Diseño lógico relacional de DW

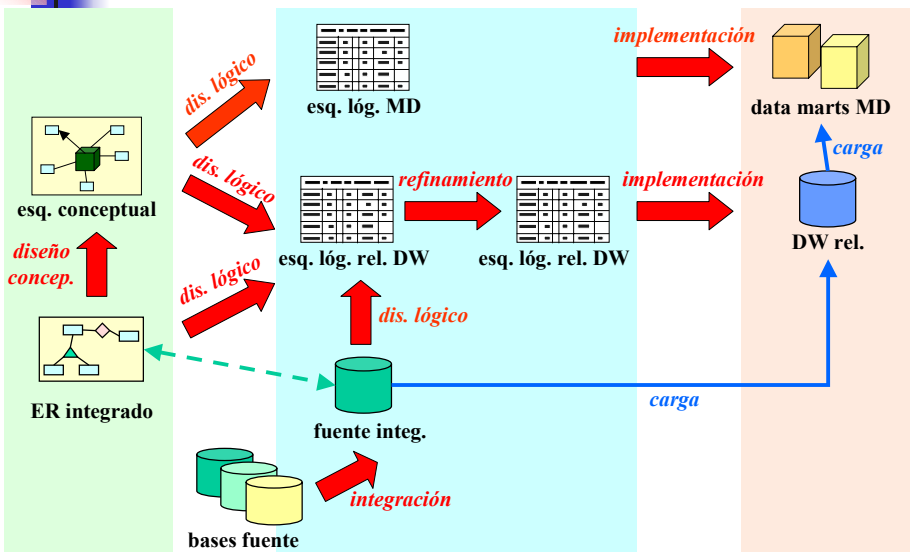
- **Metodologías**
  - Pasaje / traducción del esquema conceptual multidimensional al esquema lógico (ya visto)
  - Diseño a partir de las fuentes
    - Partiendo del esquema ***conceptual*** integrado de las fuentes (MER)
    - Partiendo del esquema ***lógico*** integrado de las fuentes (Modelo Relacional)
- **Refinamientos**
  - Estudio de problemas frecuentes y sus soluciones

Diseño Lógico Relacional

- **Vimos dos grandes enfoques**
  - Traducción de un esquema conceptual multidimensional.
  - Diseño a partir de esquema conceptual de las fuentes.
- **Limitación de ambos enfoques**
  - Se enfatiza poco la expresión de la correspondencia entre esquema logico del DW y esquema lógico de las fuentes.

➡ Técnica basada en operaciones de transformación de esquemas.

Proceso de Diseño



Diseño Lógico Relacional

- **Limitaciones de los enfoques vistos**
  - No se obtienen correspondencias entre esquemas lógicos de DW y fuente.
  - Pocas posibilidades de refinamiento (u optimización) del esquema lógico del DW.
- **Primitivas de transformación de esquemas**
  - Desarrollado en el InCo. Tesis de Maestría. Adriana Marotta, 2000.
  - Una herramienta de ayuda para el diseño lógico de Data Warehouses

Objetivo y enfoque

- **Objetivo**
  - Herramienta para diseñar esquemas de DW a través de transformaciones, permitiendo obtener:
    - la *traza* del diseño
    - un esquema lógico adecuado y optimizado para DW
- **Enfoque: conj. de transformaciones de esquema (primitivas)**
  - Abstraen y materializan técnicas de diseño de DW
  - Se comportan como"building-blocks" de diseño
  - Se apoyan en un mecanismo de control de consistencia del esquema de DW

Modelo utilizado (1)

