Concepción de Sistemas de Información

Instituto de Computación – Facultad de Ingeniería – Universidad de la República

SEMINARIO

Estudio de modelos y técnicas de workflow en vista a la definición de un proceso para la carga y mantenimiento de data warehouses

Presentación día 13 con fecha 23/07/2001

Artículo: Efficient Resumption of Interrupted Warehouse Loads

Expositor: Pablo Morales

Efficient Resumption of Interrupted Warehouse Loads

Wilburt Juan Labio, Janet L. Wiener, Hector Garcia-Molina, Vlad Gorelik

Presentación de una algoritmo (DR) de recuperación ante fallas en el proceso de carga y refresque del data warehouse.

Dos características principales:

1. No impone un overhead en el tiempo de carga normal
2. Utiliza únicamente ciertas propiedades básicas de las transformaciones

Idea general y estrategia

Otros Algoritmos:

- Rehacer toda la carga
- Dividir la entrada en lotes y procesarlos en secuencia
- Tomar "snapshots" periódicos del estado del workflow de carga
- Dividir el workflow en estados (identificarlos) y guardar resultados intermedios

Idea general y estrategia

Algoritmo DR:

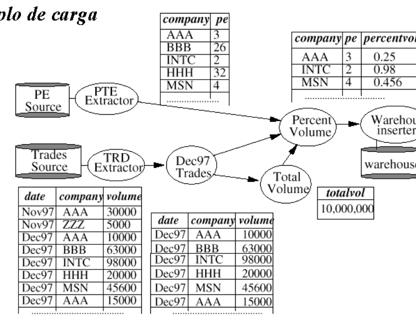
- No hay overhead en el proceso de carga normal
- No es necesario modificar los procesos de transformación
- Quién codifica los elementos de transformación (wrappers, etc) debe declarar si éstos cumplen alguna propiedades simples
- Explota la semántica del workflow de carga para ser selectivo al reasumir luego de una falla

Aportes

- Presentación del marco que describe los procesos de carga normales y con fallas identificando ciertas propiedades
- Presentación del algoritmo DR que filtra tuplas de entrada ASAP
- Conclusiones

Proceso de carga normal

Ejemplo de carga



company	*pe*
AAA	3
BBB	26
INTC	2
HHH	32
MSN	4
.....	

company	*pe*	*percentvol*
AAA	3	0.25
INTC	2	0.98
MSN	4	0.456
.....		

date	*company*	*volume*
Nov97	AAA	30000
Nov97	ZZZ	5000
Dec97	AAA	10000
Dec97	BBB	63000
Dec97	INTC	98000
Dec97	HHH	20000
Dec97	MSN	45600
Dec97	AAA	15000
.....		

date	*company*	*volume*
Dec97	AAA	10000
Dec97	BBB	63000
Dec97	INTC	98000
Dec97	HHH	20000
Dec97	MSN	45600
Dec97	AAA	15000
.....		

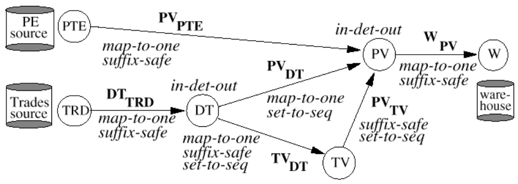
totalvol
10,000,000

Definición:

Secuencia de tuplas: $\mathcal{T} = [t_1..t_n]$; **con atributos** $[a_1..a_m]$

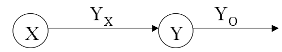
Proceso de carga normal

Representación del workflow de carga con un Grafo Acíclico Dirigido (DAG):



DAG

Notación:



X : Componente

Y : Componente

Y_X : Parámetro de entrada del componente Y producido por X

Y_O : Parámetro de salida del componente Y

Y_X : Secuencia que instancia el parámetro de entrada Y_X

Y_O : Secuencia que instancia el parámetro de salida Y_O

Y(... Y_X...) = Y_O : Y produce Y_O procesando Y_X (y otras secuencias)

W : Secuencia que se carga en el warehouse en ausencia de fallas

Proceso de carga con falla

Tipos de fallas:

El algoritmo hace foco en fallas a nivel de sistema y no a nivel lógico

Observación:

Ante cualquier falla de cualquier componente solo un prefijo de la secuencia W es cargado en el warehouse

Datos para re-asumir la carga:

- El prefijo de la secuencia de entrada al warehouse W
- Procedimientos provistos por los *extractors*

  GetAllReordered(), GetAll(), GetSubSet(), GetSuffix()

Proceso de carga con falla

Propiedades para re-asumir la carga:

(Same-set(Y)) If Y is an extractor then Same-set(Y) = true if Y uses GetAll or GetAllReordered during resumption. Otherwise, Same-set(Y) = true if . Y_X : Same-set(X) and, given the same sets of input tuples, Y produces the same set of output tuples.

En general, se deben contestar estas dos preguntas para evitar el re-proceso de tuplas:

1) Para una tupla dada del warehouse, cuales de las tuplas de Y_X contribuyen a ella.

