MODELADO AUTOMATICO DE DATOS Y "QUERY LANGUAGES" INTELIGENTES

Breogán Gonda Vázquez
Juan Nicolás Jodal
São Paulo - *Brasil* - Montevideo - *Uruguay*

I. Introducción

En los últimos años el uso de bancos de datos soportados por sistemas de gerencia de banco de datos ha aumentado mucho. Los resultados, sin embargo, muchas veces no corresponden a las expectativas.

Al principio los sistemas de gerencia de banco de datos fueron utilizados cuando se trataba de administrar acervos de datos de gran volumen o de estructura compleja o que necesitaban múltiples vías de acceso. Actualmente su uso se ha generalizado buscando fundamentalmente aumentar la productividad del desarrollo de sistemas y disminuir sus costos de mantenimiento.

Sin duda la baja productividad en el desarrollo de sistemas y los altos costos de mantenimiento son problemas importantes de la informática y vistos relativamente su ponderación sube cada día con relación a los de eficiencia del uso de los recursos de hardware por el gran abaratamiento relativo de estos. Ante esta situación parece especialmente inadecuado tratar de representar los datos pensando en una "eficiencia" que no conseguimos definir bien mientras que, muchas veces, no logramos la imprescindible eficacia.

Estas consideraciones nos orientan a una representación más objetiva del dato.

Debemos, sin embargo, considerar otros elementos: son realmente el costo de mantenimiento y la baja productividad de desarrollo de sistemas los problemas principales de la informática en las empresas?. Estos son, sin duda, problemas importantes pero no son los esenciales: las empresas, de algún modo, tienen cubiertas sus necesidades operativas pero la información de apoyo a las decisiones tácticas y estratégicas es muy inadecuada.

O sea que no estamos dando el debido soporte a las

decisiones más riesgosas.

Cabe preguntarse si es posible implementar sistemas de informacíon que den un buen soporte a estas decisiones.

Sin duda es posible implementar sistemas de información para soportar las actividades operativas, pero estas son previsibles y, en consecuencia, estructurables. Lo son las actividades de toma de decisiones tácticas y estrategicas? ciertamente debemos concluir que en muchos casos no lo son. Entonces, como podremos resolverlas por medio de sistemas?.

El elemento esencial para proveer a la empresa la información necesaria es la "usabilidad" del dato. Necesitamos bancos de datos tales que si existen en ellos los datos necesarios para derivar una determinada información esa derivación sea posible a bajo costo. Este "bajo costo" necesitamos medirlo fundamentalmente en tiempo: tiempo de proceso pero sobre todo tiempo de formulación del pedido de información ya que casi siempre estamos respondiendo a una necesidad no previsible.

Esto nos lleva a la necesidad de modelos que representen fielmente la realidad y nos aseguren una gran "usabilidad" de los datos. Sería ideal pensar en una implementación directa de estos modelos pero las condiciones actuales del ambiente software/hardware a veces nos obligan, por ineficiencias de este ambiente, a introducir desvíos. Eso es totalmente cierto: nuestra experiencia profesional nos dice que en todos los bancos de datos en que hemos trabajado debimos introducir algún desvío para hacer frente a algunos factores críticos pero también nos dice que su importancia relativa, medida en cantidad de tablas con desvíos sobre cantidad total de tablas, es muy pequeña.

Esto nos sugiere manejarnos con un modelo teoricamente ideal e introducir los desvíos que sean estrictamente necesarios.

Vista, entonces, la especial importancia del modelo, falta encontrar el camino para construirlo.

Existen varias metodologías para la obtención de modelos con diferentes orientaciones. Es posible obtener un buen modelo basándonos sólo en elementos formales?. Es implementable en las condiciones software/hardware de hoy o previsibles en el corto plazo un modelo basado en elementos semánticos?.

Un modelo que sólo considera elementos formales puede ser insuficiente para una representación fiel mientras que un modelo que coloca el énfasis en elementos semánticos tiende a complicarse y a hacerse subjetivo. Donde está el equilibrio?.

El modelado de datos no es una tarea individual sino una

tarea de equipo donde intervienen usuarios, analistas de aplicaciones, administradores de datos y administradores de banco de datos. En consecuencia debemos tener un procedimiento claro y simple porque de otra manera el resultado de este trabajo de equipo es imprevisible.

En este trabajo exponemos un procedimiento que cumple con estas condiciones.

La existencia del procedimiento viabiliza la concreción de un programa que efectúe las tareas de modelado en forma incremental.

La existencia de un modelo totalmente representado en forma computacional permite "query languages inteligentes" que sustituirán y/o complementarán a los actuales para el acceso a los bancos de datos relacionales y a los, cada vez más importantes, bancos de datos públicos.

