"Diseño y mantenimiento de Data Warehouses a través de transformación de esquemas"

Adriana Marotta

Proyecto CSIC

InCo - Facultad de Ingeniería

Contexto

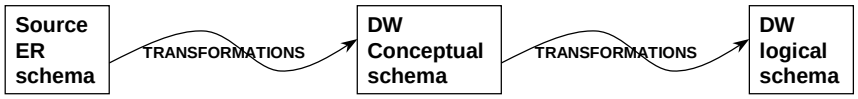
- **Data Warehouse: BD que almacena información para satisfacer requerimientos de toma de decisiones.**
  - **Requerimientos orientados a ver la informacion en forma multidimensional**
  - **Información del DW**
    - » **extraida de bds operacionales**
    - » **resultado de transformacion, limpieza, integración**
  - **Acceso y mantenimiento de datos en el DW**
    - » **Se considera *solo-lectura*. El mantenimiento no se hace via sistema OLTP, sino en forma "batch".**
- **Modelos de datos**
  - **Fuente y DW: en Modelo Relacional**

Metodologias de diseño de DW existentes (1)

- **Algunas: [Bal98][Gol98]**

- **Otras: [Kor99]**

- **Otras: [Cab98]**



Metodologías de diseño de DW existentes (2)

- **En general, todas son pobres en:**
  - **soporte para la *trazabilidad* del diseño**
    - » **No se genera explícitamente la traza de la transformación realizada**
    - » **Traza**
      - **gestión y reutilización del proceso de diseño**
      - **generación de procesos de carga**
      - **manejo de evolución de esquemas fuentes**
      - **detección de errores en los datos**
  - **capacidad para lograr un diseño lógico refinado (optimizado) del esquema del DW**
    - » **Construcciones como:**
      - **manejo de datos históricos (versionamiento de dimensiones)**
      - **datos pre-calculados**
      - **generalización de claves**

Objetivo y enfoque

- **Objetivos**
  - **Brindar una herramienta que permita diseñar esquemas de DW a traves de transformaciones, permitiendo obtener:**
    - » **un esquema logico adecuado y optimizado para DW**
    - » **la traza del diseño**
- **Enfoque: conj. de transformaciones de esquema (primitivas)**
  - **Primitivas:**
    - » **Abstraen y materializan técnicas de diseño de DW**
    - » **Proporcionan una traza del diseño realizado**
    - » **Se comportan como"building-blocks" de diseño**
    - » **Se apoyan en un mecanismo de control de consistencia del esquema de DW**

Plan

- **Diseño mediante primitivas de transformacion de esquemas**
  - **Modelo utilizado**
  - **Transformaciones de esquema**
  - **Un ejemplo**
  - **La traza**
- **Conclusiones**

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**
    - » $Att(R), Att_M(R), Att_D(R), Att_C(R), Att_K(R), Att_{FK}(R)$

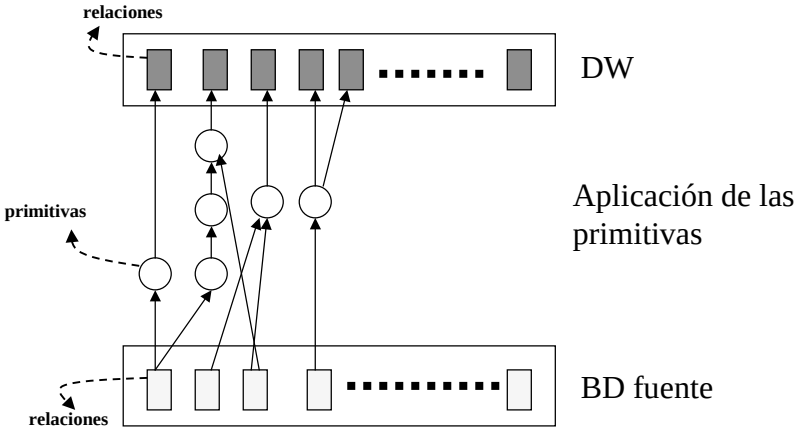
Modelo utilizado (2)

- **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**

Transformaciones de esquema

- **Arquitectura de la transformación**
- **El conjunto de las primitivas**
- **Especificación de una primitiva**
- **Reglas de consistencia y Estrategias de aplicación**

Arquitectura de la transformación



El conjunto de transformaciones

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**	It allows materialising a dimension data crossing in a new relation. It also is useful to simply de-normalize relations, which improves queries performance.