2) Cuándo es seguro filtrar esas tuplas de Y_X

Proceso de carga con falla

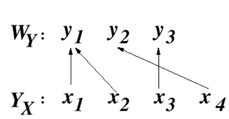
Definición:

(Contributes, Contributors) Given transform Y , let Y(...Y_X…) =Y_O and Y(… Y'_X...) =Y'_O . Also let Y_X = [x_1..x_{i-1} x_i x_{i+1}..x_n] and Y'_X = [x_1..x_{i-1} x_{i+1}..x_n].

Contributes(x_i,y_j) = true, if y_j . Y_O and y_j . Y'_O .Otherwise, Contributes(xi,yj) = false.

Contributors(Y_X, y_j) =$\mathcal{T}$, where IsSubsequence($\mathcal{T}$, Y_X) and

(x_i . $\mathcal{T}$: Contributes(x_i, y_j)) and (x_i . Y_X : Contributes(x_i,y_j) . x_i . $\mathcal{T}$).

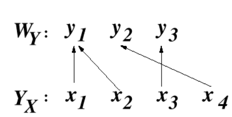


Proceso de carga con falla – Filtrado Seguro

Propiedad:

map-to-one(Y_X)

Given transform Y with input parameter Y_X, Y_X is map-to-one if. Y_X;. Y_O;. x_i . Y_X: (Y (:::Y_X:::) = Y_O) . (.. y_j ; y_k . Y_O such that Contributes(x_i,y_j) and Contributes(x_i,y_k) and j . k).



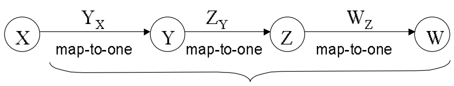
Proceso de carga con falla – Filtrado Seguro

Definición:

Subset-feasible(Y_X)

Given transform Y with input parameter Y_X, Subset-feasible(Y_X) = true if Y is the warehouse inserter.

Otherwise, Subset-feasible(Y_X) = true if Y_X is map-to-one and . Z_Y : Subset-feasible(Z_Y). Otherwise, Subset-feasible(Y_X) = false.



. Subset-feasible(Y_X)

13

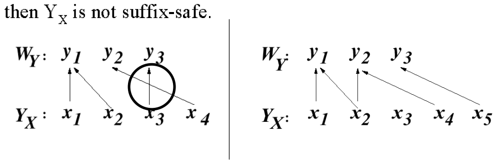
Proceso de carga con falla – Filtrado Seguro

Propiedad:

suffix-safe(Y_X)

Given $\mathcal{T}$ = [t1::tn], let First($\mathcal{T}$) =t1, Last($\mathcal{T}$) =tn, and ti $._{\mathcal{T}}$ tj if ti is before tj in $\mathcal{T}$ or i = j. Given transform Y with input parameter Y_X, Y_X is suffix-safe if . Y_X; . Y_O; . yj ; yj+1 . Y_O: (Y (:::Y_X:::) = Y_O))(Last(Contributors(Y_X, yj)) $._{YX}$ First(Contributors(Y_X,yj+1))).

If Contributors(YX, yj) = [ ] or Contributors(YX, yj+1) = [ ], then Y_X is not suffix-safe.



14

Proceso de carga con falla – Filtrado Seguro

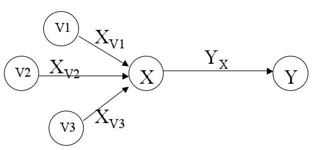
Definición:

Prefix-feasible(Y_X)

Given transform Y with input parameter Y_X,

Prefix-feasible(Y_X) = true if Y is the warehouse inserter.

Otherwise, Prefix-feasible(Y_X) = true if Y_X is suffix-safe and . Z_Y : Prefix-feasible(Z_Y). Otherwise, Prefix-feasible(Y_X) = false.



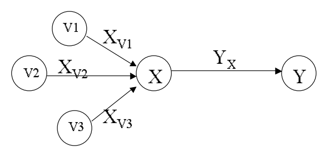
15

Proceso de carga con falla – Filtrado Seguro

Propiedad:

in-det-out(Y)

Transform Y is in-det-out if Y produces the same output sequence whenever it processes the same input sequences.



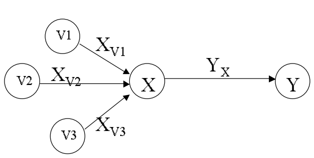
16

Proceso de carga con falla – Filtrado Seguro

Propiedad:

set-to-seq(Y_X)

Given transform Y with input parameter Y_X, Y_X is set-to-seq if (Y is in-det-out) and (Y_X; Y'_X : ((Y_X and Y'_X have the same set of tuples) and (all other input parameters of Y receive the same sequence)) . Y (:::Y_X:::) =Y (:::Y'_X:::)).



17

Proceso de carga con falla – Filtrado Seguro

Definición:

Same-seq(Y_X)

If X is an extractor then Same-seq(Y_X) = true if X uses the GetAll re-extraction procedure.

Otherwise, Same-seq(Y_X) = true if X is in-det-out and . X_V : (Same-seq(X_V) or (X_V is set-to-seq and Same-set(V))). Otherwise, Same-seq(Y_X) = false.

