Desarrollo de Sistemas de Información

Tema 9. Diseño Multidimensional

Marta Elena Zorrilla Pantaleón

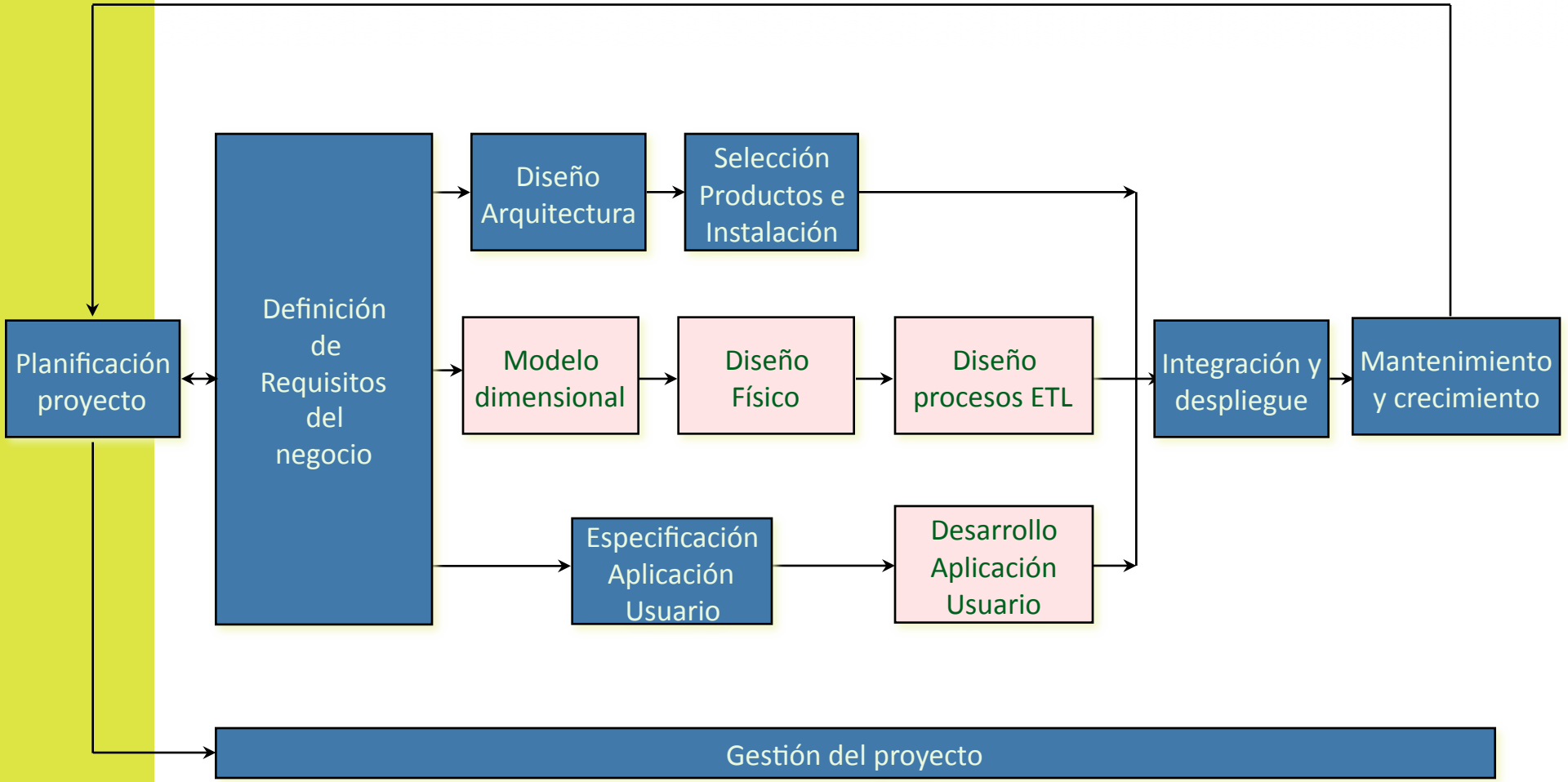
Dpto. de Matemáticas, Estadística y Computación

Tabla de contenido

¿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

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

Planificación del proyecto

- Riegos:
  - Respecto a la gestión del proyecto
    - Coordinación entre secciones, información que no se puede ofrecer, miedo a perder el poder de los jefes
    - Proyectos de larga duración
  - Respecto a la tecnología
    - Elección de herramientas que faciliten la escalabilidad en términos de datos y usuarios
    - Facilidad para incorporar nuevas soluciones tecnológicas (definición componentes y metadatos completos)
    - Habilidad en la construcción de los componentes
  - Respecto al diseño y los datos
    - Poca calidad de los datos
    - No acceso a las fuentes originales con la frecuencia necesaria (actualización)
    - No propiedad de los datos
    - Difícil integración
  - Respecto a la organización
    - No implicar a usuarios finales
    - Cambio cultural en la empresa
    - No tomar ventaja de los resultados

Definición de requisitos

- Seleccionar la información relevante en relación a los objetivos de la empresa
  - Listar “hechos” y los “procesos” que los soportan
  - Caracterizar estos hechos y especificar métricas y perspectivas de observación
    - 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
  - Listar cuestiones a responder e informes a generar
    - Consultas a las que se quiere dar respuesta

Diseño dimensional y físico

- Diseño conceptual
  - Diferentes propuestas: ***Dimensional Fact Model*** que modeliza los hechos, las medidas, las dimensiones y jerarquías (Golfarelli et al.); extensiones de UML (Abelló et al.), etc. Ninguno implementado en herramientas CASE, se diseñan con ER o UML.
- Validación del diseño conceptual
  - Consultas sobre el diseño conceptual para comprobar que los requisitos se cumplen
- Diseño lógico
  - Construir el lógico en estrella, copo de nieve, constelación. Determinar si se implantará de forma ROLAP o MOLAP. En caso de ROLAP especificar qué vistas materializadas y fragmentaciones en función de las consultas que se requieran (fase validación)
- Diseño físico
  - Determinar el gestor donde se implantará y determinar estrategias de indexación y técnicas de rendimiento en éste

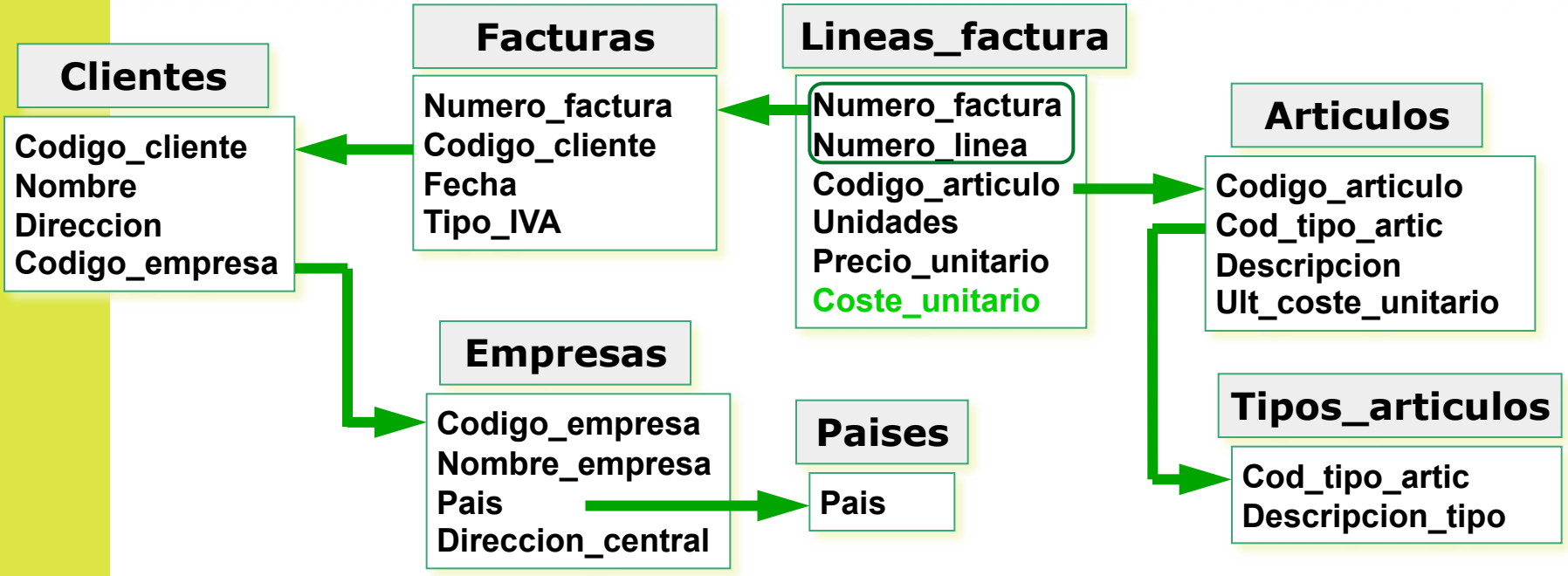
Evaluación de herramientas

- 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 de Microsoft.
- Herramientas EIS, caras y monotemáticas (desuso). Excel Add-ins, la alternativa de cliente más sencilla.
  - PivotTable de Excel no suficiente
  - XLCubed, IntelligentApps, MIS AG: comerciales, muy potentes
  - JPALO client, Jpalo Web client: open source, menos prestaciones
- BI platform - Open source (www.sourceforge.net):
  - JasperSoft Business Intelligence Suite: MySQL, Jasper Report, Mondrian OLAP, Jasper ETL, etc.
  - Pentaho project: reporting, analysis, dashboard, data mining and workflow (firebird RDBMS, Weka DM, Mondrian OLAP, Enhydra ETL, JaWE workflow, BIRT reporting components)
  - Vanilla - a True Open Source BI Platform sponsored by BPM-Conseil.

