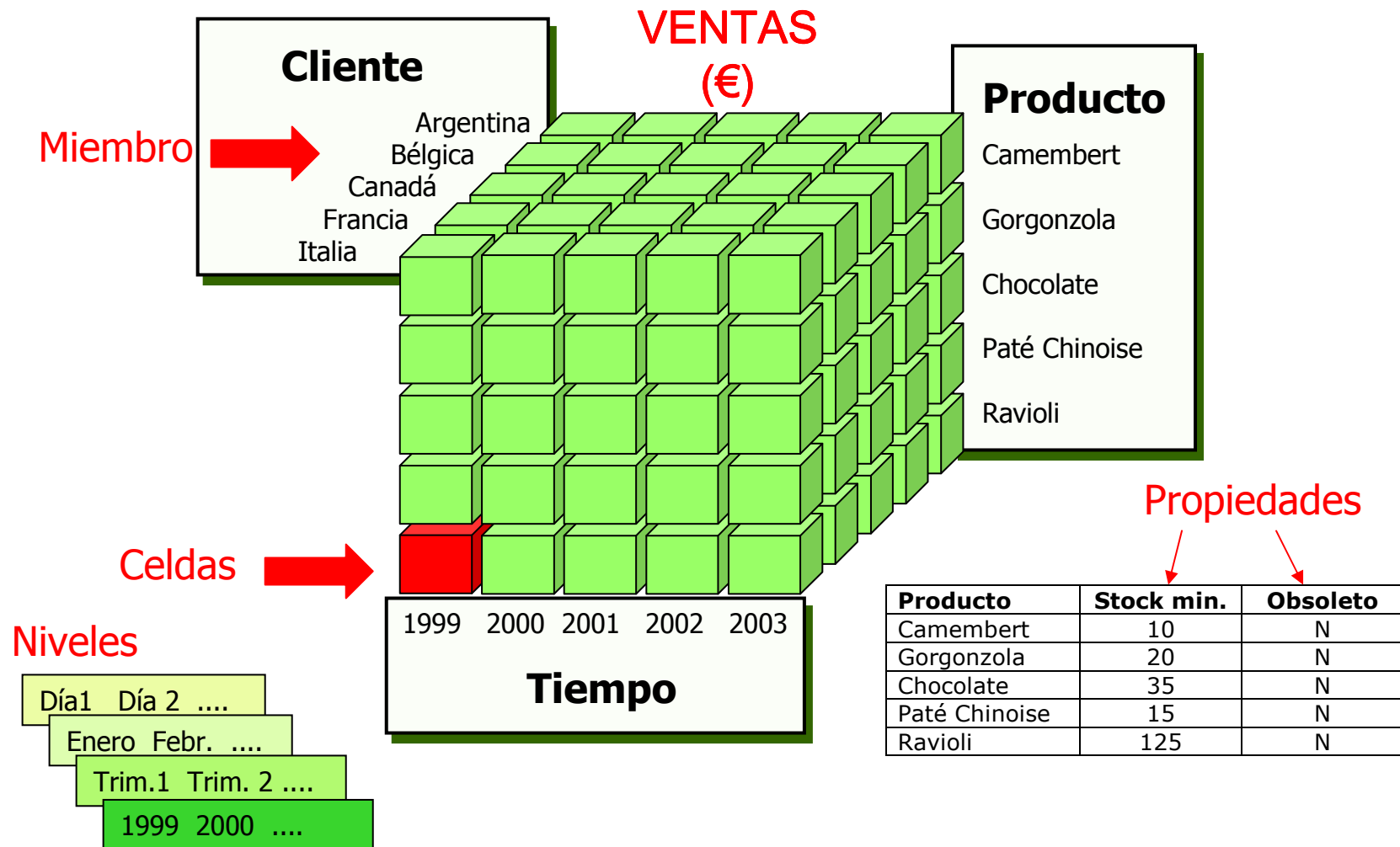
**Análisis de Alquileres y Ventas de video  
15-NOV-01**

Page Items: Departamento: Beverage

	Beneficios			región en el	Porcentaje de cada año en la región		
	1995	1996	Total (1995-1996)		1995	1996	
> Central	\$523	\$742	\$1,265	20%	31%	41%	59%
> East	\$1,229	\$1,160	\$2,391	47%	48%	51%	49%
> West	\$872	\$519	\$1,391	33%	21%	63%	37%
Total	\$2,624	\$2,421	\$5,047				

T-Beneficios (región-dpto-año) TC-Beneficios (Dpto-Region-fecha) mati Sheet 1 Sheet 3 Sheet 4 Sheet 5