Observación: Dado un path desde una fuente hasta un componente Y, para que se cumpla Same-seq(Y_X) exijo que todos los componentes del path sean in-det-out, y si uno no lo es, entonces exijo que el siguiente sea set-to-seq (más fuerte)

18

Proceso de carga con falla – Ident. Contribuyentes

Objetivo: Identificar las tuplas de una secuencia Y_X que son contribuyentes a una tupla del warehouse w_k

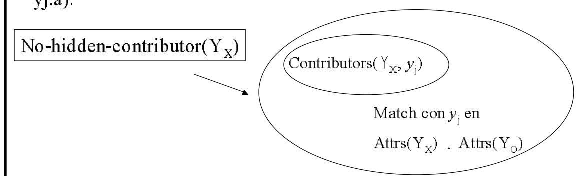
Idea: Matchear los atributos que Y_X y w_k tienen en común

Proceso de carga con falla – Ident. Contribuyentes

Propiedades a satisfacer para poder identificar los contribuyentes:

no-hidden-contributor(Y_X)

Given transform Y with input parameter Y_X, no-hidden-contributors(Y_X) if . Y_X, . Y_O, . yj . Y_O, . xi . Contributors(Y_X, yj), . a . (Attrs(Y_X) . Attrs(Y_O)): (Y (::: Y_X:::) = Y_O) . (xi.a = yj.a).



Proceso de carga con falla – Ident. Contribuyentes

Propiedades a satisfacer para poder identificar los contribuyentes:

Definición:

CandAttrs(Y_X,P)

Let P be a path from input parameter Y_X to the warehouse.

There are three possibilities for CandAttrs(Y_X , P):

1. If Y is the warehouse inserter, then CandAttrs(Y_X, P) =Attrs(Y_X).
2. If . no-hidden-contributors(Y_X), then CandAttrs(Y_X; P) = [ ].
3. Else CandAttrs(Y_X , P) = CandAttrs(Z_Y , P') . Attrs(Y_X), where P = [Y_X Z_Y ::W_I], and P' is P excluding Y_X.

Proceso de carga con falla – Ident. Contribuyentes

Propiedades a satisfacer para poder identificar los contribuyentes:

Propiedad:

no-spurious-output(Y)

A transform Y produces *no spurious output* if . input parameters Y_X, . Y_X, . Y_O, . yj . Y_O : (Y (::: Y_X:::) = Y_O) . (Contributors(Y_X,yj) . [ ]).

Proceso de carga con falla – Ident. Contribuyentes

Observaciones:

no-hidden-contributor(Y_X) = Se cumple si todas las tuplas de entrada Y_X que contribuyen a una tupla de salida y_j matchean con y_j en Attrs(Y_X) . Attrs(Y_O)

CandAttrs(Y_X, P) = Todos los atributos presentes en el path P desde Y_X al warehouse. Estos atributos identifican a los potenciales contribuyentes de una tupla final w_k

idAttrs(Y_X) = Conjunto de atributos utilizados para identificar los contribuyentes de Y_X a la tupla del warehouse w_k. Los contribuyentes serán los que matcheen con w_k en idAttrs(Y_X)

Proceso de carga con falla – Ident. Contribuyentes

Por lo tanto:

Dados w_k y CandAttrs(Y_X, P) puedo identificar a los potenciales contribuyentes de w_k en Y_X.

Dados w_k y idAttrsPath(Y_X, P) puedo identificar a los contribuyentes de w_k en Y_X.

Proceso de carga con falla – Ident. Contribuyentes

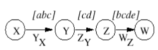


Figure 7: Example Component DAG

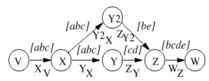


Figure 8: Component DAG with Replicated Outputs

1. IdAttrs(Y_X) = KeyAttrs(Y_X) if KeyAttrs(Y_X) . CandAttrs(Y_X, P) and both Y and Z satisfy the no-spurious-output property.

2. IdAttrs(Y_X) = KeyAttrs(W_Z) if KeyAttrs(W_Z) . CandAttrs(Y_X, P).

3. IdAttrs(Y_X) = IdAttrs(Z_Y) if IdAttrs(Z_Y) . CandAttrs(Y_X, P).

Proceso de carga con falla – Ident. Contribuyentes

Definición:

IdAttrsPath(Y_X,P), IdAttrs(Y_X)

Let P be the only path from Y_X to the warehouse.

There are three possibilities for IdAttrsPath(Y_X,P) (i.e., IdAttrs(Y_X)).

1. If (KeyAttrs(Y_X) . CandAttrs(Y_X, P) and . Z_V . P : Z_V has no spurious output tuples), then (IdAttrsPath(Y_X, P) = KeyAttrs(Y_X)).

2. Otherwise, let Z_V . P but Z_V . Y_X. Let P’ be the path from Z_V to the warehouse. If (IdAttrsPath(Z_V, P’) . [ ] and IdAttrsPath(Z_V, P’) . CandAttrs(Y_X, P)), then (IdAttrsPath(Y_X,P) = IdAttrsPath(Z_V ,P’)).

3. Otherwise IdAttrsPath(Y_X,P) =[ ].