MODELO DE DATOS DIMENSIONAL

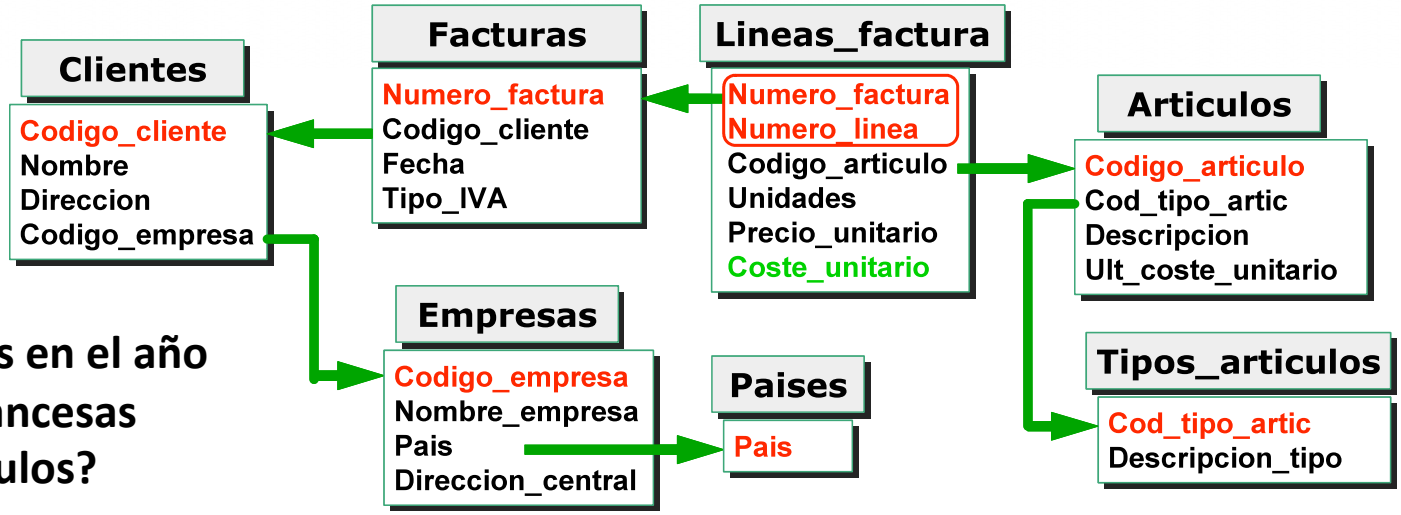
Tabla de contenido

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

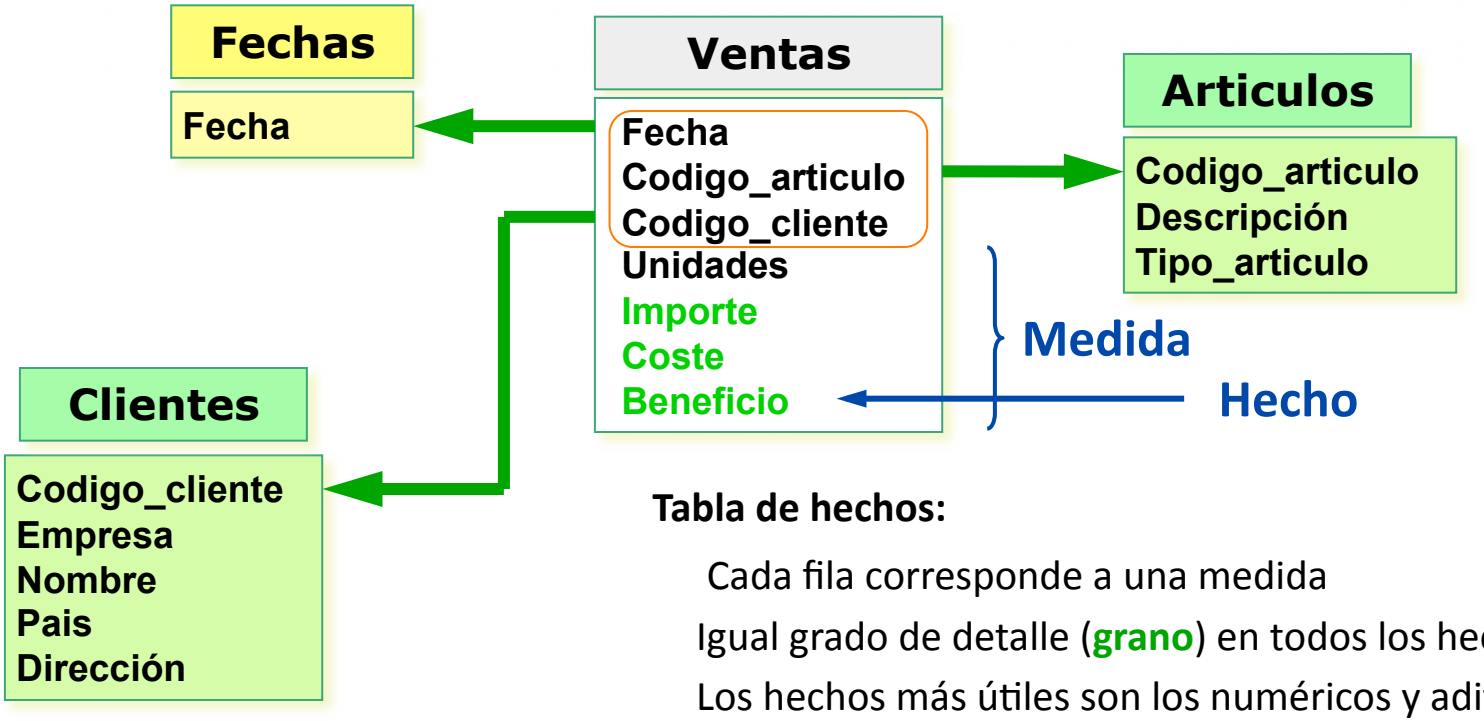


Tabla de hechos:

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

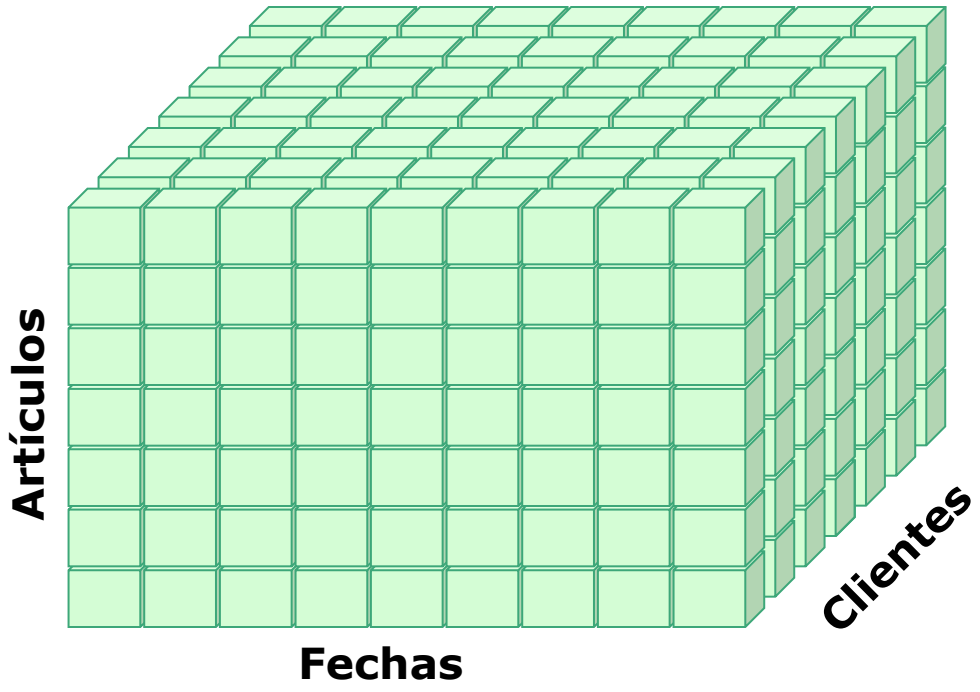
Son los puntos de entrada en la tabla de hechos

Transformación de datos

Desnormalización

Aún no es el modelo dimensional

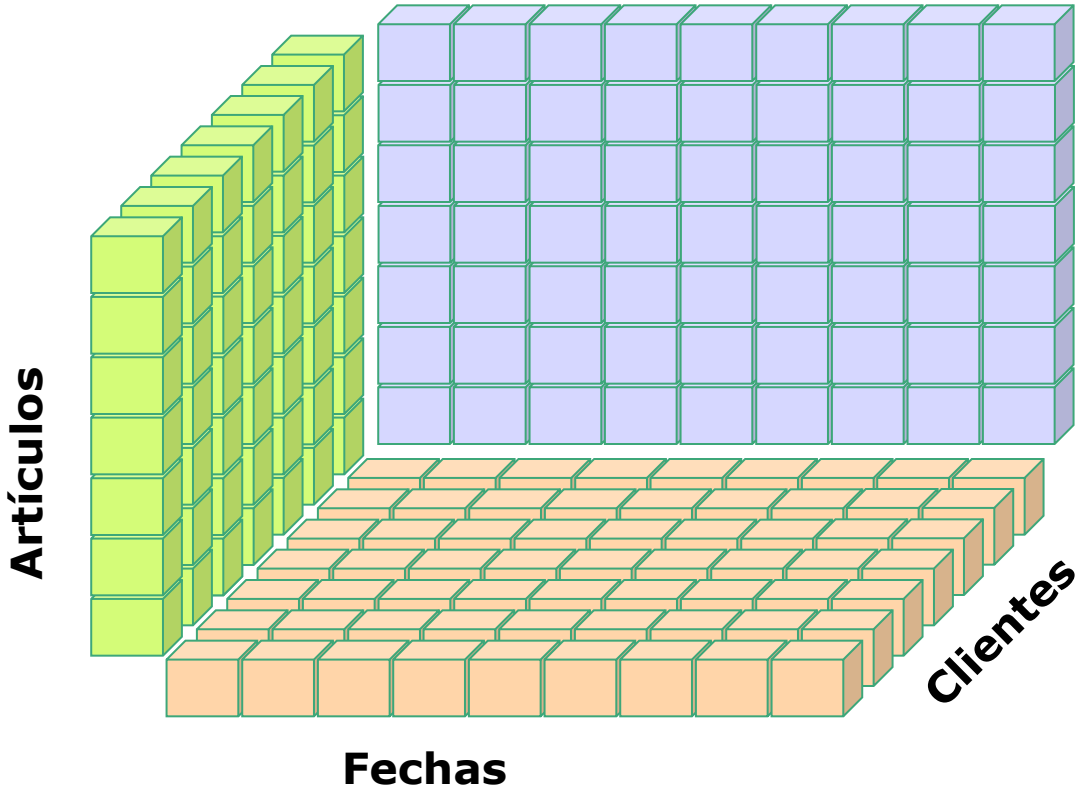
Cubos e hipercubos



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.

