"Modeling ETL Activities as Graphs" (Vassiliadis, Simitsis, Skiadopoulos) y posicionamiento de la propuesta de "Primitivas de Transformación de Esquemas"

"Modeling ETL Activities…"

- Introduccion
- El "Architecture Graph"
- Explotación del Architecture graph
- ARKTOS
- Conclusiones

Introduccion

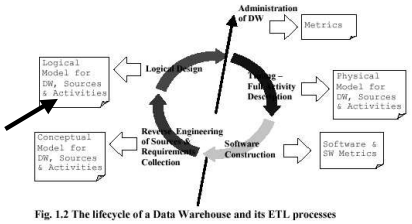


Fig. 1.2 The lifecycle of a Data Warehouse and its ETL processes

El trabajo se centra en el diseño lógico del escenario de ETL de un DW.

Introduccion

- Estado del arte:
  - Herramientas ETL
  - Ajax
  - Articulo de Workflow para ETL, de Mokrane
  - Otros trabajos de Vassiliadis sobre procesos de DW, metadata y calidad
- Pero…
  - Estos enfoques no han considerado en profundidad la *estructura interna de las actividades de ETL.*

Introduccion

- Contribuciones
  - Definición de un modelo lógico formal como abstracción lógica de procesos ETL
    - Incluye depositos de datos, actividades ETL y sus componentes (son todas entidades)
  - Reducción del modelo a un grafo, *Architecture Graph*
    - Las *entidades* anteriores son *nodos*, y las *relaciones* (4 tipos) entre ellas son las *aristas*.
  - Explotación del *Architecture Graph*
    - Transformaciones del grafo (zoom in/out)
    - Importance metrics: *dependence* y *resposibility*

El "Architecture Graph"

- Definiciones previas
- Escenario ETL
- Ejemplo
- Architecture Graph
- Ejemplo (cont.)

Definiciones previas

- Nodos
  - Tipos de datos
  - Tipos de funciones
  - Constantes
  - Atributos
  - Actividades
  - Conj. de registros
  - Funciones

  } **terminos**

  Esquema: lista finita de atributos
- Aristas (relaciones entre nodos)
  - Part-of
  - Instance-of
  - Proveedor
  - Regulador
  - Proveedor derivado

Nodos

- **Data Type** - nombre, dominio
- **Attributes** - nombre, tipo de datos
- **RecordSet** - nombre, esquema, extension
- **Function Type** - nombre, tipos de datos de parametros, tipo de datos de resultado.
- **Function** - instancia de un tipo de funcion
- **Activities**
  - Terminologia de [WfMC98] para procesos/programas
  - Abstracciones lógicas representando parte/modulos completos de código
  - Usan sentencias SQL para representar la semantica de las actividades

Aristas

- **Part-of -** Relaciona atributos y parámetros con actividades, conj. de registros o funciones
- **Instance-of -** Relacion entre data/function type y sus instancias
- **Provider -** Relaciones 1:N entre atributos (provider-consumer). Se define con nombre, mapping.
- **Regulator -** Relaciones entre los parametros de las actividades y los terminos que las instancian (populate).
- **Derived provider -** Caso particular de provider, cuando output attributes se derivan de la combinacion de input attributes y parametros.

Activities

- Actividad, descripta formalmente por:
  - *Name*
  - *Input Schema*
    - Recibe los datos del data provider
  - *Output Schema*
    - "placeholder" para las tuplas que pasan el chequeo de la activ.
  - *Rejections Schema*
    - "placeholder" para las tuplas que no pasan el chequeo de la act.
  - *Parameter list* (esquema, atributo, funcion o cte.)
  - *Output Operational Semantics* (SQL)
  - *Rejection Operational Semantics* (por def. la neg. del ant.)

Escenario ETL

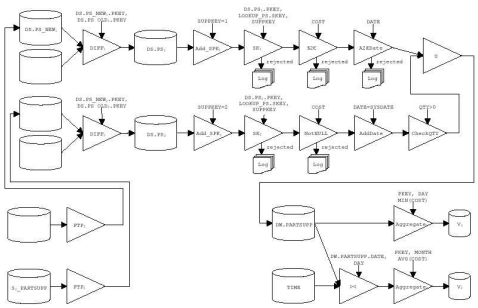
- Consiste de
  - Name
  - Activities
  - RecordSets (fuente)
  - Targets
    - tablas del DW que seran pobladas por las actividades
  - Provider Relationships
    - Relaciones "provider" entre actividades y recordsets del escenario
- Intuitivamente
  - Conj. de actividades presentadas en un grafo en una secuencia de ejecución.