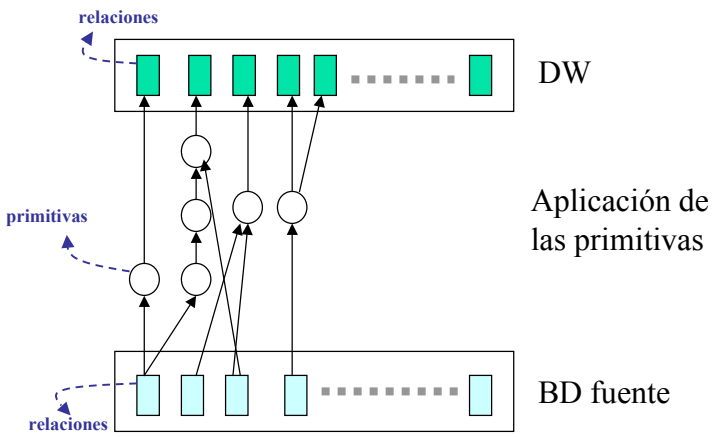
- **Modelo Relacional + Definición de conjuntos de elementos del M. Relacional**
- **Conjuntos de elementos del M. Relacional**
  - Relaciones
    - Rel, Rel_D, Rel_C, Rel_M, Rel_J, Rel_H
    - $Rel_M \subset Rel_C$, $Rel_J \subset Rel_D$, $Rel_H \subset Rel_C \cup Rel_D$
  - Atributos
    - $Att(R)$, $Att_M(R)$, $Att_D(R)$, $Att_C(R)$, $Att_K(R)$, $Att_{FK}(R)$

Modelo utilizado (2)

- **Invariantes de esquema de DW**
  - Propiedades que un esquema relacional de DW debe satisfacer para ser consistente.
  - Relativas a:
    - Integridad referencial
    - Jerarquías
    - Relaciones de Historia
    - Relaciones de Medida
  - Ej.:

    "If a measure relation has an attribute from some dimension relation, then it must have a foreign key relative to this relation."

Proceso de transformación



El conjunto de primitivas

High-Level Transformation		Description
T 1	**Identity**	Given a relation, it generates another that is exactly the same as the source one.
T 2	**Data Filter**	Given a source relation, it generates another one where only some attributes are preserved. Its goal is to eliminate purely operational attributes.
T 3	**Temporalization**	It adds an element of time to the set of attributes of a relation.
T 4	**Key Generalization ***	These transformations generalize the primary key of a dimension relation, so that more than one tuple of each element of the relation can be stored.
T 5	**Foreign Key Update**	Through this transformation, a foreign key and its references can be changed in a relation. This is useful when primary keys are modified.
T 6	**DD-Adding ***	The transformations of this group add to a relation an attribute that is derived from others.
T 7	**Attribute Adding**	It adds attributes to a dimension relation. It should be useful for maintaining in the same tuple more than one version of an attribute.
T 8	**Hierarchy Roll Up**	This transformation does the roll up by one of the attributes of a relation following a hierarchy. Besides, it can generate another hierarchy relation with the corresponding grain.
T 9	**Aggregate Generation**	Given a measure relation, it generates another measure relation, where data are resumed (or grouped) by a given set of attributes.
T 10	**Data Array Creation**	Given a relation that contains a measure attribute and an attribute that represents a pre-determined set of values, it generates a relation with a data array structure.
T 11	**Partition by Stability ***	These transformations partition a relation, in order to organize its history data storage. Vertical Partition or Horizontal Partition can be applied, depending on the design criterion used.
T 12	**Hierarchy Generation ***	This is a family of transformations that generate hierarchy relations, having as input, relations that include a hierarchy or a part of one.
T 13	**Minidimension Break off**	This transformation eliminates a set of attributes from a dimension relation, constructing a new relation with them.
T 14	**New Dimension Crossing**	This transformation allows to materialize a dimension crossing in a new relation.

Especificación de una primitiva

T9 - Aggregate Generation

ENTRADA

- esquema origen : $R(A_1, ..., A_n) \in Rel_M$
- Z conjunto de atributos / card(Z) = k (medidas)
- $\{e_1, ..., e_k\}$, expresiones de agregaciones
- $Y / Y \subset \{A_1, ..., A_n\} \wedge Y \subset (Att_D(R) \cup Att_M(R))$(a eliminarse)