Especificacion de una transformacion

T 9 - Aggregate Generation

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

- **ESQUEMA RESULTADO**
  - » $R' (A'_1, ..., A'_m) \in Rel_C$ / $\{ A'_1, ..., A'_m \} = \{ A_1, ..., A_n \} - Y \cup Z$

- **TRANSFORMACION DE LOS DATOS**
  - » $r' =$ select ($\{ A'_1, ..., A'_m \} - Z$) $\cup \{ e_1, ..., e_k \}$
    from R
    group by $\{ A'_1, ..., A'_m \} - Z$

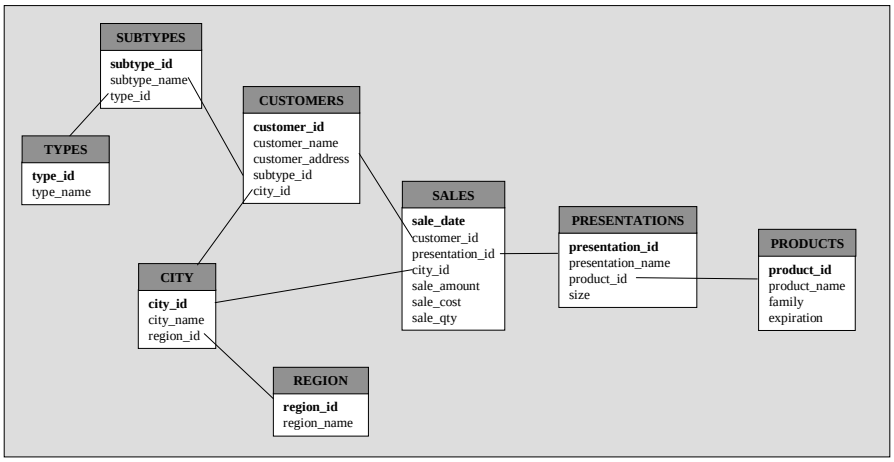
Reglas de Consistencia y Estrategias de Aplicación

- **Invariantes de esquema ⇒ Reglas para la aplicación de primitivas (E.C.A.)**
  - **Actualizacion de claves externas (foraneas)**
  - **Relaciones de Medida**
  - **Actualizacion de Relaciones de Historia**
- **Estrategias de Aplicación**
  - **Versionamiento**
  - **Agregaciones y cruzamientos de datos**
  - **Identificación e independización de jerarquías**
  - **Datos calculados**

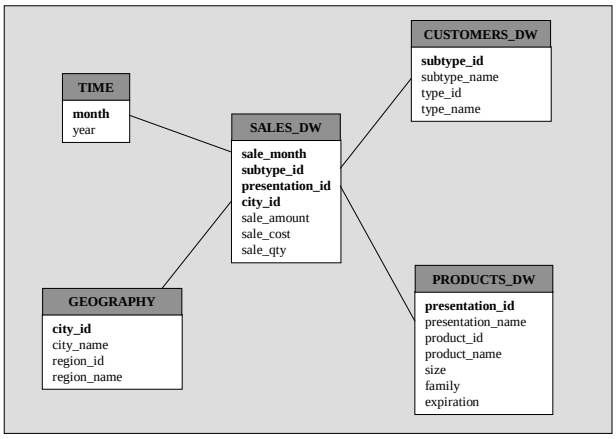
Un Ejemplo

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

Esquema lógico de la fuente



Esquema de DW deseado



Este esquema fue diseñado por medio de las metodologías nombradas anteriormente

Transformaciones para generar esquema deseado

- **Pasos que vamos 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 para generar dimensiones

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

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

Refinamiento del diseño

- **Ya construimos el esquema de DW diseñado previamente.**
- **Detectamos 2 posibles mejoras al esquema:**
  - **La dimension *Product* es una "slowly changing dimension" [Kim96,97], y nos interesa su historia ⇒ relacion separada para mantener la historia de *Product.***
  - **Seria mas eficiente tener la cantidad de clientes que pertenecen a cada ciudad junto a cada una de estas ⇒ agregar a la dimension *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.**