II. Reglas propuestas

Un primer elemento que sirve de antecedente a este trabajo está constituído por las investigaciones del Dr. Edgar F. Codd de I.B.M., realizados en fines de la década del 60 y principio de la del 70.

Codd introduce lo que llamaremos Modelo Relacional Básico que consiste fundamentalmente en lo siguiente:

a) Representación simple: Representación según un conjunto de tablas

b) Procedimiento para obtener representaciones de los datos que no contengan redundancia lógica: normalización

c) Operadores poderosos para tratar las tablas: álgebra relacional

d) Restricciones de integridad para la consistencia intra-tablas

Podemos considerar aquí los siguientes elementos:

d1) Las filas de una tabla son todas diferentes y, en consecuencia :

d2) Existe por lo menos una super-llave que es cualquier identificador unívoco de una fila entre todas las de una tabla

d3) Existe por lo menos una llave candidata que es cualquier super-llave mínima (una super-llave tal que ningún sub-conjunto estricto de ella sea una super-llave)

d4) Llave que es la llave candidata elegida en el modelo para una determinada tabla

d5) Ningún atributo de la llave puede tomar valores nulos

d6) El valor que toma el atributo - simple o compuesto - llave, en una determinada fila de una tabla no puede modificarse

Todo esto constituye una buena base y además está implementado en la mayor parte de los sistemas de gerencia de

banco de datos autoproclamados como "relacionales" que están hoy en el mercado, pero es insuficiente ya que no nos asegura un modelo consistente.

Cuales son los problemas?. El esencial es que en este modelo las tablas son independientes, lo que no es cierto en la realidad.

El problema más visible se soluciona con las siguientes reglas:

e) Si x, atributo simple o compuesto, es llave de una tabla T1 lo llamaremos "primario" y puede aparecer también en otras tablas. Si y, atributro simple o compuesto, es tal que no existe en el modelo una tabla que lo tenga como llave, lo llamaremos "secundario" y no puede aparecer en ninguna otra tabla.

f) No existirán dos tablas con la misma llave

g) Si x, atributo simple o compuesto, es llave de una tabla T1 y aparece también en una segunda tabla T2, dada una fila cualquiera de T2, j, el valor correspondiente de x, T2.xj debe ser tal que exista en T1 una fila i tal que el valor correspondiente de x, T1.xi cumpla T1.xi = T2.xj.
(∀ x EX T2.x ===> EX T1.x)

En este caso decimos, más brevemente que T2 está subordinada a T1 según T2.X ===> T1.X.

Esto, sin embargo, es de utilidad en un ambiente padronizado de nombres y el modelo relacional básico no lo es.

Adicionalmente, hasta el momento, el significado del modelo reside en el conjunto de los nombres de las tablas y los nombres de los atributos, lo que constituye un casuismo bastante indeseable.

En particular es causante del bajo nivel de los "query languages" basados en álgebra relacional, como se muestra a continuación:

```
select CLIENTE.CLIENTE#, NOMBRE_CL, FACTURA.FACT#,
       FECHA-FACT,     LINEA_FAC.PRODUCTO#,     DESC_PR,
       CANT_VEND, PU_VEND, CANT_VEND * PU_VEND
from   CLIENTE, FACTURA, LINEA_FAC, PRODUCTO
where  CLIENTE.CLIENTE# = FACTURA.CLIENTE#
and    FACTURA.FACT# = LINEA_FAC.FACT#
and    LINEA_FAC.PRODUCTO# = PRODUCTO.PRODUCTO#
 and    condiciones del problema
           .........................
           .........................
           .........................
```

La causa del bajo nivel de estos lenguajes reside en que el sistema "no conoce" el modelo, dada su casuística, y es el usuario que debe sustituirlo en esta tarea.

Una primera regla de padronización es la siguiente:

h) Si dos atributos tienen el mismo "significado" deben tener el mismo nombre y si tienen diferente "significado" deben tener diferente nombre, **siempre que ello sea posible.**

i) No existirán en una tabla dos atributos con el mismo nombre.

Esta regla introduce un primer elemento semántico que es el "significado" que podríamos caracterizar como la "naturaleza del dato". Notese que lo caracterizamos y no lo definimos: apelamos a nuestros conceptos primitivos.

Esta dificultad para definir debe ser contornada a la hora de especificar porque, de lo contrario, introduciremos una fuerte subjetividad en el modelo. Para ello, introduciremos el concepto de dominio asociado a un atributo y el de "familia de atributos". El "dominio" consistirá, para nosotros, en un conjunto de valores posibles, dado explicita o implicitamente, un formato de esos valores y el significado del atributo. Un atributo tiene asociado un determinado dominio del que toma sus valores.

Diremos que los atributos **a** y **b** forman una familia si se cumple:

1. **a** y **b** son equivalentes o bien

2. **a** es una generalización de **b** o (lo que es equivalente) **b** es una particularización de **a**

Diremos que **a** es una generalización, en sentido amplio, de **b** si se cumple (1) o (2).