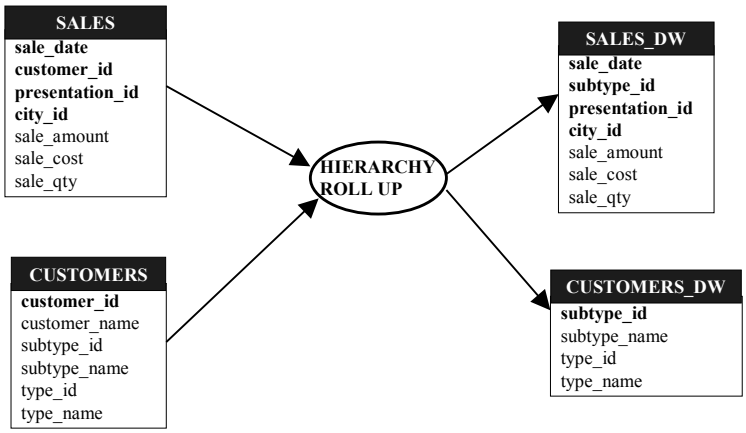
ESQUEMA RESULTADO

- $R'(A'_1, ..., A'_m) \in Rel_M$ /
  $\{ A'_1, ..., A'_m \} = \{ A_1, ..., A_n \} - Y \cup Z$

TRANSFORMACION DE LOS DATOS

```
r' =   select ( { A'1, ...., A'm } - Z ) ∪ { e1, ...., ek }
       from R
       group by { A'1, ...., A'm } - Z
```

Aplicación de una primitiva



Reglas de Consistencia

- **Invariantes de esquema ⇒ Reglas para la aplicación de primitivas (E.C.A.)**
- **Ejemplo. Regla para integridad referencial:**

ON APPLICATION OF:

Temporalization (agregando atr. de tiempo a la clave)

o

Key Generalization

a R, donde X = clave vieja and Y = clave nueva

APPLY: ***Foreign Key Update***

a todos los R_i / $Att_{FK}(R_i,R)$ = X, obteniendo:

$Att_{FK}(R_i,R)$ = Y

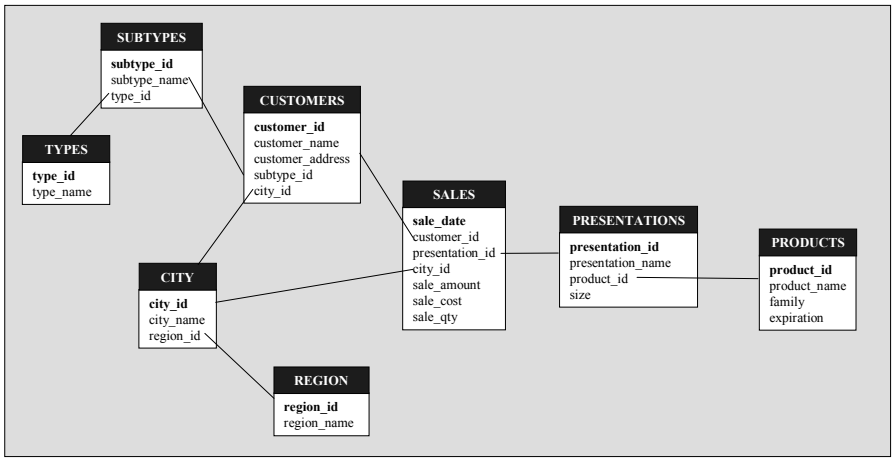
Estrategias de Aplicación

- **Versionamiento**
  - Alternativas de diseño para poder mantener la historia de los objetos de una dimensión
- **Agregaciones y cruzamientos de datos**
  - Como construir nuevos cruzamientos y agrupaciones de datos
- **Identificación e independización de jerarquías**
  - Mecanismo para extraer jerarquías comunes a varias dimensiones y alternativas de diseño
- **Datos calculados**
  - Como construir atributos cuyos valores son calculados a partir de otros. Se analizan distintos tipos de cálculos.