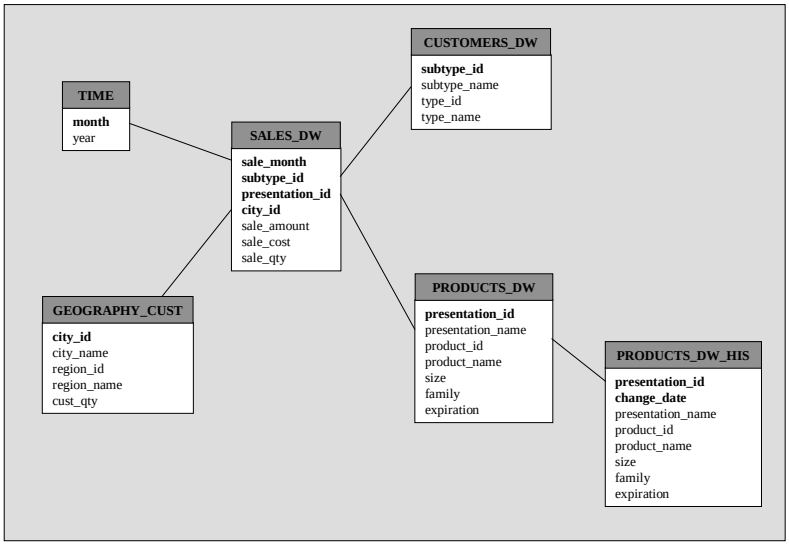
Agregar atributo calculado

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

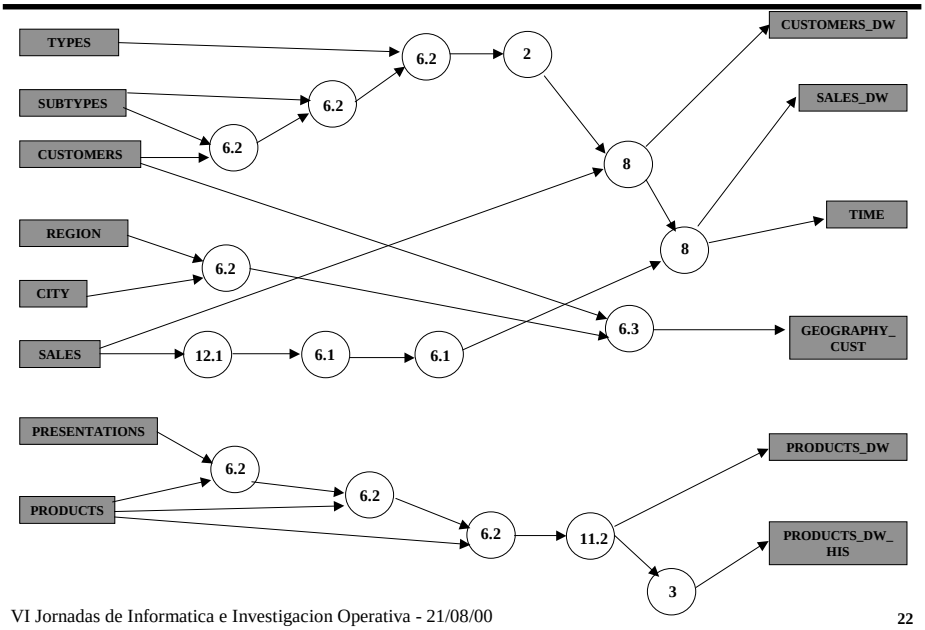
T6.3 a Geography y Customers

GEOGRAPHY_CUST
city_id
city_name
region_id
region_name
cust_qty

Esquema final del DW



La Traza



Conclusiones

- **Aportes**
  - **Herramienta de ayuda al diseño de DW**
- **Implementación**
  - **Herramienta de diseño: ya implementada (Taller 5 y tesis maestria)**
- **Trabajo en curso**
  - **Aplicación automatica de las primitivas a partir de los requerimientos y el conocimiento de las bases fuentes**
- **Trabajo futuro**
  - **Carga y mantenimiento de datos**
  - **Evolución generada por cambios en los requerimientos del DW**