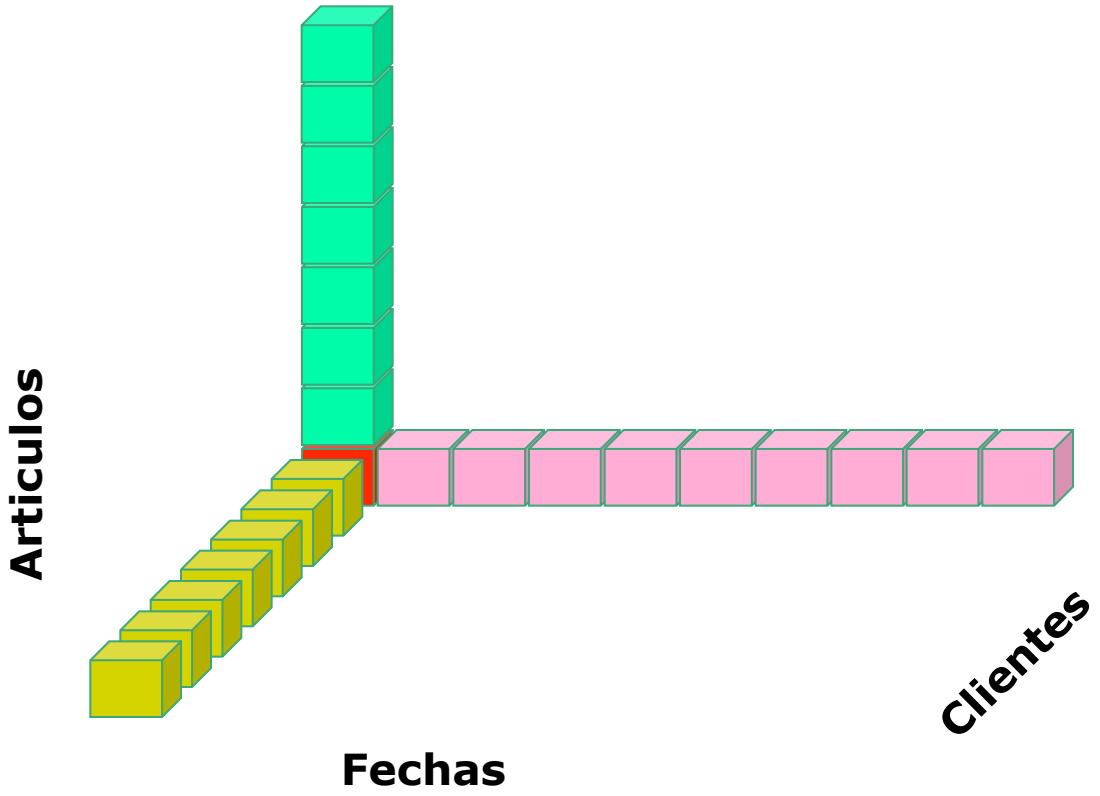
Acumulados según una dimensión



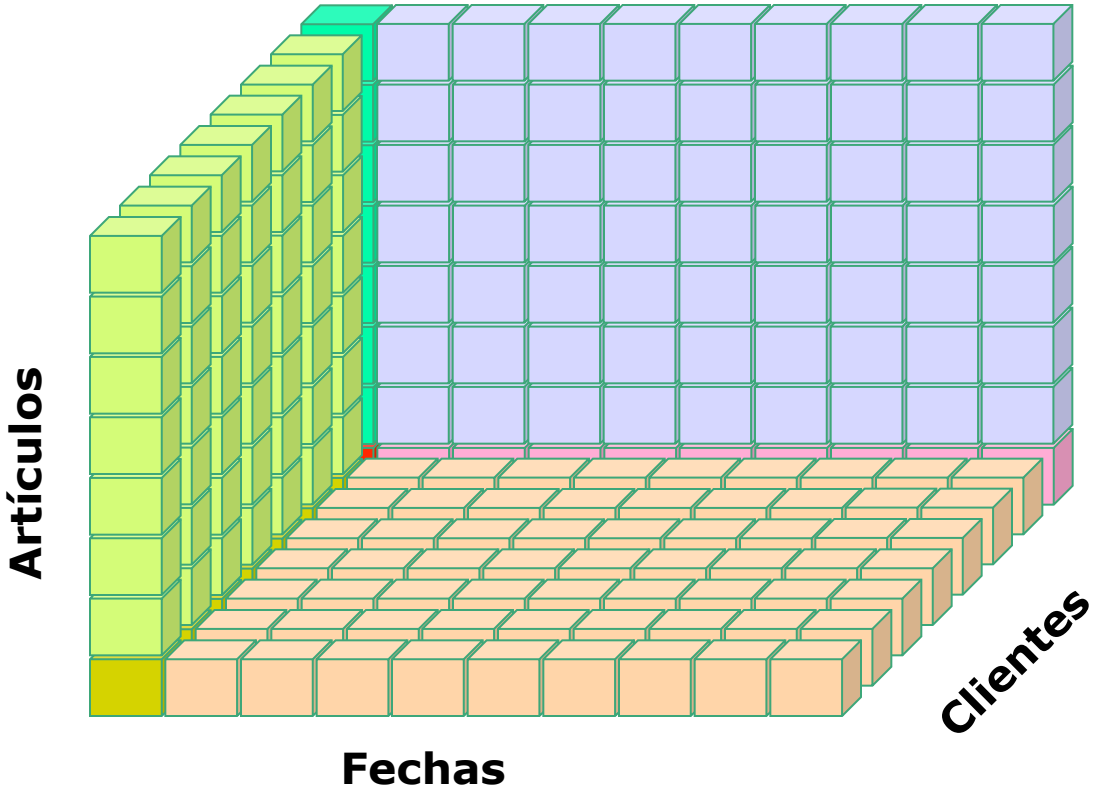
Las medidas (como el beneficio) tiene la propiedad de ser aditivas. Es decir, tiene sentido la suma según todas las dimensiones (beneficios en una fecha, o con un artículo o con relación a un cliente).

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

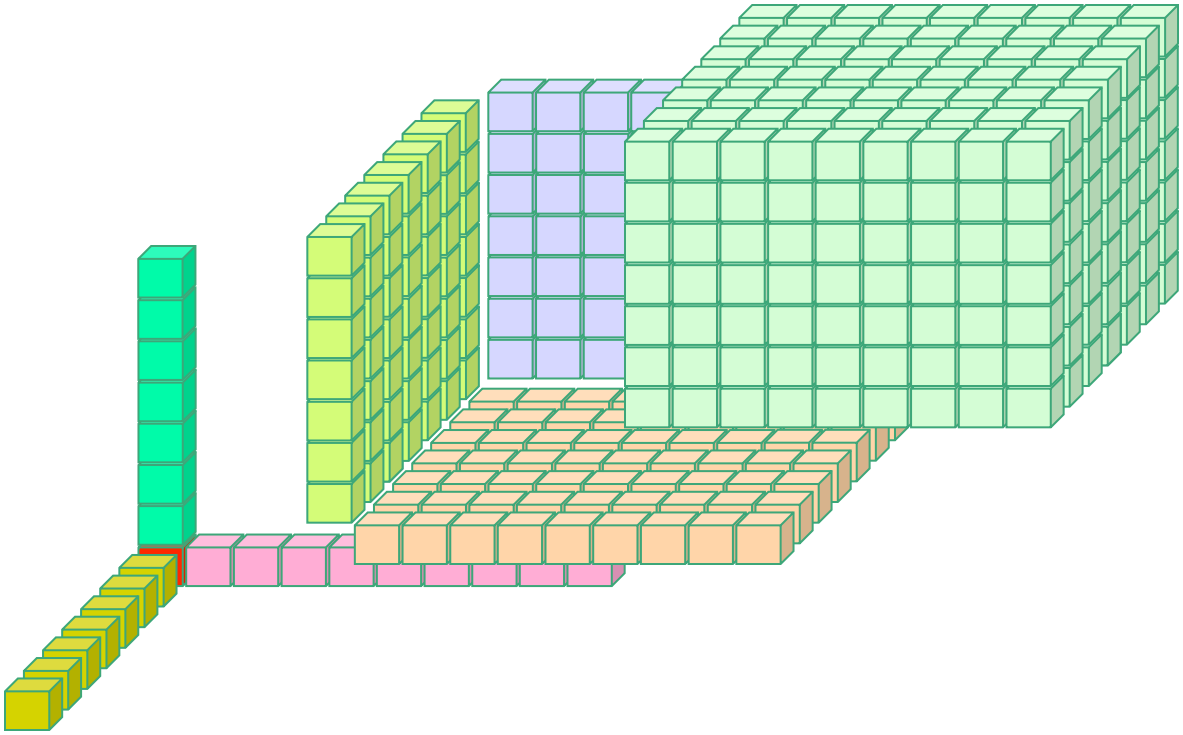
Acumulados según dos y las tres dimensiones



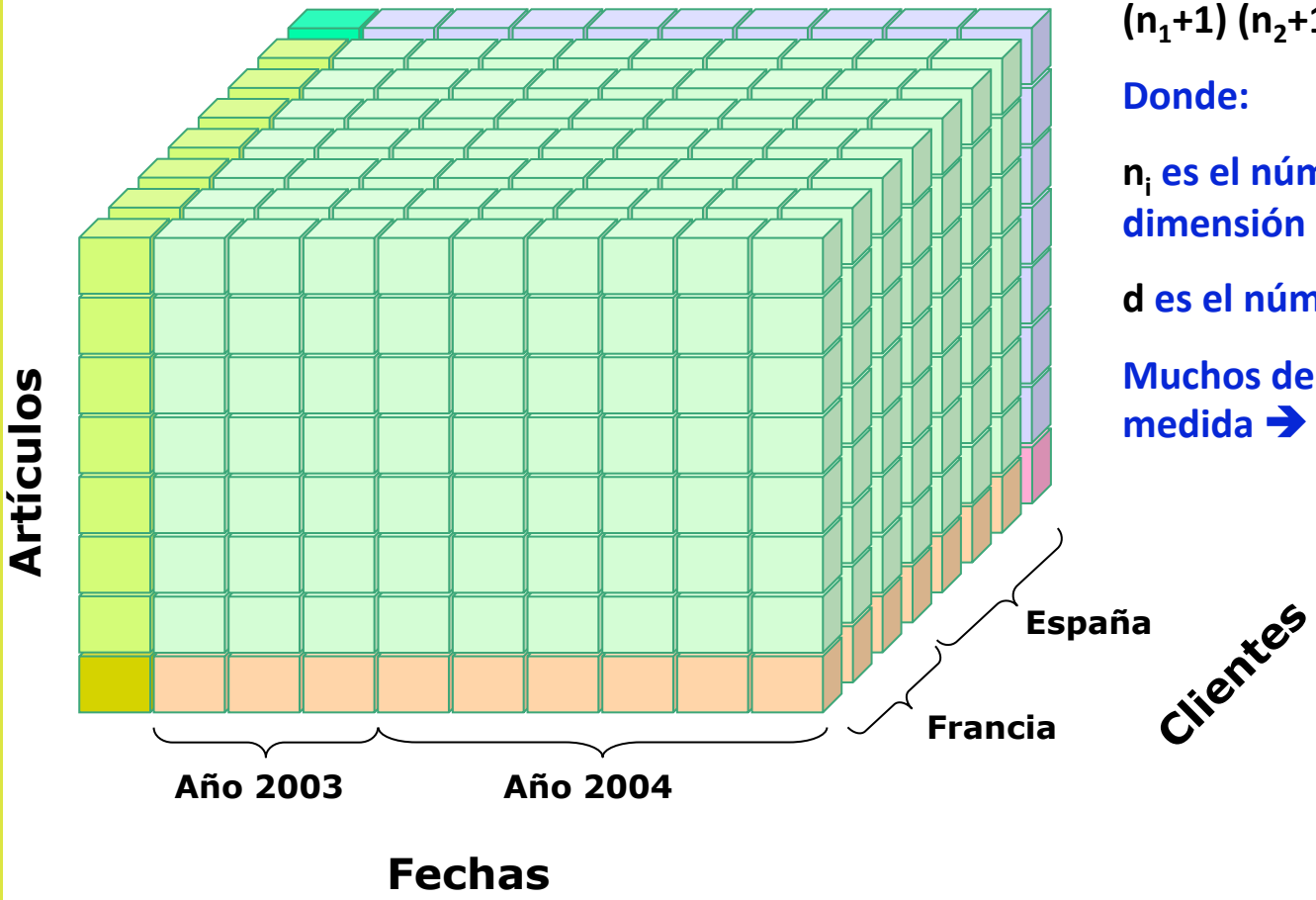
Acumulados según todas las dimensiones



UNIÓN DEL CUBO Y SUS ACUMULADOS



El cubo de medidas con todos sus acumulados



Tamaño máximo del cubo:

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

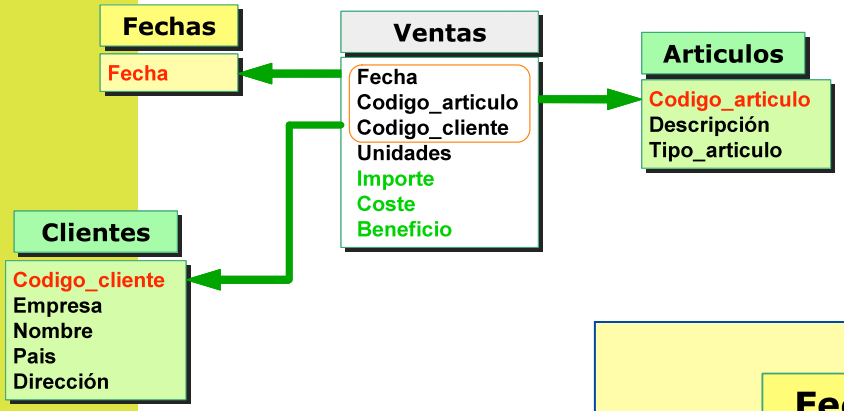
Donde:

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 ➔ (no ocupan espacio)

Modelo de datos dimensional (lógico)



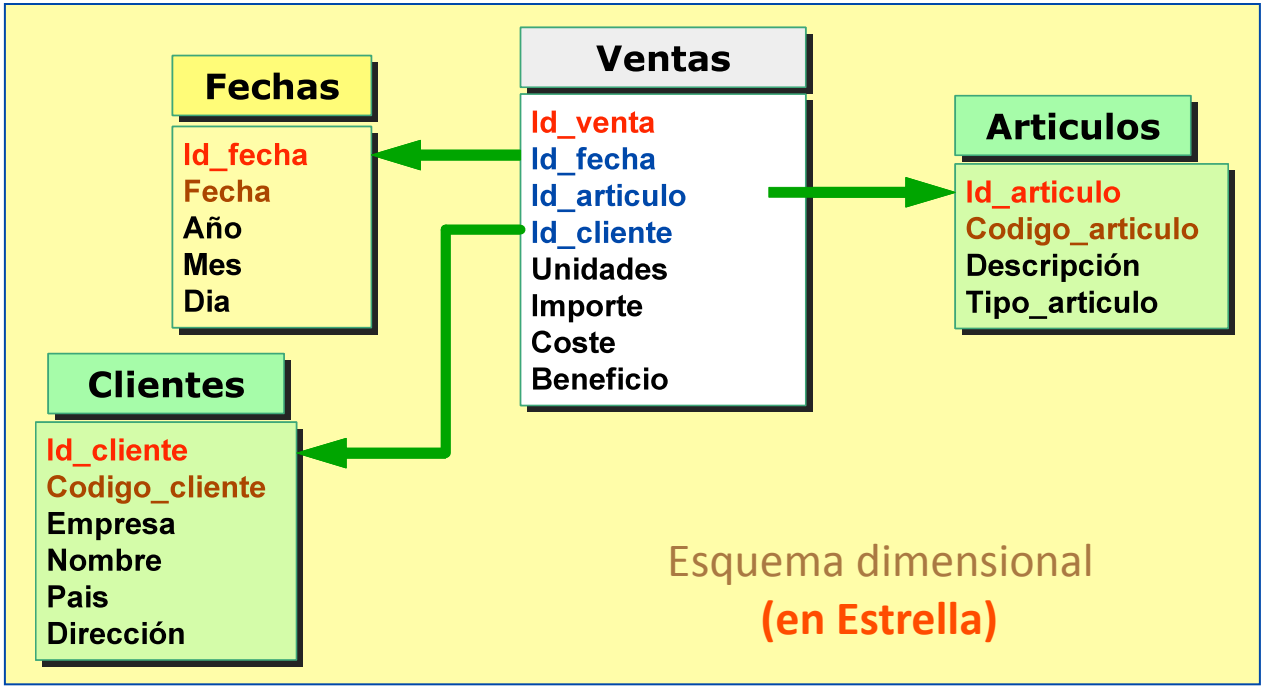
Tablas de dimensión:

- Claves de gestión
- Atributos [criterios de agregación]
- Claves simples (autonuméricas)

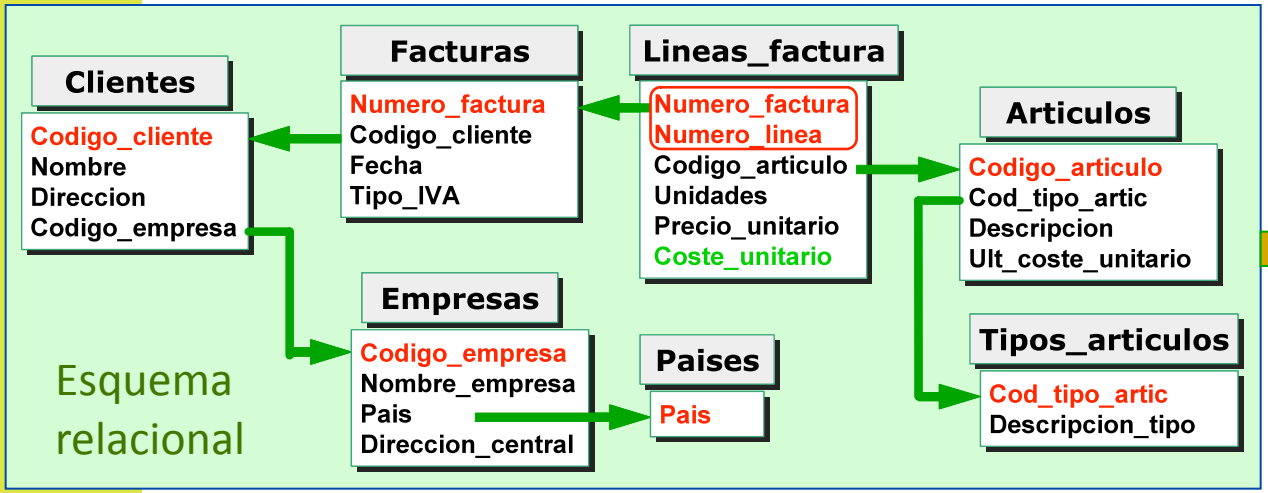
Tabla de hechos:

- Atributos (hechos) [aditividad]
- Claves de referencia
- Clave simple (autonumérica)

Claves propias sin significado ➔ Independencia ante cambios de claves de producción