# Componentes de un cubo

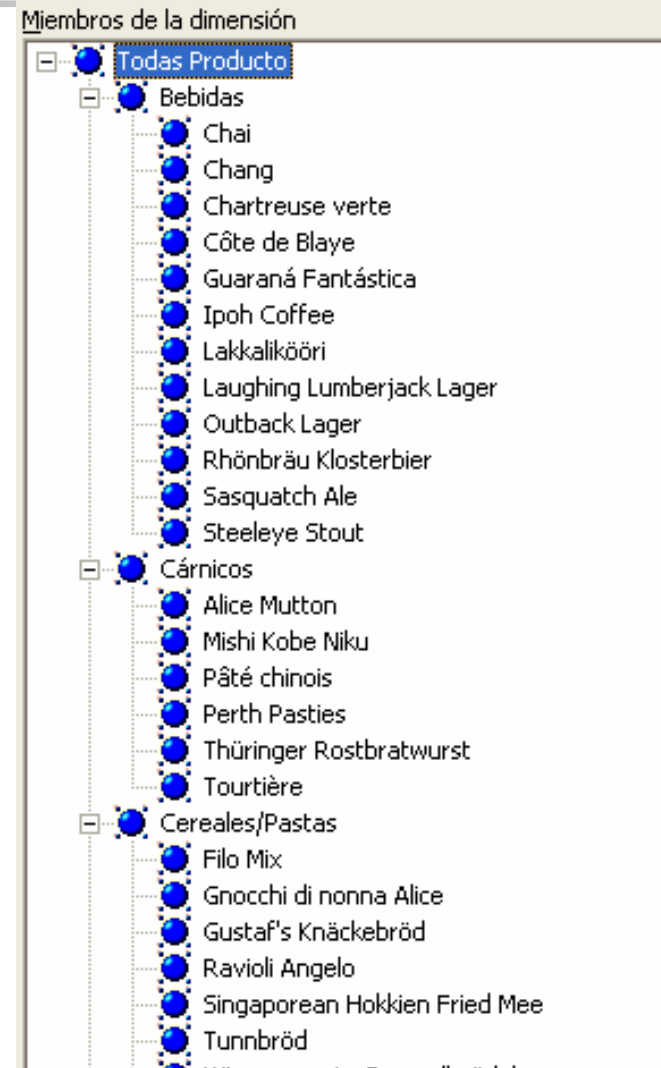


## Definir dimensiones

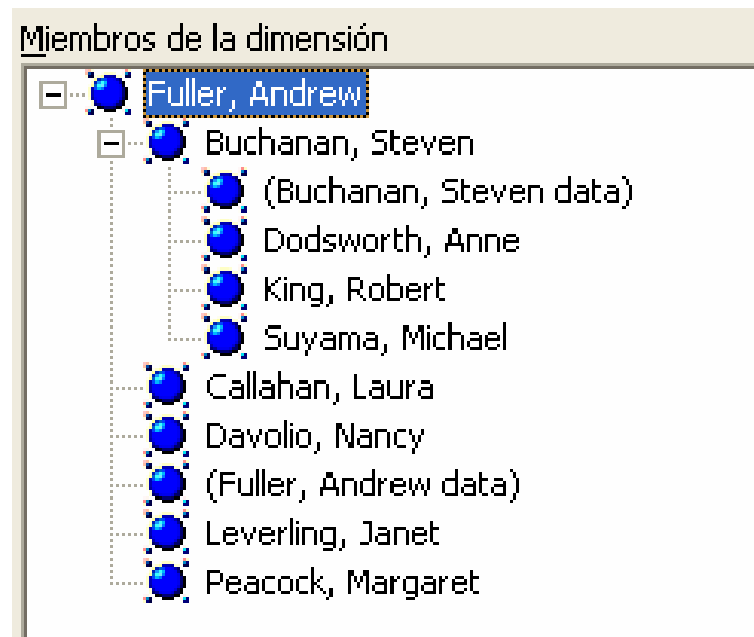
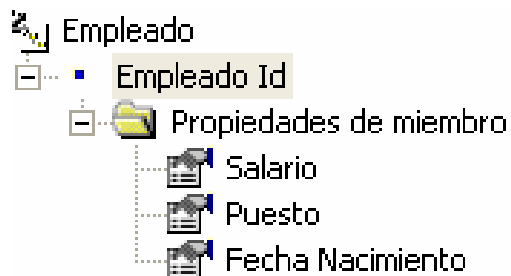
---

- Identificar las columnas de las tablas que participan en la dimensión
- Definir los niveles y las propiedades
- Pueden ser:
  - balanceada (producto)
  - no balanceada (empleado)
  - desigual (el padre de un miembro no se encuentra en el nivel que está por encima inmediatamente de éste, cliente)

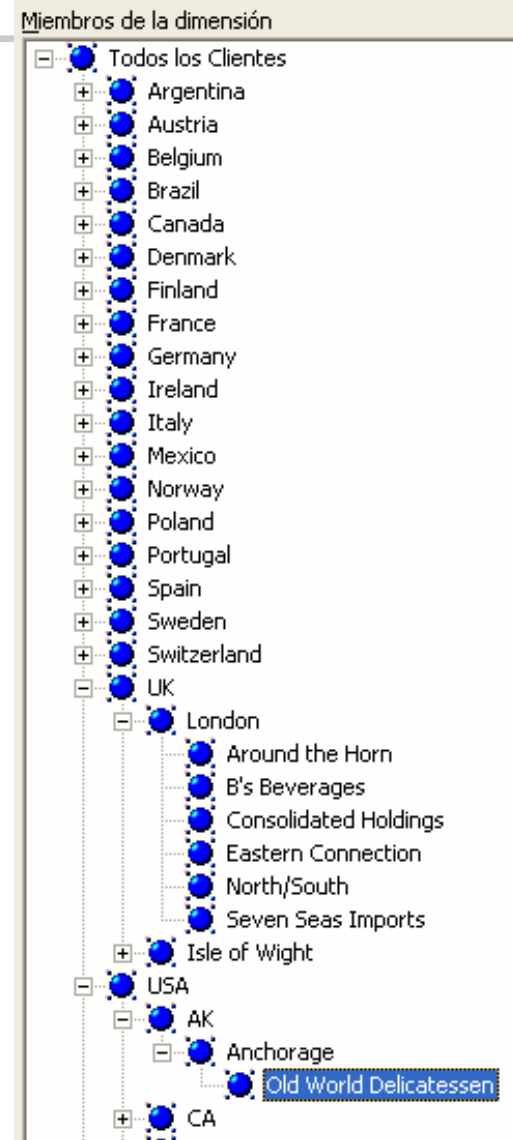
# Dimensión equilibrada



# Dimensión no equilibrada



# Dimensión desigual



## Definir cubos

- Identificar la tabla de hechos
  - Elegir las medidas
    - Aditivas y no aditivas
    - Métricas calculadas
  - Nivel de detalle (filtro)
  
- Métricas calculadas:
  - La diferencia entre **medida derivada** y **miembro calculado** es cuándo se realiza el cálculo.
  - Una medida derivada se calcula antes que las agregaciones sean creadas y los valores son almacenados en el cubo.

$$\text{Beneficio} = \text{venta} - \text{coste}$$

- Los miembros calculados no son almacenados en el cubo y se calculan las agregaciones antes que se efectúe la operación.

$$\text{Margen ud. Prod.} = \text{suma}(\text{beneficio\_linea}) / \text{suma}(\text{unidades\_linea})$$

# Almacenamiento de los cubos

## Opciones de almacenamiento

Rendimiento → **MOLAP**  
Base Datos Multidimensional

Los datos que subyacen en los hipercubos son almacenados junto con las agregaciones en una estructura multidimensional