Ejemplo de Escenario

Source	Recordset Name	Recordset Schema
S_1	S_1.PARTSUPP	PKEY, DATE, QTY, COST
S_2	S_2.PARTSUPP	PKEY, QTY, COST
DSA	$DS.PS\_NEW_1$	PKEY, DATE, QTY, COST
	$DS.PS\_OLD_1$	PKEY, DATE, QTY, COST
	$DS.PS_1$	PKEY, DATE, QTY, COST
	$DS.PS\_NEW_2$	PKEY, QTY, COST
	$DS.PS\_OLD_2$	PKEY, QTY, COST
	$DS.PS_2$	PKEY, QTY, COST
DW	DW.PARTSUPP	PKEY, SUPPKEY, DATE, QTY, COST
	LOOKUP_PS	PKEY, SOURCE, SKEY
	V1	PKEY, DAY, MIN_COST
	V2	PKEY, MONTH, AVG_COST
	TIME	DAY, MONTH, YEAR

Fig. 2.2 The schemata of the source databases and of the data warehouse

Ejemplo de Escenario



Architecture Graph

- El Architecture Graph y el escenario
  - El AG contiene todas las actividades y depositos de datos del escenario, junto con sus componentes
  - El AG captura el flujo de datos dentro del ambiente ETL
  - El AG da la informacion de
    - tipos de las entidades
    - regulacion de la ejecucion del escenario

Dominios y notacion

	Entity	Model-specific	Scenario-specific
Built-in	Data Types	$\mathbf{D}^1$	**D**
	Function Types	$\mathbf{F}^1$	**F**
	Constants	$\mathbf{C}^1$	**C**
User-provided	Attributes	$\mathbf{\Omega}^1$	**Ω**
	Functions	$\mathbf{\Phi}^1$	**Φ**
	Schemata	$\mathbf{S}^1$	**S**
	RecordSets	$\mathbf{RS}^2$	**RS**
	Activities	$\mathbf{A}^1$	**A**
	Provider Relationships	$\mathbf{Pr}^2$	**Pr**
	Part-Of Relationships	$\mathbf{Po}^2$	**Po**
	Instance-Of Relationships	$\mathbf{Io}^2$	**Io**
	Regulator Relationships	$\mathbf{Rr}^2$	**Rr**
	Derived Provider Relationships	$\mathbf{Dr}^2$	**Dr**

Fig. 3.1 Formal definition of domains and notation

Notacion grafica (AG)

Data Types	Black ellipsis	integer
Function Types	Black squares	$2€
Constants	Black cycles	1
Attributes	Hollow ellipsoid nodes	PKEY
Part-Of Relationships	Simple edges annotated with diamond*	
Instance-Of Relationships	Dotted arrows (from instance towards the type)	
Regulator Relationships	Dotted edges	

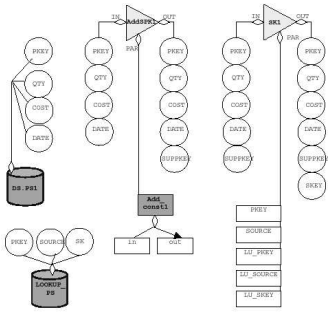
RecordSets	Cylinders	R
Functions	Gray squares	my$2€
Parameters	White squares	rate
Activities	Triangles	SK
Provider Relationships	Bold solid rrows (from provider to consumer)	
Derived Provider Relationships	Bold dotted arrows (from provider to consumer)	

* We annotate the part-of relationship among a function and its return type with a directed edge, to distinguish it from the rest of the parameters.

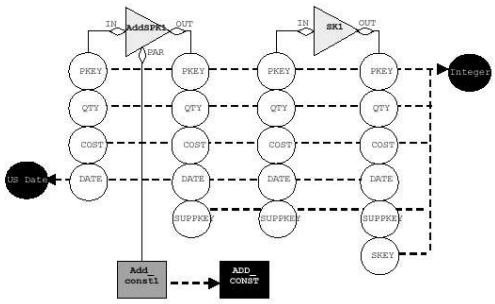
Ejemplo (cont.)