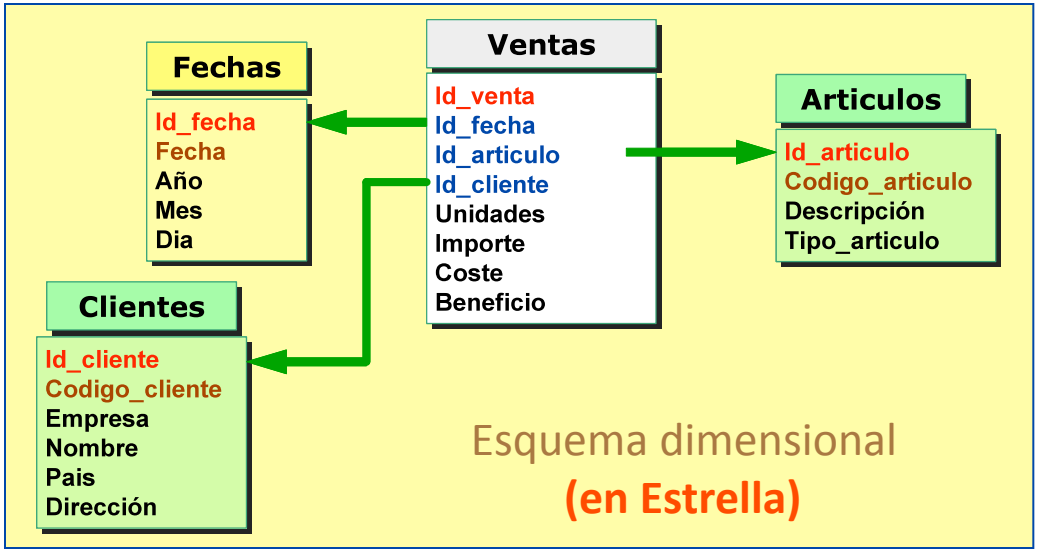
Del modelo relacional al dimensional



Extracción

Transformación

Carga



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

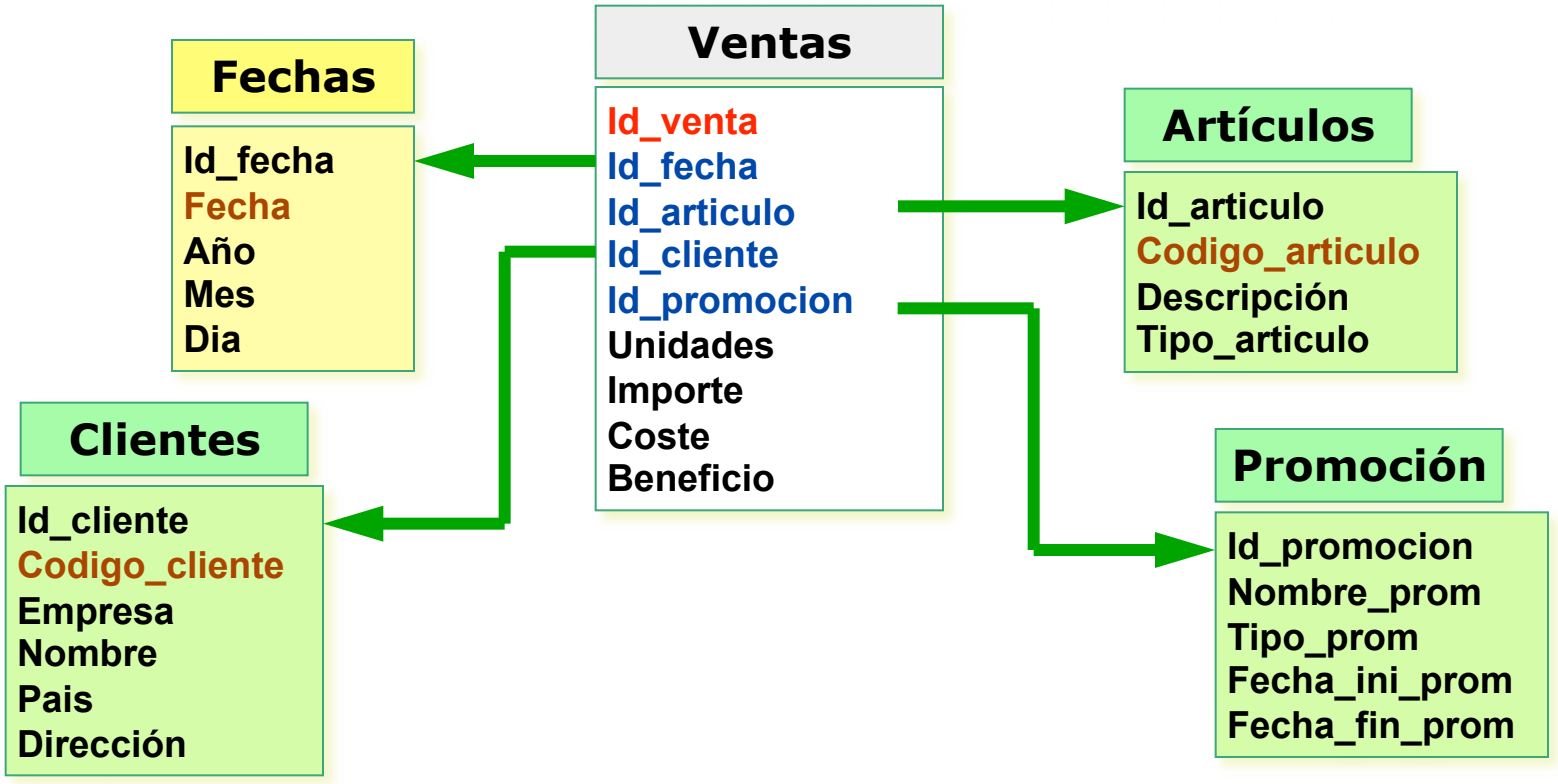
Nuevas medidas:

- 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

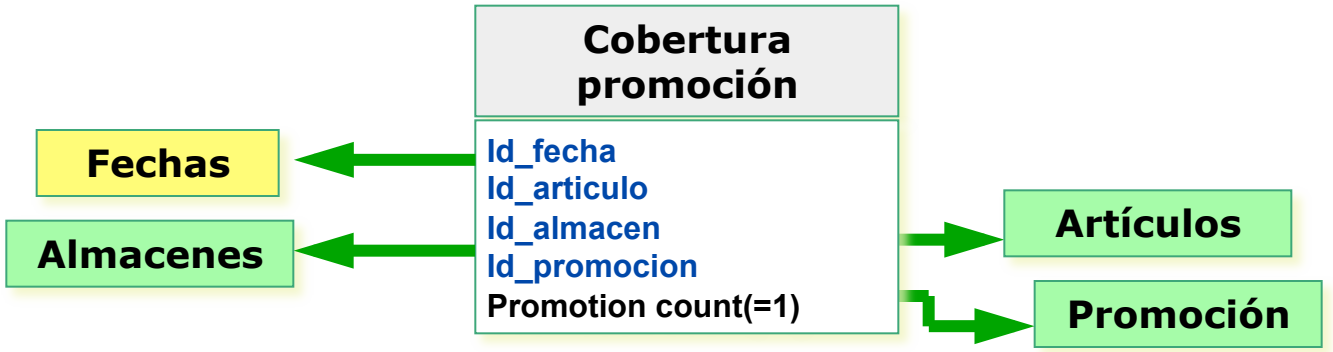
Para ello, deberá haber una fila en la dimensión que indique que no es aplicable

Ejemplo: ventas sin promoción

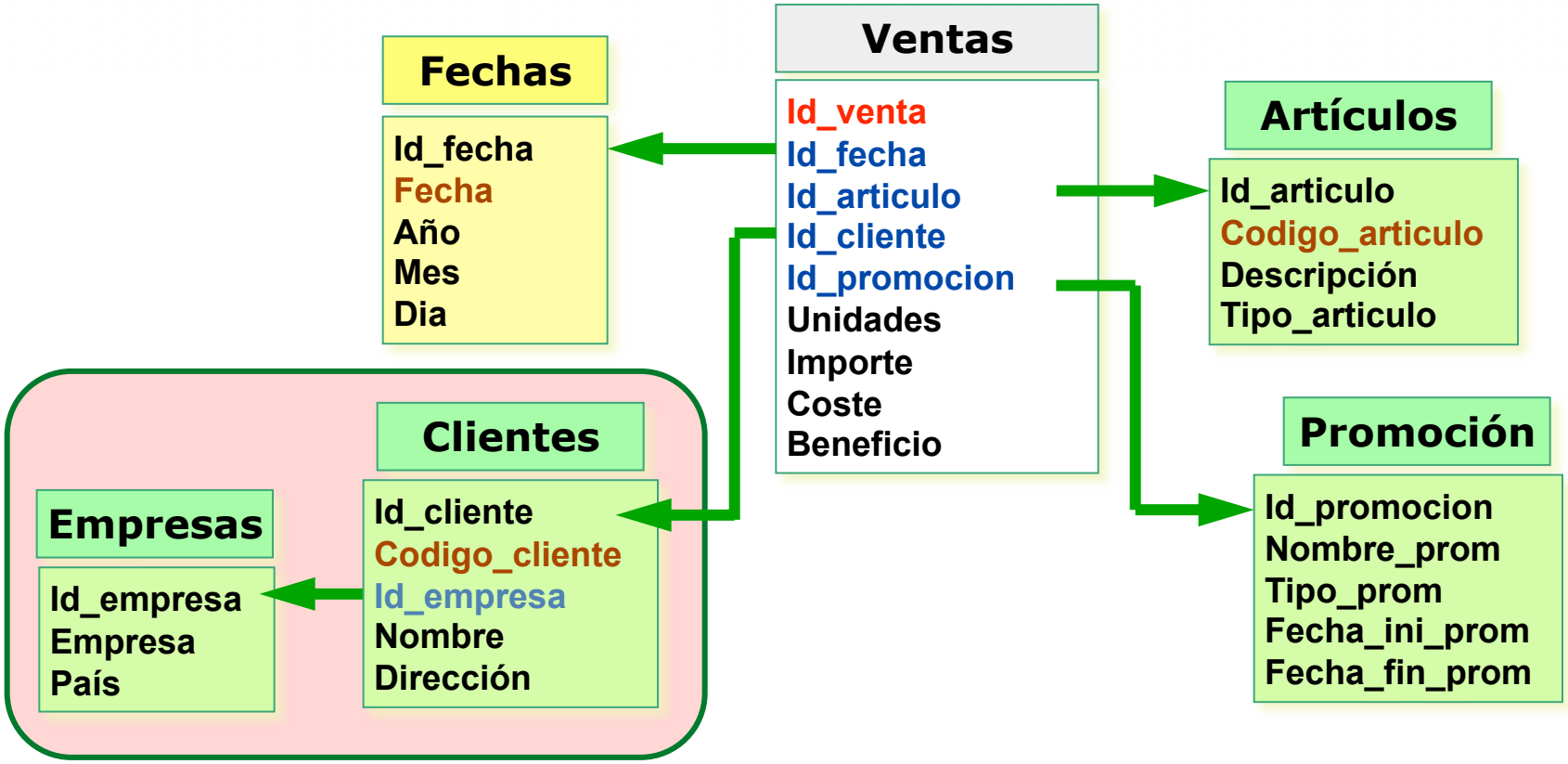
Tabla de hechos sin hechos

- 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)

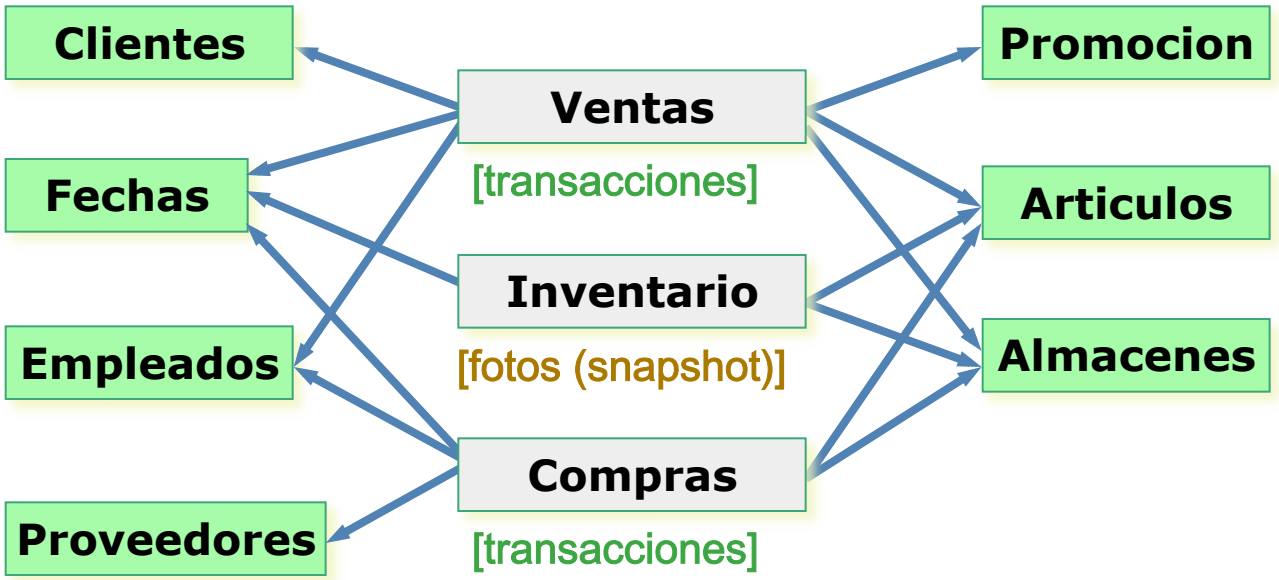


Algunas dimensiones se jerarquizan en varias tablas (normalización).