Capacidad → **ROLAP**  
Base Datos Relacional

Los datos que subyacen en los hipercubos son almacenados junto con las agregaciones en una estructura relacional

**HOLAP**  
Sistema híbrido

Los datos que subyacen en los hipercubos son almacenados en una estructura relacional y las agregaciones en una estructura multidimensional

**DOLAP**  
Desktop OLAP

Instalación MOLAP en un equipo cliente

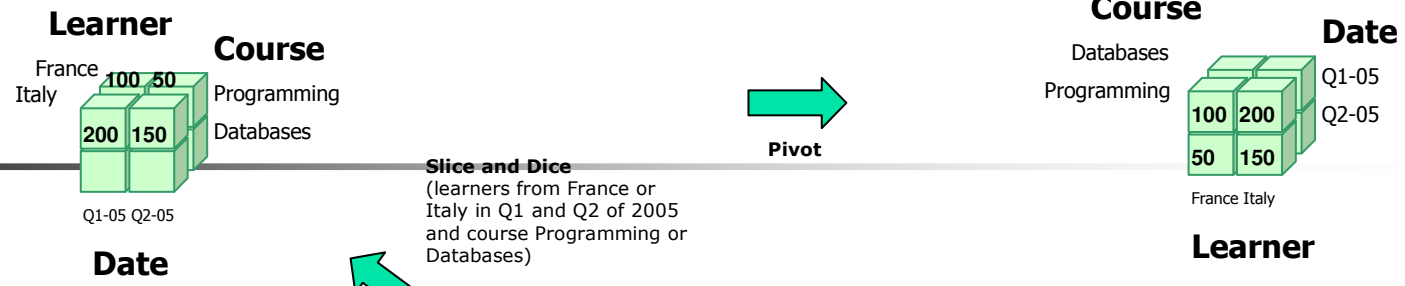
## Herramientas OLAP

---

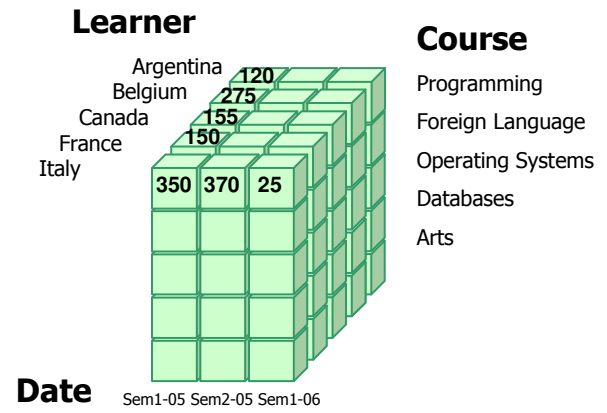
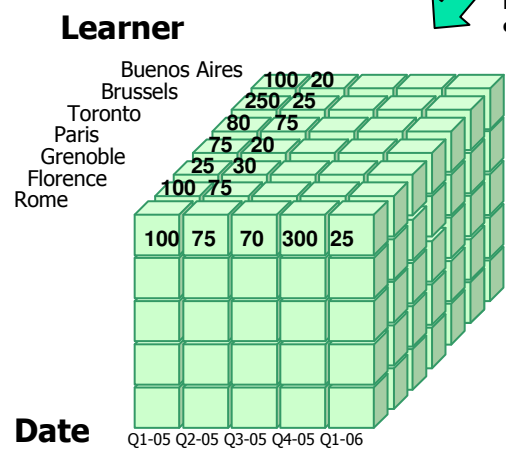
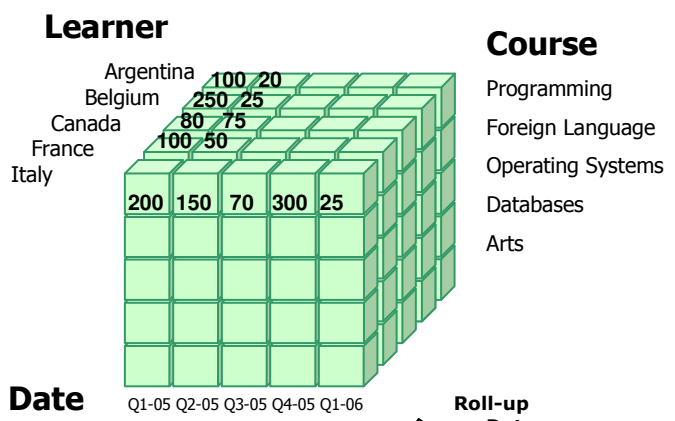
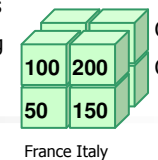
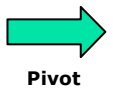
- Lo interesante no es poder realizar consultas que, en cierto modo, se pueden hacer con selecciones, proyecciones, concatenaciones y agrupamientos tradicionales.
- Lo realmente interesante de las herramientas OLAP son sus **operadores de refinamiento o manipulación de consultas.**
  - DRILL
  - ROLL
  - SLICE & DICE
  - PIVOT

# Análisis de datos: operaciones en cubos OLAP

- **Roll up (drill-up):** resumir los datos
  - Subir en la jerarquía o reducir las dimensiones
- **Drill down (roll down):** el contrario del anterior
  - bajar en la jerarquía o introducir nuevas dimensiones
- **Slice and dice:**
  - Selección y proyección
- **Pivot (rotar):**
  - Reorientar el cubo
- **Drill:**
  - Se utilizan las coordenadas dimensionales especificadas por un usuario para una celda en un cubo para moverse a otro cubo a ver información relacionada
    - *drill across:* implica utilizar más de una tabla de hechos
    - *drill through:* Ir desde el nivel de máximo detalle del cubo a sus tablas relacionales (utilizando SQL)



**Slice and Dice**  
(learners from France or Italy in Q1 and Q2 of 2005 and course Programming or Databases)



# Herramientas OLAP

---

## Las herramientas de OLAP se caracterizan por:

- ✓ ofrecer una visión multidimensional de los datos (matricial).
- ✓ no imponer restricciones sobre el número de dimensiones.
- ✓ ofrecer simetría para las dimensiones.
- ✓ permitir definir de forma flexible (sin limitaciones) sobre las dimensiones: restricciones, agregaciones y jerarquías entre ellas.
- ✓ ofrecer operadores intuitivos de manipulación: *drill-down*, *roll-up*, *slice-and-dice*, *pivot*.
- ✓ ser transparentes al tipo de tecnología que soporta el almacén de datos (ROLAP o MOLAP).