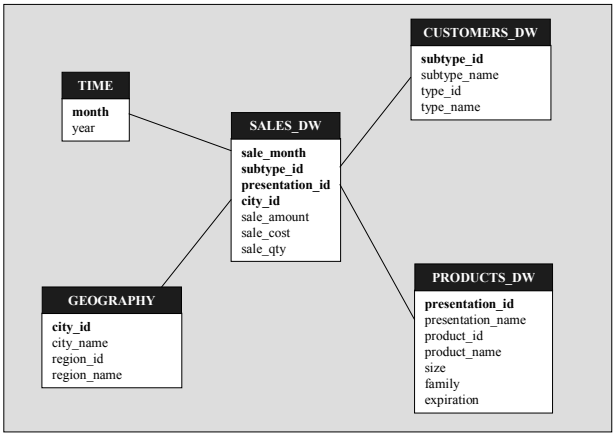
Un Ejemplo

- **Esquema logico de la fuente**
- **Esquema de DW deseado**
- **Transformaciones para generar esquema deseado**
- **Refinamiento del diseño**
- **Esquema final del DW**
- **Traza generada**

Esquema lógico de la fuente



Esquema de DW deseado



Generar esquema deseado

- **Pasos a seguir:**
  - Desnormalizar las relaciones que corresponden a las dimensiones, generando una relacion para cada dimension del esquema deseado.
  - Eliminar atributos innecesarios en las dimensiones.
  - Generar la dimension *Time.*
  - Generar la tabla de hechos *Sales_DW* con la granularidad deseada (*subtype* para *Customer* y *month* para *Time*)

Desnormalizar, generar dims.

- **T6.2 DD-Adding N-1**

T6.2 a Presentations y Products

PRODUCTS_DW
presentation_id
presentation_name
product_id
product_name
size
family
expiration

T6.2 a Customers, Subtypes y Types

CUSTOMERS_DW_01
customer_id
customer_name
customer_address
subtype_id
city_id
subtype_name
type_id
type_name

T6.2 a City y Region

GEOGRAPHY
city_id
city_name
region_id
region_name

Eliminar atributos innecesarios

- **T2 Data Filter**

T2 a Customers_DW_01

CUSTOMERS_DW_02
customer_id
subtype_id
subtype_name
type_id
type_name

Generar la dimension *Time*

- **T12.1 De-Normalized Hierarchy Generation**

T12.1 a Sales

TIME_01
date

- **T6.1 DD-Adding 1-1**

T6.1 a Time_01

TIME_02
date
month

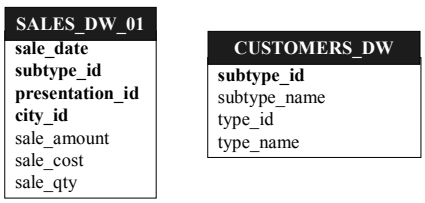
T6.1 a Time_02

TIME_03
date
month
year

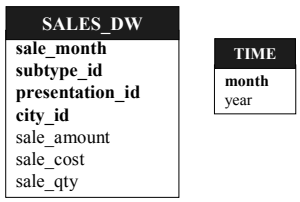
Generar *Sales_DW*

- **T8 Hierarchy Roll-Up**

T8 a Sales y Customers_DW_02



T8 a Sales_DW_01 y Time_03



Refinamiento del diseño

- **Ya construimos el esquema de DW diseñado previamente.**
- **Detectamos 2 posibles mejoras al esquema:**
  - La dimensión *Product* es una "slowly changing dimension" [Kim96,97], y nos interesa su historia ⇨ relación separada para mantener la historia de *Product*.
  - Sería mas eficiente tener la cantidad de clientes que pertenecen a cada ciudad junto a cada una de estas ⇨ agregar a la dimensión *Geography* un atributo calculado con la cantidad de clientes.
- **Nota:**
  - Este tipo de decisiones se toma estudiando y comparando los costos de consultas (considerando frecuencias) y mantenimiento.