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)

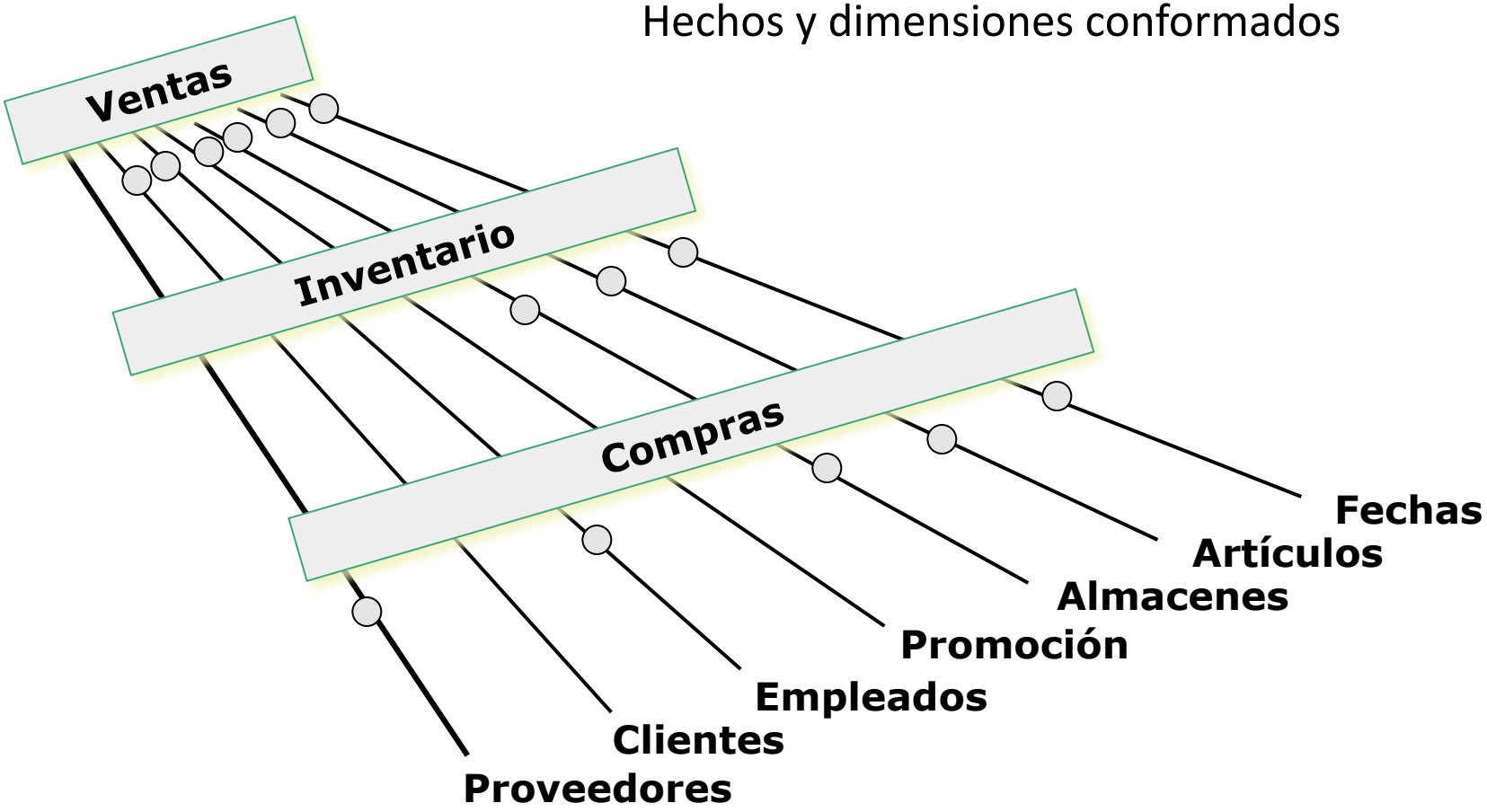
Margen de retorno de inventario = $\frac{\text{Uds\_vendidas} * (\text{precio\_ult\_venta} - \text{precio\_coste})}{\text{Uds\_entradas} * \text{precio\_ult\_venta}}$

Data Warehouse vs Data Marts

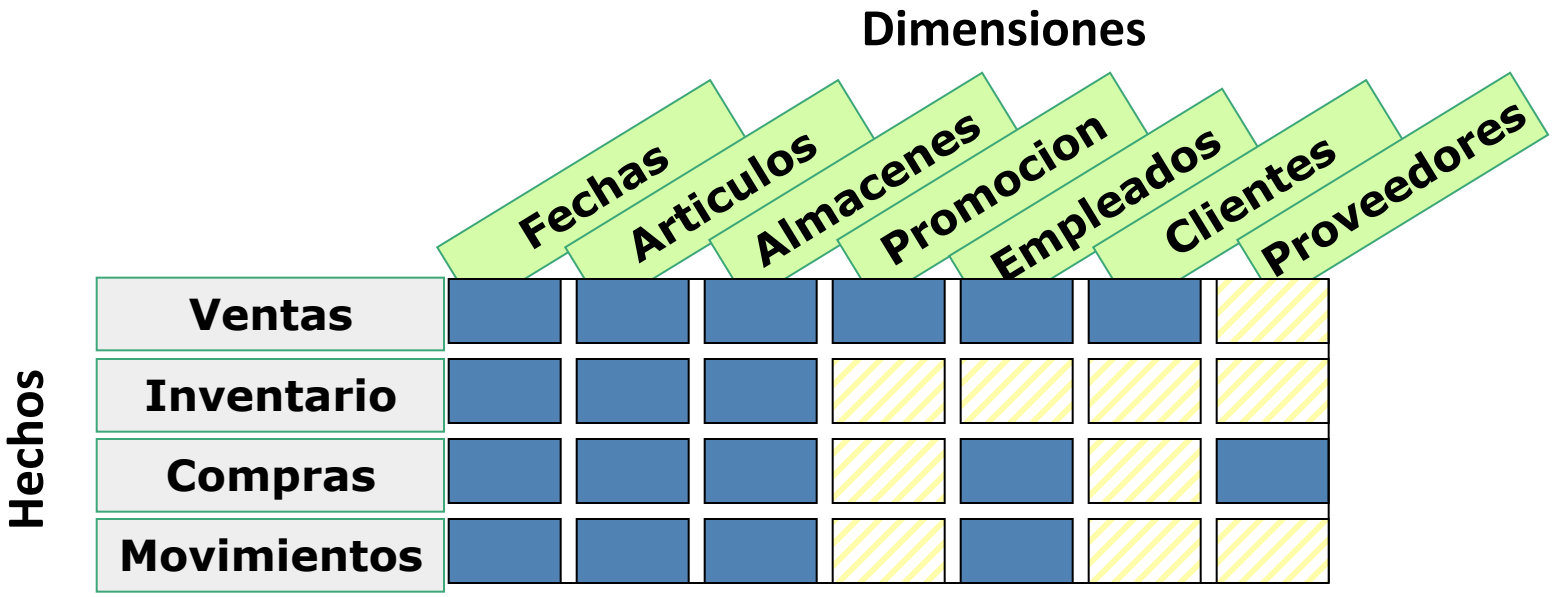
Arquitectura en Bus del Data Warehouse (Kimball)

Permite un diseño del DWH incremental

Hechos y dimensiones conformados



Matriz en Bus del BI/DW (Kimball)



Dimensiones conformadas (I)

- Cuando se diseñan las estrellas separadamente, sus dimensiones están concebidas (atributos, nivel de detalle, ...) para cada una de ellas
- Al compartir dimensiones, puede suceder que éstas no tengan el mismo nivel de detalle en todas las estrellas
- Es preciso que toda dimensión signifique lo mismo para cada tabla de hechos con la que se relacione.
  - Ejemplos: Fechas, Artículos, Almacenes, Clientes, ...
- Sin dimensiones conformadas, el data warehouse no podrá funcionar como un todo

Dimensiones conformadas (II)

- Las dimensiones conformadas hacen posible que:
  - Una única tabla de dimensión se pueda utilizar frente a varias tablas de hechos
  - El contenido de los datos sea coherente
  - Las interfaces de usuario sean consistentes
  - Haya una interpretación uniforme de los atributos
- El equipo diseñador es responsable de establecer, publicar y mantener las dimensiones conformadas
- Su definición puede llevar bastante tiempo
- Deben diseñarse con la granularidad más fina posible
- Deben tener una clave sin significado que permita realizar cambios en el futuro
- Las tablas jerarquizadas de un snowflake pueden ser dimensiones de otras estrellas

Hechos conformados

- Los datos de la tabla de hechos generalmente no son duplicados en diferentes data marts. Si existen, uno a nivel de detalle y otros consolidados.
- Se requiere que los nombres de los hechos sean únicos y con una interpretación clara e inequívoca
  - Ejemplos: beneficio, coste, precio, medidas de calidad, medidas de satisfacción del cliente
- Deben utilizarse las mismas unidades de medida
  - Ejemplo: cantidad de fabricación de un producto (en kilos) y unidades (latas por ejemplo) del mismo en el almacén

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

Kimball vs Inmon

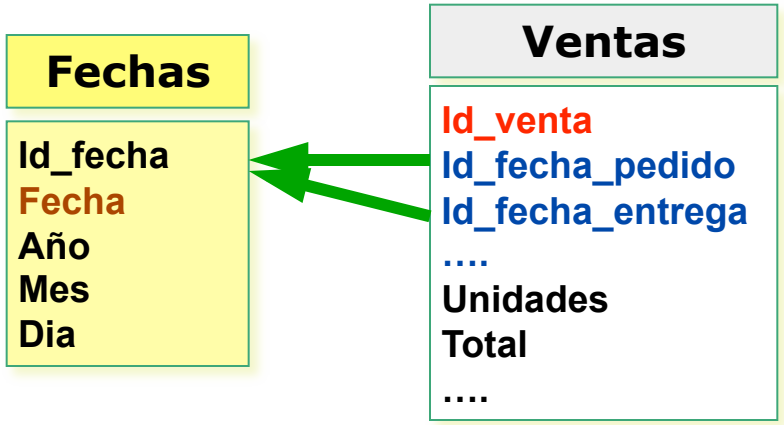
- **Bill Inmon's paradigm**: Data warehouse is one part of the overall business intelligence system. An enterprise has one data warehouse, and data marts source their information from the data warehouse. In the data warehouse, information is stored in 3rd normal form.