Podrá darse el caso de que más de un atributo tome valores del mismo dominio?. Si, supongamos el caso de que existe una tabla cuya llave es **producto#** y otra cuya llave es **producto_compuesto# + producto_componente#** aquí el significado de **producto#**, **producto_compuesto#** y **producto_componente#** es el mismo pero se da lo siguiente:

- por (i) los atributos **producto_compuesto#** y **producto_componente#** no pueden tener el mismo nombre

- todo **producto-compuesto#** tiene asociado un **producto#**, pero el inverso no es cierto (**producto** es una generalización en sentido estricto de **producto-compuesto**)

- simetricamente, todo **producto-componente#** tiene asociado un **producto#**, pero el inverso no es cierto (**producto** es una generalización en sentido estricto de **producto-componente**)

Como regla práctica usaremos la siguiente:

- si existe un único atributo con un determinado significado, daremos a atributo, dominio y significado el nombre del atributo

- si existen varios atributos con el mismo significado, debemos determinar si existe entre ellos una relación de generalización/particularización la cual pasará a integrar el modelo.

ATRIBUTO	PARTICULARIZACI'ON DEL ATRIBUTO
b	b
a	b

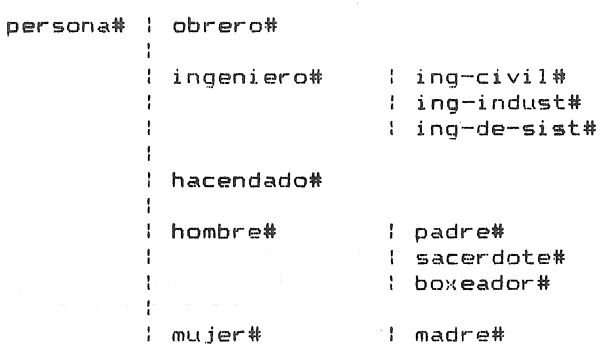
lo que deja claramente expresada la relación existente.

Véase que a esta altura surge claro el "siempre que sea posible" de (h), pero, con esa frase, la utilidad de (h) es bastante relativa.

Adicionalmente, lo que hemos visto para dos niveles, es válido para cualquier cantidad de niveles:

ATRIBUTO	PARTICULARIZACION DEL ATRIBUTO
persona#	persona#
obrero#	persona#
ingeniero#	persona#
hacendado#	persona#
ing-civil#	ingeniero#
ing-indust#	ingeniero#
ing-de-sit#	ingeniero#
hombre#	persona#
mujer#	persona#
padre#	hombre#
sacerdote#	hombre#
boxeador#	hombre#
madre#	mujer#
monja#	mujer#

lo que corresponde a la jerarquía:



Debemos re-escribir (h), de la siguiente forma:

j) Si dos atributos tienen significados diferentes tendrán nombres diferentres.

k) Si varios atributos tienen el mismo significado se dará el mismo nombre a aquellos que sean equivalentes y, en otros casos, se darán nombres que representen claramente las relaciones de generalización o particularización, directas o transitivas, que existan entre ellos.

Ahora, podemos re-formular (g) de la siguiente forma:

l) Si el atributo, simple o compuesto, x = x0,x1,x2,...,xn es llave de una tabla T1 y existe en una tabla T2 el atributo, simple o compuesto, y = y0,y1,y2,...,yn tal que, para todo i se cumple que xi es una generalización en sentido amplio de yi, por razones de consistencia, debe cumplirse que, ∀ i EX T2.yi ==> EX T1.xi siendo xi = yi, o más brevemente, diremos que T2 esta subordinada a T1 según T2.y ===> T1.x.

De esta forma estamos resolviendo los principales problemas de consistencia.

En estas condiciones el significado del modelo reside en los nombres de los atributos, siendo los nombres de las tablas irrelevantes.

Además, ahora el sistema "conoce" el modelo y podemos pensar en problemas como el del "query" que ya hemos visto y cuya solución basada en álgebra relacional listamos a continuación, y resolverlo ahora de una forma mucho más elegante y simple mediante cálculo relacional:

- Solución tradicional, mediante álgebra relacional

```
select CLIENTE.CLIENTE#, NOMBRE_CL, FACTURA.FACT#,
       FECHA-FACT,     LINEA_FAC.PRODUCTO#,     DESC_PR,
       CANT_VEND, PU_VEND, CANT_VEND * PU_VEND
from   CLIENTE, FACTURA, LINEA_FAC, PRODUCTO
where  CLIENTE.CLIENTE# = FACTURA.CLIENTE#
and    FACTURA.FACT# = LINEA_FAC.FACT#
and    LINEA_FAC.PRODUCTO# = PRODUCTO.PRODUCTO#
and    condiciones del problema
          .........................
          .........................
          .........................
```

Solución mediante cálculo relacional

```
select CLIENTE#,    NOMBRE_CL,    FACT#,    FECHA-FACT,
       PRODUCTO#, DESC_PR, CANT_VEND, PU_VEND,
       CANT_VEND * PU_VEND
where  condiciones del problema
          .........................
          .........................
          .........................
```

Aún queda, sin embargo, un problema para resolver: como conoce el usuario los atributos?