Historia de *Products*

- **T11.2 Horizontal Partition**

T11.2 a Products_DW

PRODUCTS_DW
presentation_id
presentation_name
product_id
product_name
size
family
expiration

PRODUCTS_DW_HIS_01
presentation_id
presentation_name
product_id
product_name
size
family
expiration

Historia de *Products*

- **T3 Temporalization**

T3 a Products_DW_His_01

PRODUCTS_DW_HIS
presentation_id
change_date
presentation_name
product_id
product_name
size
family
expiration

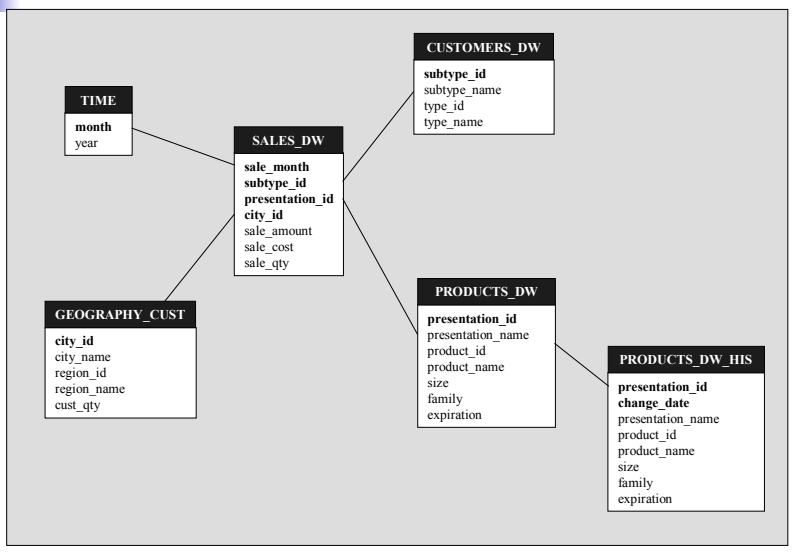
Agregar atributo calculado

- **T6.3 DD-Adding N-N**

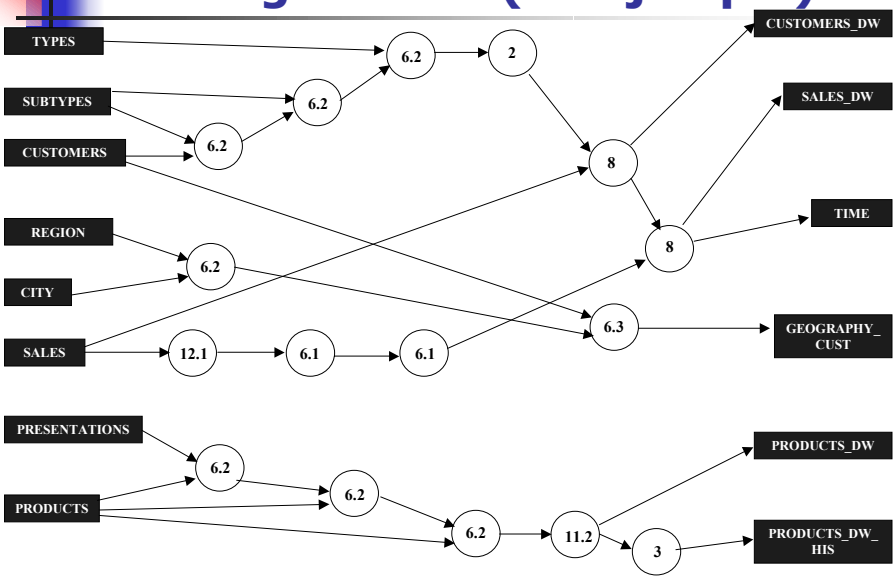
T6.3 a Geography y Customers

GEOGRAPHY_CUST
city_id
city_name
region_id
region_name
cust_qty

Esquema final del DW



Traza generada (fin ejemplo)



La traza del diseño

- **Especificación textual y gráfica**
- **Provee**
  - Trazabilidad del proceso de diseño
    - Documentación
    - Reutilización
  - Mapeo entre elementos del esquema fuente y del esquema de DW
    - Propagación de la evolución del esquema fuente
    - Generación de procesos de carga del DW
    - Detección de errores en los datos

Ejercicios

- **1) Sub-esquema:**

  Clientes (ci, nombre, dirección, telefono, cod-plan, cuota-act)

  Planes (cod-plan, cuota, monto)

  Descuentos (cod-plan, monto-desc)

  Se necesita que exista un atributo en la tabla de Clientes con el monto que debe pagar de la cuota actual. Aplicar las primitivas de transformación.