- **Ralph Kimball's paradigm**: Data warehouse is the conglomerate of all data marts within the enterprise. Information is always stored in the dimensional model.

MODELO DIMENSIONAL EXTENDIDO

Roles de una dimensión

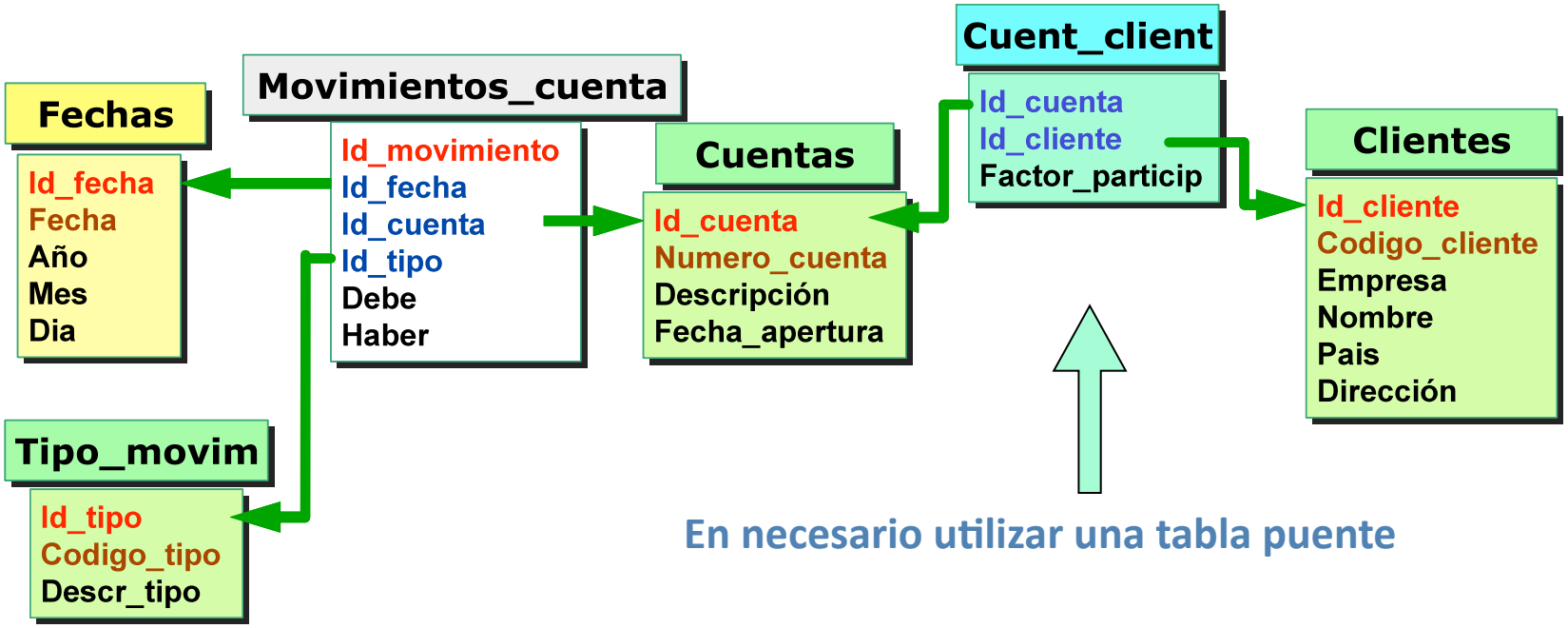
- Si en una tabla de hechos se hace referencia varias veces a una misma dimensión con diferentes significados, ésta desempeña diferentes roles
- Ejemplos:
  - Ciudad de origen – ciudad de destino
  - Fecha de venta – fecha de entrega – fecha de devolución
  - Empleado que decide – empleado que tramita
- Para diferenciar cada uno de los roles puede resultar conveniente crear vistas sobre las correspondientes tablas de dimensión



Relaciones n a m

En ocasiones puede darse el caso de que a una fila de la tabla de hechos le pueda corresponder un número variable de filas de una dimensión (además de lo normal, una fila de dimensión que pueda tener asociadas varias filas en la tabla de hechos)

Ejemplo: movimientos de cuentas que puedan ser más de un cliente



JUNK DIMENSIÓN

- En ocasiones puede ser necesario considerar en la tabla de hechos atributos, generalmente flags, que no parecen organizarse de manera coherente para conformar una dimensión (pocos datos)
  - Ej.: tipo de pago (crédito o débito), tarjeta con comisión o sin ella, venta nacional o internacional, …
- Soluciones posibles son:
  - Dejar los atributos en la tabla de hechos (prob. espacio usado)
  - Hacer dimensiones separadas para cada atributo (prob. nº de dimensiones que aparecen)
  - Quitar directamente estos atributos y crear junk dimension
- Junk dimensión, dimensión que combina todos los valores posibles de este tipo de atributos.
  - El proceso ETL es más complejo

Dimensión degenerada

- La mayoría de los hechos de una tabla de hechos están alrededor de un documento de control: nº de factura, nº de pedido, …
- Normalmente se trata de identificadores del sistema operacional (útil para ETL)
- Estos datos dan lugar a dimensiones vacías y por ello generalmente se añaden a la tabla de hechos donde el nivel de detalle coincide con el del documento de control

Ventas
Id_venta
Id_fecha
Id_articulo
Id_cliente
Id_promocion
Numero_pedido
Unidades
Total
Coste
Beneficio

Cambios lentos en los atributos de las dimensiones

- **TIPO 1:** Actualizar directamente el atributo de la dimensión, con lo que no se almacena la historia. Esta estrategia es interesante únicamente para realizar correcciones.

- **TIPO 2:** Añadir un nuevo registro en la tabla de dimensión cada vez que los atributos cambien. Esta opción permite seguir la historia aunque presenta cierta dificultad al navegar por los datos. Particiona la historia.

- **TIPO 3:** Crear un atributo que recoja el dato anterior. Dos vistas del mismo dato.

- **TIPO 1 + 2 + 3**: combina todas. Permite seguir el historial de cambios al tiempo que realizar las agregaciones

Cambios lentos en los atributos de las dimensiones

El dpto oficinas se hace responsable de las mesas, que hasta ahora gestionaba el dpto mobiliario

Articulos

Id_articulo
Codigo_articulo
Descripción
Dpto

Tipo 1

Id	Cód.	Desc.	Dpto
1	Q12	mesa	oficinas

Tipo 2

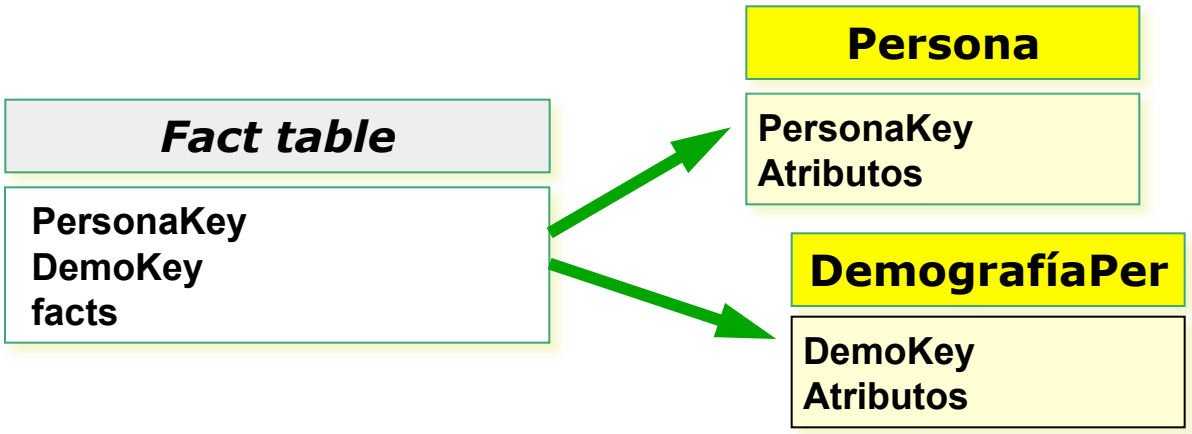
Id	Cód.	Desc.	Dpto
1	Q12	mesa	mobiliario
2	Q12	mesa	oficinas

Tipo 3

Id	Cód.	Desc.	DptoAct	DptoAnterior
1	Q12	mesa	oficinas	mobiliario

Seguimiento de cambios en dimensiones grandes

- Atributos que cambian frecuentemente a una minidimensión. Por ejemplo, para el caso de personas, crear la minidimensión con datos demográficos (sexo, rango edad, nº hijos e ingresos)
- Las minidimensiones se diseñan para tener un conjunto limitado de valores
- La tabla de hechos hace referencia a ambas dimensiones