# Soporte para BD dimensionales en SQL:1999 Y SQL:2003

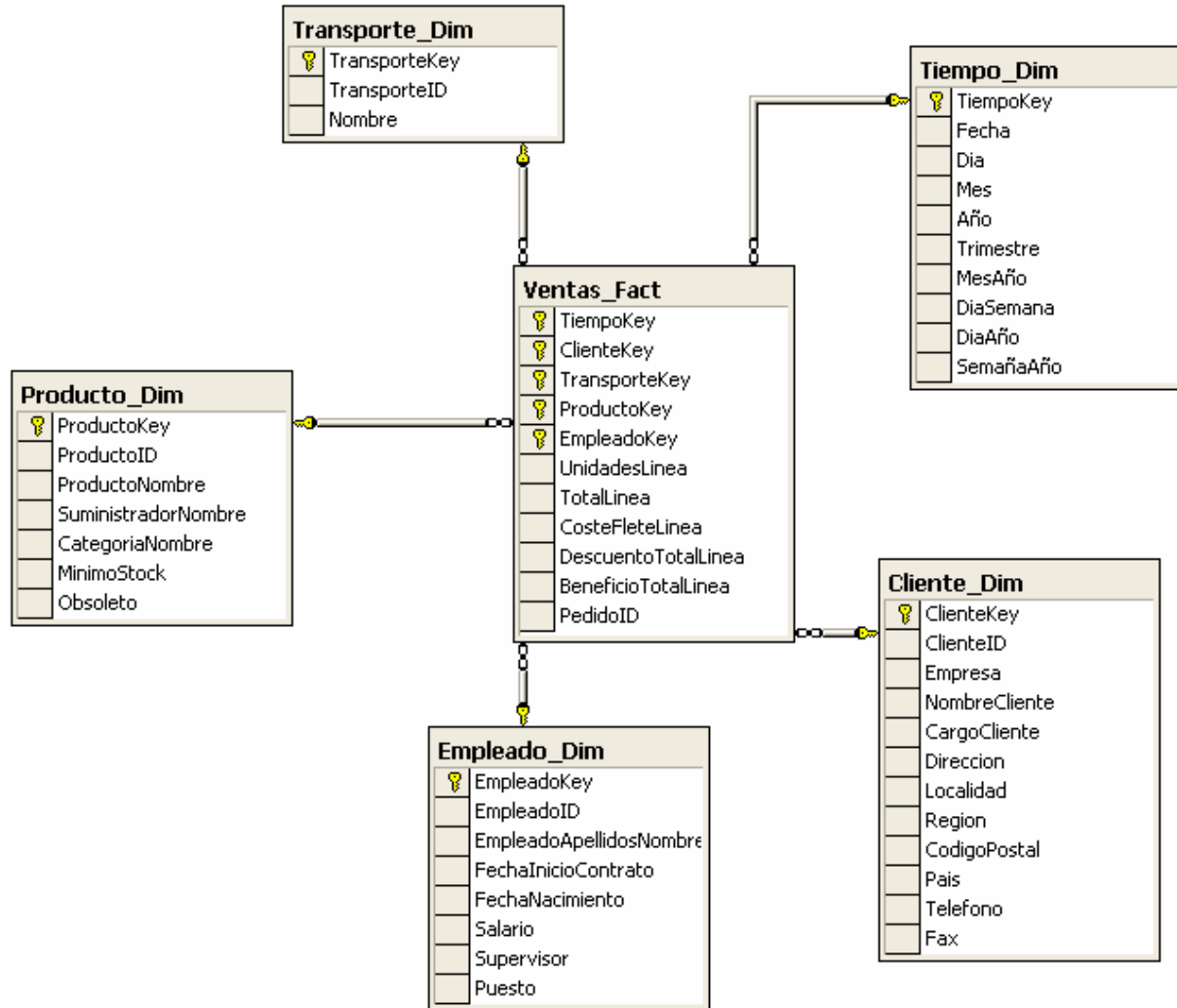
---

## SQL99 OLAP SQL extensions

GROUP BY <grouping specification>

<grouping specification> ::=  
<grouping column reference list> |  
ROLLUP (<grouping column reference list>) |  
CUBE (<grouping column reference list>) |  
GROUPING SETS (<grouping set list>) | ()

# BD ejemplo



## Agrupamiento con operador ROLLUP

El operador **ROLLUP** mantiene los grupos formados por GROUP BY y además añade los superagregados de la primera columna del group by.

- Obtener el nº de unidades pedidas por categoría, país y año, con los subtotales por categoría.

```
SELECT   CategoriaNombre, Pais, Año, SUM(UnidadesLinea) AS suma
FROM     Ventas_Fact INNER JOIN Producto_Dim ON
Ventas_Fact.ProductoKey = Producto_Dim.ProductoKey INNER JOIN
Cliente_Dim ON Ventas_Fact.ClienteKey = Cliente_Dim.ClienteKey
INNER JOIN Tiempo_Dim ON Ventas_Fact.TiempoKey =
Tiempo_Dim.TiempoKey
GROUP BY ROLLUP (CategoriaNombre, Pais, Año)
```

## Agrupamiento con operador CUBE

El operador **CUBE** mantiene los grupos formados por GROUP BY y además añade los superagregados para cada columna.