Ejercicios

- **2) Sub-esquema:**

  Producto-mes (cod-prod, mes, cantidad-vendida)

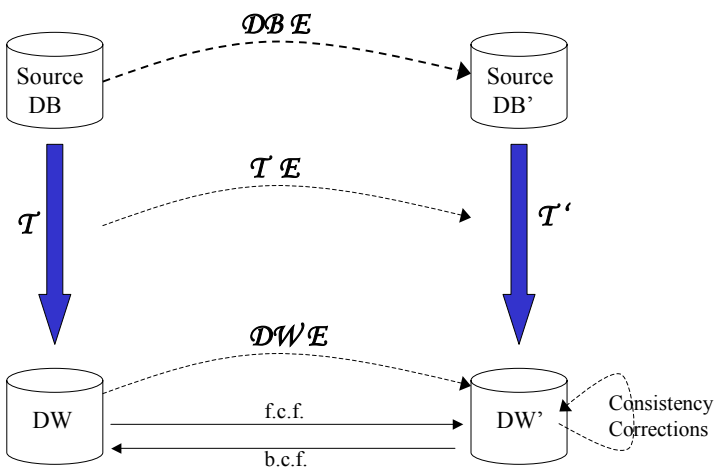
  Producto (cod-prod, desc-prod, tamaño, cod-prov)

  Se va a consultar muy frecuentemente la cantidad vendida por proveedor por mes, sin importar el producto. Obtener un sub-esquema adecuado para este requerimiento, aplicando las primitivas de transformación.

Evolución de esquema fuente

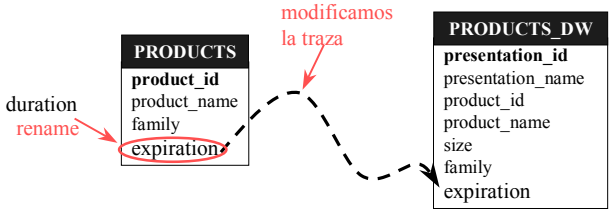
- Escenario de evolución
- Ejemplos
- Problemática

Escenario de evolución



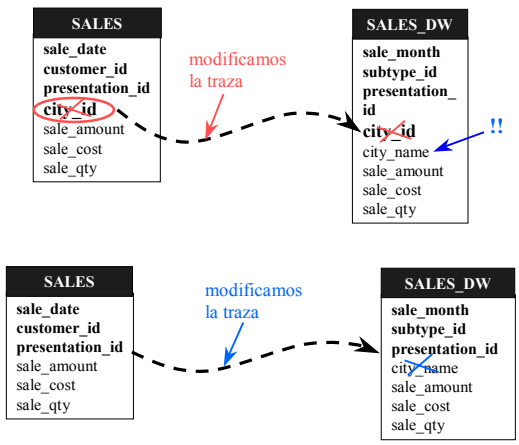
Ejemplos

- **Se renombra un atributo en el esquema fuente**



Ejemplos

- **Se borra un atributo en el esquema fuente**



Problemática

- **Cuando evoluciona el esquema fuente hay que resolver 2 problemas:**
  - Determinación de los cambios a realizarse en el esquema de DW
    - Influyen:
      - Cual fue el cambio ocurrido en el esquema fuente
      - QUE elementos del DW dependen del elemento modificado y COMO dependen
      - Chequeo de consistencia del nuevo esquema de DW
  - Aplicación de la evolución al DW
    - Generar nueva version del esquema del DW
    - Definir las *funciones de conversion de instancia*

Resumen

- **Herramienta de ayuda al diseño de DW que incluye:**
  - Conjunto de transformaciones de esquema
  - Invariantes de esquema de DW y Reglas de Consistencia para la aplicación de las transformaciones
  - Estrategias de aplicación de las transformaciones
  - Generacion automática de la traza del diseño
- **Herramienta de ayuda para propagación de la evolución del esquema fuente, al esquema del DW**