Extensión OLAP al estándar SQL

SQL99 OLAP SQL EXTENSIONS

GROUP BY <grouping specification>

<grouping specification>::=

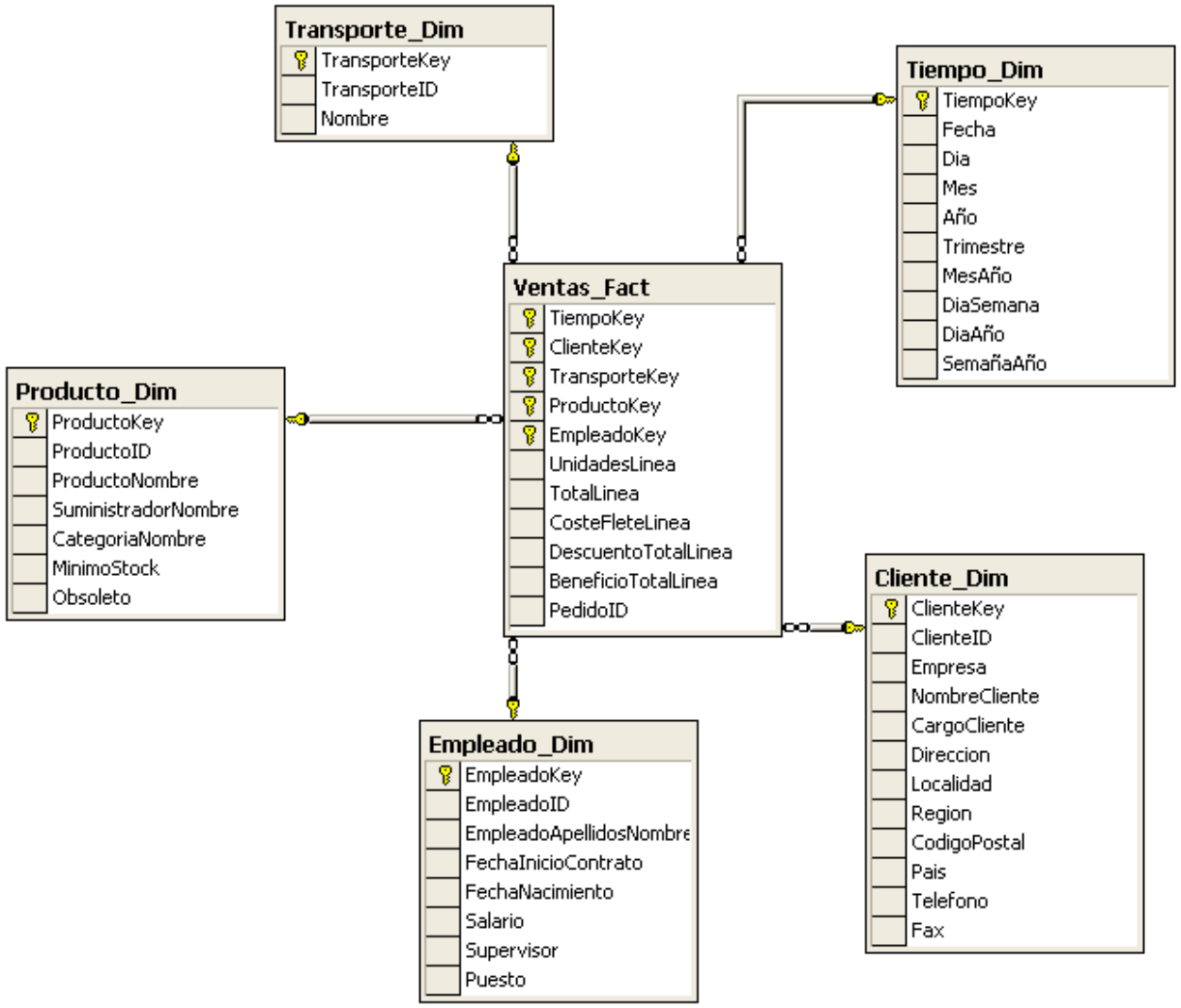
<grouping column reference list> |

ROLLUP (<grouping column reference list>) |

CUBE (<grouping column reference list>) |

GROUPING SETS (<grouping set list>) | ()

BD EJEMPLO


Transporte_Dim
TransporteKey
TransporteID
Nombre
Tiempo_Dim
TiempoKey
Fecha
Dia
Mes
Año
Trimestre
MesAño
DiaSemana
DiaAño
SemañaAño
Ventas_Fact
TiempoKey
ClienteKey
TransporteKey
ProductoKey
EmpleadoKey
UnidadesLinea
TotalLinea
CosteFleteLinea
DescuentoTotalLinea
BeneficioTotalLinea
PedidoID
Producto_Dim
ProductoKey
ProductoID
ProductoNombre
SuministradorNombre
CategoriaNombre
MinimoStock
Obsoleto
Cliente_Dim
ClienteKey
ClienteID
Empresa
NombreCliente
CargoCliente
Direccion
Localidad
Region
CodigoPostal
Pais
Telefono
Fax
Empleado_Dim
EmpleadoKey
EmpleadoID
EmpleadoApellidosNombre
FechaInicioContrato
FechaNacimiento
Salario
Supervisor
Puesto

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

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

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
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

`CUBE(a, b, c)` is equivalent to	`GROUPING SETS ((a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), ())`
`ROLLUP(a, b,c)` is equivalent to	`GROUPING SETS ((a, b, c), (a, b),(a), ())`

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:
  - RANK () OVER
  - DENSE_RANK () OVER
  - PERCENT_RANK () OVER
  - CUME_DIST () OVER
  - ROW_NUMBER () OVER

Funciones de clasificación

Funciones de distribución

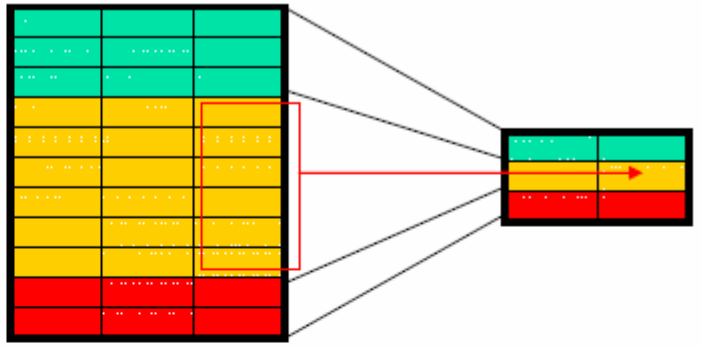
Función de numeración de filas

- 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

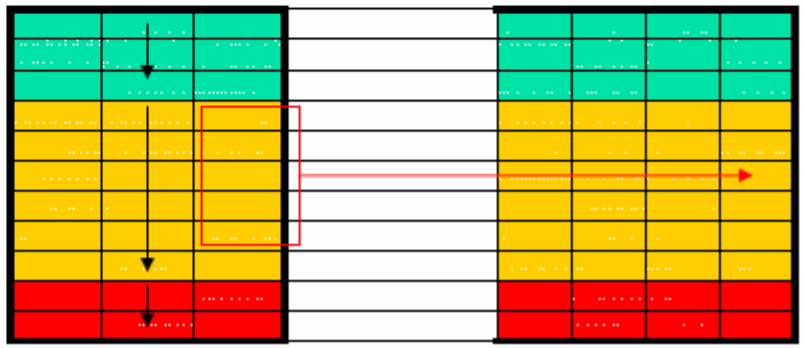
- Set functions (aggregate functions)

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



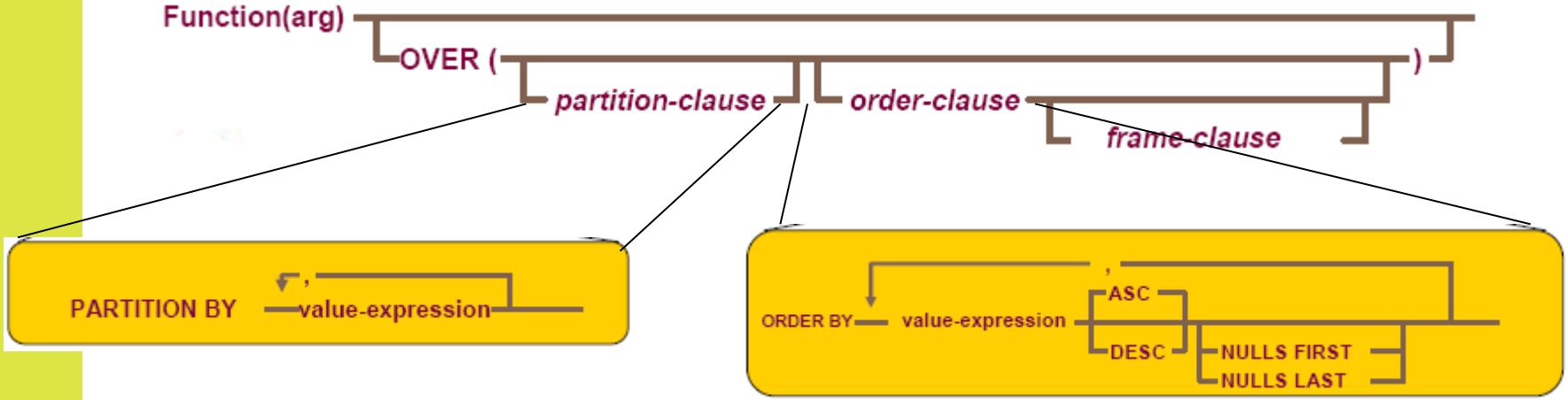
- Windowed Table Functions (tuple-based aggregation)

```
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



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 según su 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 filas iguales 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)

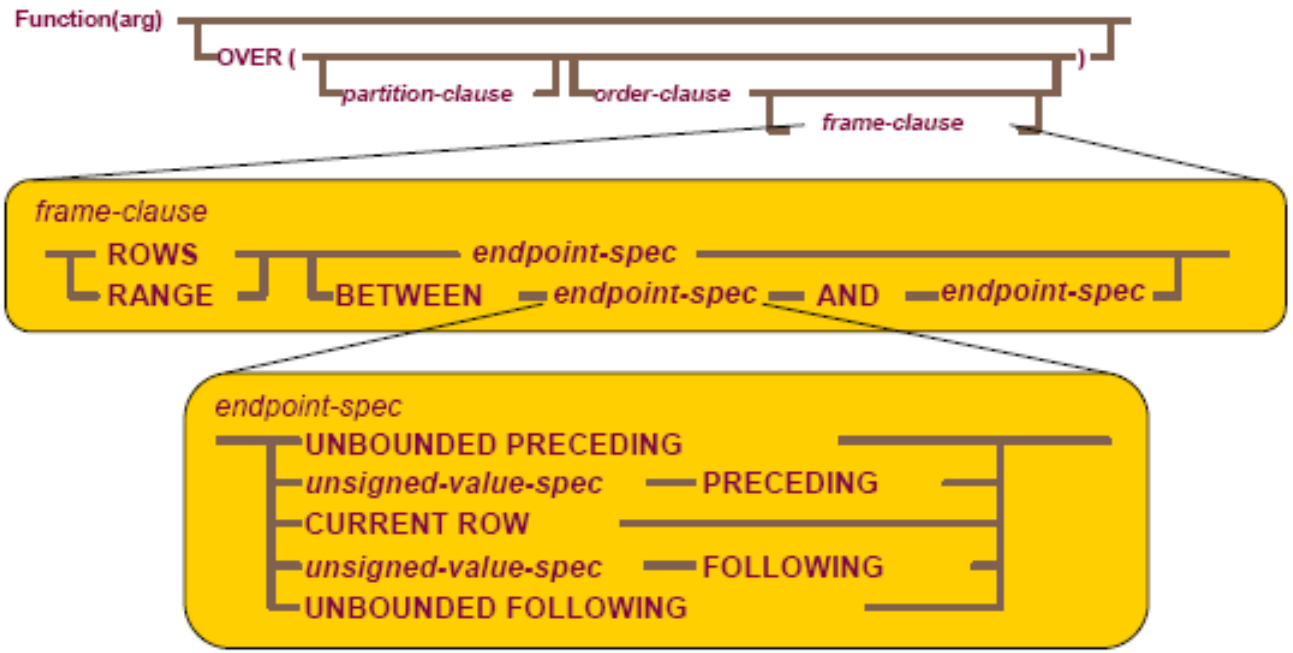
- Obtener los ingresos de ventas por trimestre y acumulado por año

```
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