- Obtener el nº de unidades pedidas por categoría, país y año con todos sus subtotales

```
SELECT   CategoriaNombre, Pais, Año, SUM(UnidadesLinea) AS suma
FROM     Ventas_Fact INNER JOIN Producto_Dim ON
Ventas_Fact.ProductoKey = Producto_Dim.ProductoKey INNER JOIN
Cliente_Dim ON Ventas_Fact.ClienteKey = Cliente_Dim.ClienteKey
INNER JOIN Tiempo_Dim ON Ventas_Fact.TiempoKey =
Tiempo_Dim.TiempoKey
GROUP BY CUBE (CategoriaNombre, Pais, Año)
```

# ROLLUP

# CUBE

CategoriaNombre	Pais	Año	suma
Bebidas	UK	2003	264
Bebidas	UK	2004	133
Bebidas	UK	NULL	397
Bebidas	USA	2003	680
Bebidas	USA	2004	672
Bebidas	USA	NULL	1352
Bebidas	NULL	NULL	1749
Cárnicos	UK	2003	92
Cárnicos	UK	2004	40
Cárnicos	UK	NULL	132
Cárnicos	USA	2003	594
Cárnicos	USA	2004	291
Cárnicos	USA	NULL	885
Cárnicos	NULL	NULL	1017
NULL	NULL	NULL	2766

CategoriaNombre	Pais	Año	suma
Bebidas	UK	2003	264
Bebidas	UK	2004	133
Bebidas	UK	NULL	397
Bebidas	USA	2003	680
Bebidas	USA	2004	672
Bebidas	USA	NULL	1352
Bebidas	NULL	NULL	1749
Cárnicos	UK	2003	92
Cárnicos	UK	2004	40
Cárnicos	UK	NULL	132
Cárnicos	USA	2003	594
Cárnicos	USA	2004	291
Cárnicos	USA	NULL	885
Cárnicos	NULL	NULL	1017
NULL	NULL	NULL	2766
NULL	UK	2003	356
NULL	UK	2004	173
NULL	UK	NULL	529
NULL	USA	2003	1274
NULL	USA	2004	963
NULL	USA	NULL	2237
Bebidas	NULL	2003	944
Cárnicos	NULL	2003	686
NULL	NULL	2003	1630
Bebidas	NULL	2004	805
Cárnicos	NULL	2004	331
NULL	NULL	2004	1136

# Agrupamiento con operador GROUPING SETS

El operador **GROUPING SET** permite construir en una sola consulta múltiples grupos. Los grupos se combinan con un UNION ALL para ofrecer el resultado final

- Obtener el nº de unidades pedidas por categoría y país y por país y año

```

SELECT   CategoriaNombre, Pais, Año, SUM(UnidadesLinea) AS suma
FROM     Ventas_Fact INNER JOIN Producto_Dim ON
Ventas_Fact.ProductoKey = Producto_Dim.ProductoKey INNER JOIN
Cliente_Dim ON Ventas_Fact.ClienteKey = Cliente_Dim.ClienteKey
INNER JOIN Tiempo_Dim ON Ventas_Fact.TiempoKey =
Tiempo_Dim.TiempoKey
GROUP BY
GROUPING SETS ((CategoriaNombre, Pais),
(Pais, Año))
    
```

CategoriaNombre	Pais	Año	suma
Bebidas	UK	NULL	397
Bebidas	USA	NULL	1352
Cárnicos	UK	NULL	132
Cárnicos	USA	NULL	885
NULL	UK	2003	356
NULL	UK	2004	173
NULL	USA	2003	1274
NULL	USA	2004	963

# CUBE, ROLLUP vs GROUPING SETS

<p>CUBE (a, b, c) is equivalent to</p>	<p>GROUPING SETS ( (a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), () )</p>
<p>ROLLUP (a, b, c) is equivalent to</p>	<p>GROUPING SETS ( (a, b, c), (a, b), (a), () )</p>

## Generar filas con totales

El operador **ROLLUP** y **CUBE** generan los totales además de los subtotales. Para incorporar los totales sin los subtotales se utiliza **()** (*grant total*) dentro de **GROUPING SETS**.

- Obtener el nº de unidades pedidas por categoría y país y el total

```
SELECT   CategoriaNombre, Pais, SUM(UnidadesLinea) AS suma
FROM     Ventas_Fact INNER JOIN Producto_Dim ON
Ventas_Fact.ProductoKey = Producto_Dim.ProductoKey INNER JOIN
Cliente_Dim ON Ventas_Fact.ClienteKey = Cliente_Dim.ClienteKey
INNER JOIN Tiempo_Dim ON Ventas_Fact.TiempoKey =
Tiempo_Dim.TiempoKey
GROUP BY
GROUPING SETS ((CategoriaNombre, Pais), ( ))
```

# Grouping Function

Se utiliza para distinguir entre los valores NULL devueltos por CUBE o ROLLUP y los valores NULL normales. Permite conocer las filas con subtotales.

- Obtener el nº de unidades pedidas por categoría, año y país y el total, con los subtotales por categoría

```

SELECT   CategoriaNombre, Pais, Año, SUM(UnidadesLinea) AS suma,
           grouping (año) as GroupingAño,grouping (pais) as GroupingPais
FROM     Ventas_Fact INNER JOIN Producto_Dim ON
           Ventas_Fact.ProductoKey = Producto_Dim.ProductoKey INNER JOIN
           Cliente_Dim ON Ventas_Fact.ClienteKey = Cliente_Dim.ClienteKey
           INNER JOIN Tiempo_Dim ON Ventas_Fact.TiempoKey =
           Tiempo_Dim.TiempoKey
GROUP BY ROLLUP (
           CategoriaNombre, Pais, Año)
    
```

CategoriaNombre	Pais	Año	suma	GroupAño	GroupPais
Bebidas	UK	2003	264	0	0
Bebidas	UK	2004	133	0	0
Bebidas	UK	NULL	397	1	0
Bebidas	USA	2003	680	0	0
Bebidas	USA	2004	672	0	0
Bebidas	USA	NULL	1352	1	0
Bebidas	NULL	NULL	1749	1	1
NULL	NULL	NULL	1749	1	1

## SQL:2003 built-in functions for OLAP

---

- 34 nuevas funciones:
  - 7 funciones numéricas
  - 16 funciones para agregación
  - 5 funciones de ventana
  - 4 funciones sobre datos muestreados
  - 2 funciones de distribución inversa
- “windowed table” functions ofrecen capacidades para calcular promedios, desviaciones, correlaciones, etc.