- Activities: *Add_SPK*$_1$ y *SK*$_1$
- Partes del grafo
  - Atributos y relaciones part-of
  - Data types y relaciones instance-of
  - Parametros y relaciones regulator
  - Relaciones provider
  - Relaciones derived provider

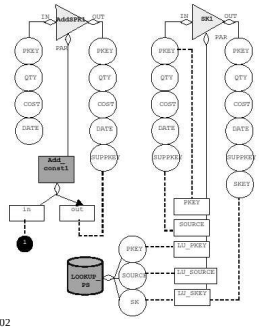
Atributos y relaciones part-of



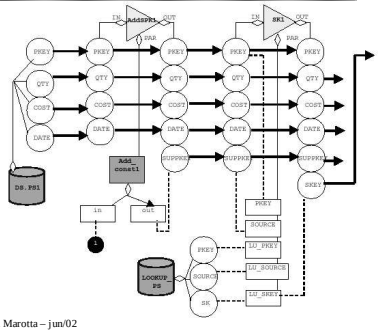
Data types y relaciones instance-of



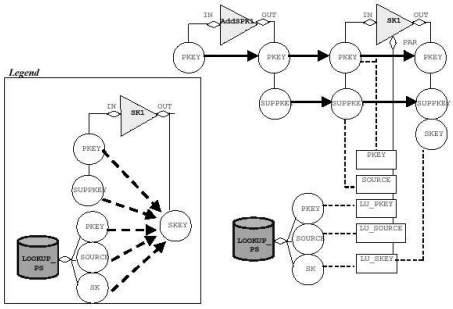
Parametros y relaciones regulator



Relaciones provider



Relaciones derived provider



Explotación del AG

- Graph transformations
  - "zoom-out" para eliminar la sobrecarga de información (gran cantidad de atributos).
  - Subgrafo crítico que incluye solo las entidades necesarias para la poblacion de los "target recordsets" del escenario.
- Importance metrics
  - Se asignan a los nodos para medir cuan crucial es su existencia para la ejecucion del escenario
  - *local dependency, local responsibility, local degree*

Importance metrics

- Local dependency (in-degree)
  - Numero de nodos que deben ser activados para poblarlo
- Local responsibility (out-degree)
  - Cuantos nodos esperan por él para ser activados
- Local degree
  - La suma de los anteriores
- Transitividad
- Estas medidas no son aplicables solo a atributos. Tambien a actividades y recordsets.

ARKTOS - Herramienta

- Objetivos
  - Facilidades graficas y declarativas para la definicion de tareas de limpieza y transformacion de DW
  - Medicion de calidad de datos (factores de calidad)
  - Ejecucion optimizada de secuencias complejas de tareas
- Metamodelo
  - El presentado anteriormente + *Error Type*, *Policy*, *Quality Factors*
- ***Transformation and cleaning primitives***

Primitivas

- Operaciones primitivas para soportar el proceso ETL
- Personalilzadas por el usuario
  - Input, output, data stores, policy, quality factors

Primitive Operation	SQL statement	SEMANTICS clause shortcuts
UNIQUENESS VIOLATION	SELECT * FROM <table> GROUP BY <attribute> HAVING COUNT(*) > 1	<table>.<attribute>
NULL EXISTENCE	SELECT * FROM <table> WHERE <attribute> IS NULL	<table>.<attribute>
DOMAIN MISMATCH*	SELECT * FROM <table> WHERE <attribute> NOT IN <domain specification>	<table>.<attribute> NOT IN <domain specification>
PRIMARY KEY VIOLATION	SELECT * FROM <table> GROUP BY (<attribute_1>,...,<attribute_n>) HAVING COUNT(*) > 1	<table>.(<attribute_1>, ..., <attribute_n>)
REFERENCE VIOLATION	SELECT * FROM <table> WHERE <attribute> NOT IN (SELECT <target_attribute> FROM <target_table>)	<table>.<attribute> NOT IN <target_table>.<target_attribute>
FORMAT MISMATCH**	SELECT APPLY(<reg_exp>,<attribute>) FROM <table> WHERE APPLY(<reg_exp>,<attribute>)	TARGET APPLY(<reg_exp>,<attribute>) SOURCE APPLY(<reg_exp>,<attribute>)
PUSH	Arbitrary SQL query	Arbitrary SQL query