Como ahora el sistema "conoce" el modelo, podemos pensar en un query guiado de mucho más alto nivel, desde el punto de vista del usuario.

Adicionalmente, podemos ver que la operación de JOIN es inconsistente y de resultados imprevisibles a menos que se cumpla, por lo menos, lo siguiente:

m) La fusión (JOIN) entre dos tablas T1 y T2 según los atributos, simples o compuestos, x = x0,x1,x2,...,xn de la tabla T1 y y = y0,y1,y2,...,yn de la tabla T2 sólo puede tener sentido si se cumple que, para todo i es significado(xi) = significado(yi).

En rigor, en las mayor parte de los casos, este operador no

tendrá utilidad alguna si no se cumple lo siguiente, que es bastante más restrictivo:

n) La fusión (JOIN) entre dos tablas t1 y t2 según los atributos, simples o compuestos, x = x0,x1,x2,...,xn de la tabla T1 y y = y0,y1,y2,...,yn de la tabla T2 tiene sentido si se cumple que, o bien T2 está subordinada a T1 según T2.y ===> T1.x, o bien que T1 está subordinada a T2 según T1.x ===> T2.y.

Adicionalmente, para cumplir con (1) de una forma viable en las condiciones software/hardware de hoy, todos los atributos primarios y aquellos secundarios que sean particularizaciones de atributos primarios deberán ser indexados en cada una de las tablas en que aparezcan (a menos que la subordinación entre las tablas en que se hallen involucrados esté asegurada por transitividad) lo que hace que, en casi todos los casos, las navegaciones consistentes en el banco de datos puedan hacerse aprovechando índices que ya existen por razones de consistencia.

III. Resultados

Las reglas enunciadas nos dan un procedimiento simple y objetivo para el modelado de datos.

Cuando el tamaño de los modelos crece, la dificultad crece mucho más, con independencia de la metodología utilizada.

La metodología propuesta, dado que esta basada en reglas claras, tiene la ventaja adicional de que puede utilizar con mucho provecho herramientas automáticas.

Estas herramientas automáticas pueden ayudarnos en lo siguiente:

- Creación y mantenimiento del modelo totalmente normalizado por síntesis, a partir de visiones objetivas de los datos.

- Introducción y mantenimiento de desvíos al modelo totalmente normalizado, las que, junto con aquel, representan el modelo físico del banco de datos.

- Pre-procesador de lenguaje "query" basado en cálculo relacional generando codigo fuente para procesadores basados en álgebra relacional.

- Pre-procesador de "query" guiado de alto nivel generando codigo fuente para procesadores basados en álgebra relacional.

- Generador de visiones y lógica de acceso al, y de mantenimiento de consistencia del, banco de datos para lenguajes de tercera y cuarta generación.

- Generador de aplicaciones extremadamente poderoso.

- Esto viabiliza una nueva generación de sistemas de gerencia de banco de datos, los que llamaremos "relacionales inteligentes" que "conocen" su modelo que incluye los siguientes puntos:

  a) Representación tabular

  b) Normalización

  c) Operadores del álgebra relacional

  d) Integridad intra-tablas

  e) Cada atributo secundario aparece en una única tabla del modelo

  f) No existencia de dos tablas con la misma llave

  g) Subordinación (esta condición es un caso particular de (l))

  h) Padronización de nombres de atributos (esta condición es un caso particular de (k))

  i) No existencia en una determinada tabla de dos atributos con el mismo nombre

  j) Atributos con significados diferentes tendrán nombres diferentes

  k) Padronización generalizada y jerarquización de atributos con el mismo significado

  l) Subordinación generalizada

  m) Fusión consistente

Bibliografía

Codd, E. F. [1970] A Relational Model of Data for Large Shared Data Banks, CACM, Junio 1970.

Codd, E. F. [1979] Extending the Database Relational Model to Capture More Meaning ACM TODS, Diciembre 1979.

Date, C.J. [1981] An Introduction to Database Systems, 3ra. Ed.,Addison-Wesley

Maier, D. [1983] The Theory of Relational Databvases, Pitman

SCI [1986] Data Expert: Manual del Usuario

SCI [1986] Data Expert: Manual de Referencia

Gonda, B. y Jodal, N. [1986] Modelado de Datos, I Seminario Latino-Americano de Administración de Datos, Petrópolis, Rio de Janeiro, Brasil, Junio de 1986