## New built-in functions

- 7 new numeric functions

- LN (expr)
- EXP (expr)
- POWER (expr, expr)
- SQRT (expr)
- FLOOR (expr)
- CEIL[ING] (expr)
- WIDTH\_BUCKET(expr, expr, expr, expr)  
EX: WIDTH\_BUCKET (age, 0, 100, 10)

- 16 new aggregate functions

- STDDEV\_POP (expr)
- STDDEV\_SAMP (expr)
- VAR\_POP (expr)
- VAR\_SAMP (expr)
- COVAR\_POP (expr, expr)
- COVAR\_SAMP (expr, expr)
- CORR (expr, expr)
- REGR\_SLOPE (expr, expr)
- REGR\_INTERCEPT (expr, expr)
- REGR\_COUNT (expr, expr)
- REGR\_R2 (expr, expr)
- REGR\_AVGX (expr, expr)
- REGR\_AVGY (expr, expr)
- REGR\_SXX (expr, expr)
- REGR\_SYY (expr, expr)
- REGR\_SXY (expr, expr)

# Windowed Table functions

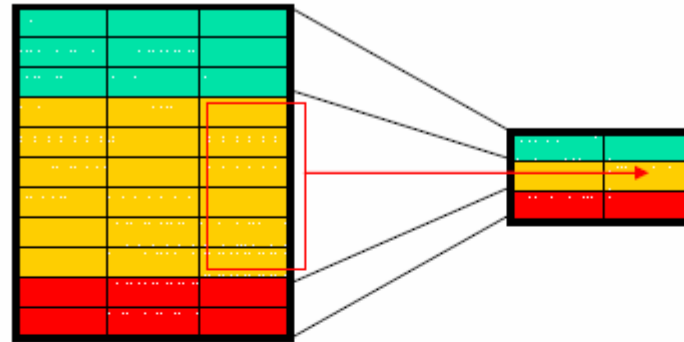
- Windowed table function
  - Opera sobre una ventana de la tabla
  - Devuelve un valor por cada fila en la ventana
  - El valor se calcula con las filas que se encuentra en la ventana
- Funciones incorporadas:
 

<ul style="list-style-type: none"> <li>■ RANK () OVER</li> <li>■ DENSE_RANK () OVER</li> <li>■ PERCENT_RANK () OVER</li> <li>■ CUME_DIST () OVER</li> <li>■ ROW_NUMBER () OVER</li> </ul>	}	Funciones de clasificación
<ul style="list-style-type: none"> <li>■ PERCENT_RANK () OVER</li> <li>■ CUME_DIST () OVER</li> </ul>	}	Funciones de distribución
<ul style="list-style-type: none"> <li>■ ROW_NUMBER () OVER</li> </ul>	}	Función de numeración de filas
- También se pueden utilizar como funciones ventanas:
  - Las funciones agregadas (COUNT, SUM, MAX, MIN, AVG, EVERY, ANY O SOME)
  - Y las 16 funciones agregadas nuevas
    - Ej: SUM(salario) OVER ...

# Comparación

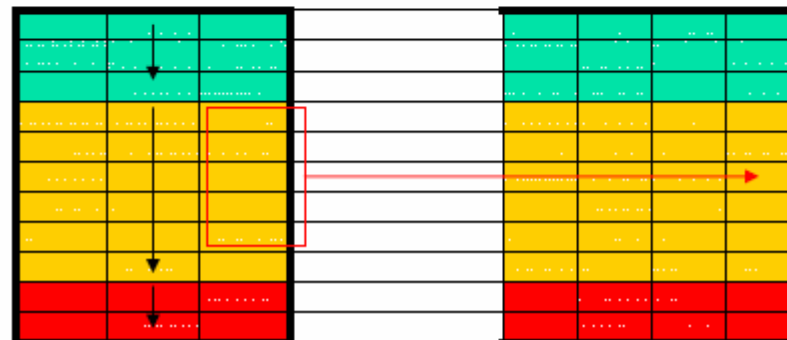
- Set functions  
(aggregate functions)

```
SELECT dept, AVG(salary)
FROM Employees
GROUP BY dept
```



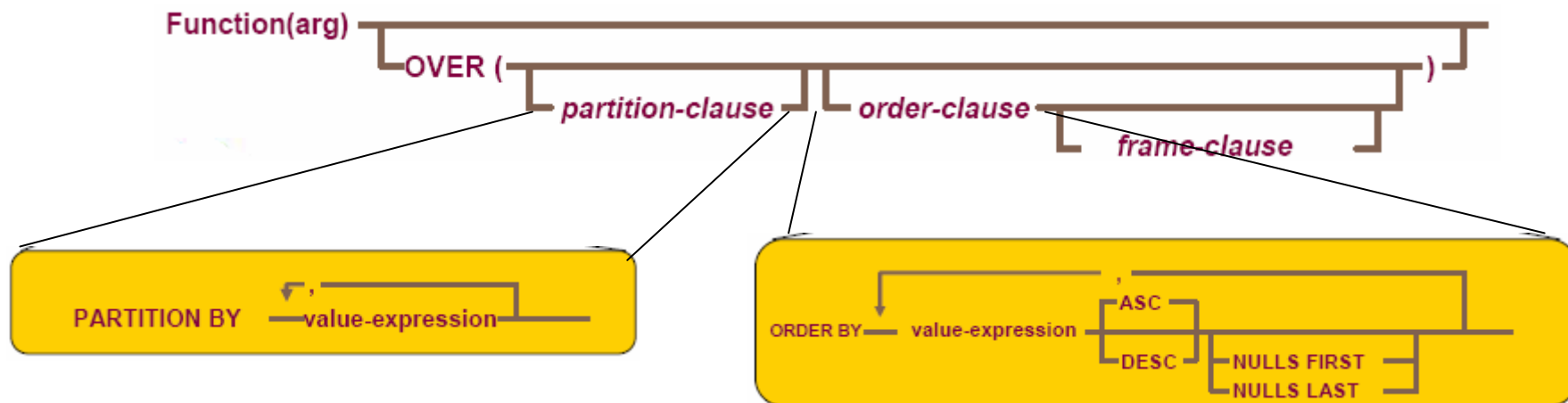
- Windowed Table Functions  
(tuple-based aggregation)

```
SELECT dept, empno, salary,
AVG(salary) OVER(
PARTITION BY dept
ORDER BY age )
FROM Employees
```