* works for intervals of numbers and strings

** where <reg_exp> is PERL regular expression acting as a formatting function

Definicion de actividades

- Graficamente
  - Paleta con las actividades provistas por ARKTOS
- Declarativamente
  - 2 lenguajes
    - XADL
      - XML-based Activity Definition Language
    - SADL
      - Simple Activity Definition Language

Conclusiones – [VASS02]

- Diseño logico de escenario ETL
- Modelo logico formal y “Architecture Graph”
- “Importance metrics”
- Herramienta ARKTOS para modelar y ejecutar escenarios
- Primitivas para las tareas comunes
- Resultados son parte de un proyecto mas grande
  - Referencian: “Conceptual modeling for ETL Processes”

Trabajo futuro – [VASS02]

- Optimizacion
- Proveer primitivas de transformacion mas ricas
- “It would be nice if the research community could provide formal design guidelines (in the sense of normal forms or extra integrity constraints) for the engineering of ETL processes.”

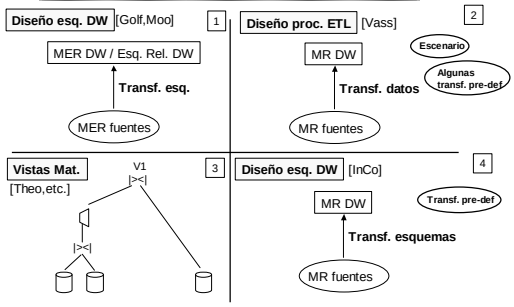
Frases célebres – [VASS02]

- Sobre VM
  - “… viewing the DW as a set of layered, materilized views is a very simplistic view.”
  - “… insufficient to describe the structure and contents of a DW.”
- Sobre evolución
  - “… the interesting problem is how to design the scenario in order to achieve effectiveness, efficiency and tolerance of the impacts of evolution.”
  - “Dependence and responsibility are crucial measures for the engineering of the evolution of the ETL env.”

Posicionamiento de las Primitivas de Transformacion de Esquemas

- Algunos enfoques… y el nuestro
- Como nos podemos posicionar?
- Ejemplo
- Conclusiones ?

Algunos enfoques… y el nuestro



Cómo nos podemos posicionar?

- Diagnóstico
  - Las propuestas 1 y 2 estan totalmente desconectadas.
  - La propuesta de diseño no se preocupa por los procesos ETL y la propuesta para ETL no menciona su relacion con el diseño del esquema de DW, por ejemplo, cómo deduce las actividades a realizar?
  - Las propuestas tipo 1 no resuelven la construccion de estructuras "complejas" de DW (historicos, dimension versioning, datos calculados, etc.), que a la vez son comunes.

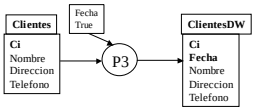
Cómo nos podemos posicionar?

- "Filling the gap between DW schema design and the corresponding ETL processes design"
- Un mecanismo para diseño logico de DW que ofrece 2 ventajas:
  - Facilidades para diseñar estructuras complejas de DW
  - Permite la deduccion semi-automatica de un "Architecture Graph" [Vass].

Ejemplo

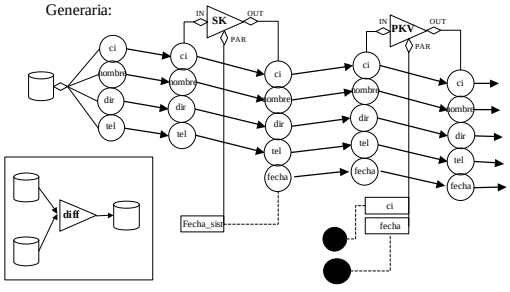
- Una de las estrategias de diseño para Dimension Versioning era aplicar *Temporalization* haciendo que el nuevo atributo forme parte de la clave.

Diseño del esquema:



Ejemplo cont.

Generaria:



Conclusiones ?