## Sintaxis estándar

- Ventana se define con la cláusula OVER
  - **partition\_clause**: criterio para formar grupos, aunque las filas se mantienen. Si no se indica, sólo hay una partición.
  - **order\_clause**: orden de las filas dentro de la partición
  - **frame\_clause**: especificación del rango de filas relativa a la fila actual que participaría en la función de evaluación



## Ejemplo

- Obtener el salario promedio de los empleados por cada localidad

```
SELECT EmpleadoID, Localidad, Salario,
       AVG(salario) over (partition by localidad) Salario_promed
FROM Empleados
ORDER BY localidad
```

	EmpleadoID	Localidad	Salario	Salario_promed
1	3	Kirkland	3100,00	3100,00
2	5	London	4500,00	3337,50
3	6	London	3000,00	3337,50
4	7	London	2850,00	3337,50
5	9	London	3000,00	3337,50
6	4	Redmond	2950,00	2950,00
7	1	Seattle	3000,00	4000,00
8	8	Seattle	5000,00	4000,00
9	2	Tacoma	30000,00	30000,00

## Funciones ventana

- Las funciones ventana se computan después de aplicar el FROM, WHERE, GROUP y HAVING
  - No se pueden utilizar en ninguna de estas cláusulas
  - Se pueden utilizar funciones de agregación en la definición de la ventana
  - Si quieres referirte a ellas debes anidarlas o usar una expresión de tabla común
- 
- Clasificación de las localidades en función de los salarios de sus empleados

```
SELECT Localidad, sum(Salario),
       rank() over (order by sum(salario)) as posición
FROM Empleados
GROUP BY Localidad
```

	Localidad	(Sin nombre de columna)	posición
1	Redmond	2950,00	1
2	Kirkland	3100,00	2
3	Seattle	8000,00	3
4	London	13350,00	4
5	Tacoma	30000,00	5

## Ejemplo

ORDER BY necesario en  
funciones ventana

- Obtener el ranking de los empleados según su salario, asignar un nº de fila y distribuir los empleados en 4 grupos según salario

```
SELECT EmpleadoID, Localidad, Salario,
rank() over (order by salario ) as Posición,
dense_rank() over (order by salario ) as Posición_sinhuecos,
row_number() over (order by salario ) as Nro_fila,
cume_dist() over (order by salario ) as Percentil_acum
FROM Empleados
ORDER BY salario
```

EmpleadoID	Localidad	Salario	Posición	Posición_sin	Nro_fila	Percentil_acum
7	London	2850	1	1	1	0
4	Redmond	2950	2	2	2	0,125
1	Seattle	3000	3	3	3	0,25
6	London	3000	3	3	4	0,25
9	London	3000	3	3	5	0,25
3	Kirkland	3100	6	4	6	0,625
5	London	4500	7	5	7	0,75
8	Seattle	5000	8	6	8	0,875
2	Tacoma	30000	9	7	9	1

## Ejemplo agregaciones anidadas

- Obtener los ingresos de ventas por trimestre y acumulado por año

```
SELECT Año, Trimestre, sum(TotalLinea) as Suma,
sum (sum (TotalLinea)) over (partition by año order by trimestre) as
Acumulado
FROM Ventas_Fact INNER JOIN
Tiempo_Dim ON Ventas_Fact.TiempoKey = Tiempo_Dim.TiempoKey
GROUP BY Año, Trimestre
```

sin especificación "window frame" el cálculo se hace con todas las iguales filas o precedentes a la actual dentro de la misma partición (RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

Año	Trimestre	Suma	Acumulado
2002	3	84437,5	84437,5
2002	4	141861	226298,5
2003	1	147879,9	147879,9
2003	2	151611,09	299490,99
2003	3	165179,64	464670,63
2003	4	193718,12	658388,75
2004	1	315242,12	315242,12
2004	2	127085,46	442327,58

# Ejemplo Agregaciones usando expresión de tabla común (CTE)

- Obtener los ingresos de ventas por trimestre y acumulado por año

**WITH** tablaLineas **AS**

(select Año, Trimestre, sum(TotalLinea) as TotalTrimestre

FROM Ventas\_Fact INNER JOIN

    Tiempo\_Dim ON Ventas\_Fact.TiempoKey = Tiempo\_Dim.TiempoKey

GROUP BY Año, Trimestre)

**SELECT** Año, Trimestre, TotalTrimestre

sum (TotalTrimestre) over (partition by Año order by Trimestre ) as Acumulado

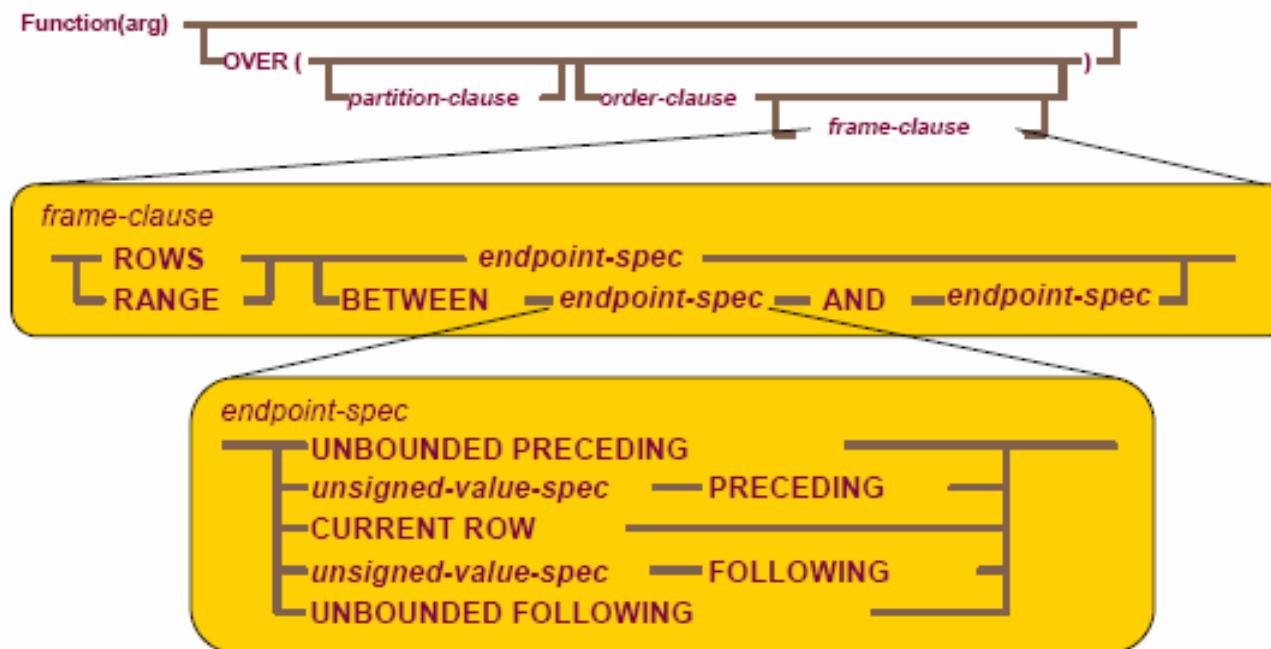
**FROM** tablaLineas

order by año, trimestre

Año	Trimestre	Suma	Acumulado
2002	3	84437,5	84437,5
2002	4	141861	226298,5
2003	1	147879,9	147879,9
2003	2	151611,09	299490,99
2003	3	165179,64	464670,63
2003	4	193718,12	658388,75
2004	1	315242,12	315242,12
2004	2	127085,46	442327,58

## Window frame

- Refinan el conjunto de filas de una ventana cuando existe el “order by”. Permiten incluir/excluir rangos de valores o filas dentro de la ordenación



## Ejemplo

- Obtener para cada empleado su sueldo promedio con respecto al sueldo del anterior y el siguiente en su departamento

```
SELECT department_id, employee_id, salary,
avg(salary) OVER (PARTITION BY department_id ORDER BY salary ROWS
BETWEEN 1 PRECEDING AND 1 FOLLOWING) promedio_especial
FROM employees
```

DEPARTMENT_ID	EMPLOYEE_ID	SALARY	PROMEDIO
10	200	4400	4400
20	202	6000	9500
20	201	13000	9500
30	119	2500	2550
30	118	2600	2633,3333333...
30	117	2800	2766,6666666...
30	116	2900	2933,3333333...
30	115	3100	5666,6666666...
30	114	11000	7050

# Funciones analíticas en ORACLE

AVG \*  
CORR \*  
COVAR\_POP \*  
COVAR\_SAMP \*  
COUNT \*  
CUME\_DIST  
DENSE\_RANK  
FIRST  
FIRST\_VALUE \*  
LAG  
LAST  
LAST\_VALUE \*  
LEAD  
MAX \*  
MIN \*  
NTILE  
PERCENT\_RANK  
PERCENTILE\_CONT  
PERCENTILE\_DISC  
RANK  
RATIO\_TO\_REPORT  
REGR\_ (Linear Regression) Functions \*  
ROW\_NUMBER  
STDDEV \*  
STDDEV\_POP \*  
STDDEV\_SAMP \*  
SUM \*  
VAR\_POP \*  
VAR\_SAMP \*  
VARIANCE \*

asterisk (\*) allow the full syntax, including the *windowing\_clause*.

## Hypothetical aggregate functions

---

- Se calcula la clasificación, percentil, etc. para una fila hipotética que se especifica en los argumentos de la función
- Funciones:
  - ▶ RANK (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)
  - ▶ DENSE\_RANK (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)
  - ▶ PERCENT\_RANK (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)
  - ▶ CUME\_DIST (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)

## Ejemplo

- Obtener la posición que tendría un empleado con un salario de 2300

```
SELECT employee_id, job_id, salary,
rank() over (order by salary ) as posición FROM employees
```

EMPLOYEE_ID	JOB_ID	SALARY	POSICIÓN
132	ST_CLERK	2100	1
128	ST_CLERK	2200	2
136	ST_CLERK	2200	2
127	ST_CLERK	2400	4
135	ST_CLERK	2400	4
119	PU_CLERK	2500	6
131	ST_CLERK	2500	6
140	ST_CLERK	2500	6
144	ST_CLERK	2500	6
182	SH_CLERK	2500	6
191	SH_CLERK	2500	6
118	PU_CLERK	2600	12

```
SELECT
rank(2300) within group (order by salary asc)
FROM employees
order by salary
```

```
RANK(2300)WITHINGROUP(ORDERBYSALARYASC)
4
```

## Ejemplo

- Obtener la posición que tendría un empleado hipotético con 15500 y comisión del 5%

```
SELECT DENSE_RANK(15500, .05) WITHIN GROUP
(ORDER BY salary , commission_pct) "Dense Rank"
FROM employees;
```

Dense Rank
3

# Inverse Distribution Functions

---

- Devuelven el valor de las expresiones especificadas en <sort specification list> que corresponden al valor del percentil especificado
- Funciones
  - ▶ PERCENTILE\_DISC (expr) WITHIN GROUP (ORDER BY <sort specification list>)
  - ▶ PERCENTILE\_CONT (expr) WITHIN GROUP (ORDER BY <sort specification list>)

## Ejemplo

- Computa el salario medio en cada departamento

```
SELECT department_id,
PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary DESC)
"Median cont",
PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY salary DESC)
"Median disc"
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	Median cont	Median disc
10	4400	4400
20	9500	13000
30	2850	2900
40	6500	6500
50	3100	3100
60	4800	4800
70	10000	10000
80	8900	9000
90	17000	17000
100	8000	8200
110	10150	12000

PERCENTILE\_CONT devuelve el resultado después de hacer la interpolación lineal

PERCENTILE\_DISC devuelve un valor del conjunto de valores que son agregados

Pueden no coincidir