Una Extensión Espacial al Operador Data Cube

A Spatial Extension to Data Cube Operator

Andrés F. Urrea M., Ing., Francisco J. Moreno A., PhD(c)
Universidad Nacional de Colombia - Sede Medellín
afurrea@gmail.com, fjmoreno@unal.edu.co

Recibido para revisión 28 de Noviembre de 2007, aceptado 14 de Febrero de 2008, versión final 28 de Febrero de 2008

Resumen—**El operador map cube es una extensión espacial del operador data cube. Su objetivo es soportar agregación espacial en un contexto multidimensional y proporcionar la visualización correspondiente de la información en forma de álbumes de mapas. Sin embargo map cube presenta inconsistencias que son corregidas en este artículo. Además el operador se extiende para soportar diferentes funciones de agregación. Se presentan dos casos de estudio que muestran las extensiones y mejoras propuestas.**

Palabras Clave—**Bases y Bodegas de Datos, Minería de Datos, Data Cube, Map Cube.**

Abstract—**Map Cube is a spatial extension to data cube. The purpose of map cube is to support spatial aggregations in a multidimentional context and to visualize the information in the form of albums of maps. However, map cube has inconsistencies which are corrected in this article. We also extend it to support diferent aggregation functions. We present three case studies in order to illustrate the proposed extensions and improvements.**

Keywords—**Databases and Data Warehouses, Data Mining, Data Cube, Map Cube.**

I. INTRODUCCIÓN

El operador *map cube* fue concebido por Shekhar [1] [2], como una extensión espacial al operador *data cube* [3].

A su vez el operador data cube es una generalización de la cláusula Group By de SQL [4]. Dadas n columnas de agrupamiento, data cube genera subtotales para todas los posibles combinaciones de éstas (2^n). Cada combinación se denomina cuboide [5]. Por ejemplo, un data cube por tienda y producto, genera subtotales por (tienda, producto), por (tienda), por (producto) y el gran total.

Map cube genera un mapa asociado a cada cuboide generado por el operador data cube, integrando datos y mapas en un conjunto conocido como *vista map cube*. Esta vista permite una visualización amigable de los datos, esto es importante ya que el operador fue concebido en el dominio de las bodegas de datos espaciales donde uno de los principales problemas es la visualización.

Sin embargo al revisar el operador se encuentran inconsistencias en su gramática. Éstas se describen en la Sección 2.

El operador también posee limitaciones:

- sólo se puede realizar la agregación espacial mediante la operación de unión geométrica [6]. Esto impide el uso de otras funciones de agregación, útiles en diferentes contextos, véase la Sección 3, tales como envoltura convexa (*convex hull*), intersección, rectángulo de mínima frontera, centro de masa, entre otras.
- imposibilidad de realizar más de una operación de agregación espacial en una misma operación.

En este artículo se corrigen las inconsistencias gramaticales y se extiende el operador con el fin de superar las limitaciones enunciadas.

El resto del artículo está organizado así: la Sección 2 detalla las características principales del operador *map cube*, seguido de una descripción de las inconsistencias gramaticales, las limitaciones identificadas y la propuesta de extensión. En la Sección 3 se presentan los casos de estudio. Finalmente, en la Sección 4 se presentan conclusiones y el trabajo futuro.

II. MAP CUBE

A. Características

Para generar la *vista map cube* de un conjunto de datos, el operador separa los datos textuales y los datos espaciales. Dadas n columnas de agrupamiento, map cube genera subtotales para todas las posibles combinaciones de éstas (2^n) cada una con su correspondiente mapa.

La sintaxis del operador *map cube* propuesta en [1], definida mediante el lenguaje "Yacc" [7] se presenta a continuación.

Map Cube

Base-map = <base-map name>
Base-table = <base-table name>
(Where <join attribute list>
(And <join attribute list>)*)?
Aggregate by <aggregate list>
Reclassify by <attribute list>
Data cube dimension <attribute list>
Cartographic preference <carto attribute list>

<base-map name> → <name> (, <name>)*
<base-table name> → <name>
<aggregate list> → <aggregate unit>
(<operator><aggregate unit>)?
<aggregate unit> → <aggregate func> : <name>
<aggregate func> → SUM | MAXN | MINN | COUNT | MEDIAN
<join attribute list> → <name> <operator> <name>
<attribute list> → <name>? | <name> (, <name>)*
<carto attribute list> → <carto-attribute-value pair>
(, <carto-attribute-value pair>)*
<carto-attribute-value pair> → <carto-attribute> = <carto-value>
<carto-attribute> → Color | Thickness | Texture | Annotation|Text | Symbol | Layout | Legend | Title | No-of-map-per cuboid | Normalize
<carto-value> → <name> | <num>
<num> → <digit>+ (. <digit>+)? (E (+ | -)? <digit>+)?
<name> → <letter> (<letter> | <digit> | <symbol>)*
<letter> → A | B | … | Z | a | b | … | z
<digit> → 0 | 1 | 2 | 3 | 4 | … | 9
<symbol> → - | _ | , | . | :
<operator> → = | > | < | + | - | * | /

Cada consulta realizada mediante el operador *map cube* es expresada como una consulta SQL. Por ejemplo la siguiente consulta:

Base Map MapaBase
Base Table TablaBase
Aggregate by SUM:Medida
Reclassify by Dim1, Dim2
Data cube dimension Dim1, Dim2
Cartographic Preference No-of-map-per-cuboid = 1, Symbol = Medida

Se expresa así:

```
SELECT Dim1, Dim2, SUM(Medida) as Medida,
       GEOMETRIC_UNION(ColumnaEspacial) as
       ColumnaEspacial
FROM   TablaBase
GROUP BY CUBE Dim1, Dim2
```

Donde las columnas de la cláusula SELECT se obtienen a partir del término *Aggregate by* y del término *Reclassify by*, el FROM a partir de *Base Table*, y los atributos del CUBE a partir de la unión de los términos *Reclassify by* y *Data cube dimension*.

B. Revisión

Al revisar la gramática del operador *map cube*, se identifican aspectos que merecen ser aclarados y mejorados:

- El elemento <*name*> permite el uso de símbolos como el guión (-), el subrayado (_), la coma (,), el punto (.), y los dos puntos (:), los cuales generan inconsistencias. Por ejemplo, un nombre válido en esta gramática es 'variable,variable' el cual podría confundirse con el nombre de dos variables separadas por coma. Además si se utiliza este nombre en el término <*join attribute list*> se obtiene una expresión de comparación inválida. Igualmente ocurre con los demás elementos que contienen el elemento <*name*>.

  Estas inconsistencias también se generan para el resto de los símbolos permitidos <symbol> excepto para el '_'.

- El elemento <*base-table name*> debería definirse de la misma forma que el elemento <*base-map name*>. <*base-map name*> puede ser una lista de nombres explícitamente, mientras que <*base-table name*> puede llegar a expresar una lista de nombres por medio de comas, debido a la forma como se define el término <*name*>. Esto sugiere una falta de uniformidad en la gramática [8].

- El elemento terminal <*operator*> contiene operadores aritméticos (+, -, *, /) y de comparación (=, <, >). Al definirlo así se pueden generar errores. Por ejemplo, en el elemento <*join attribute list*>, el cual tiene sentido para operaciones de comparación, puede formarse una expresión que indique una operación aritmética así: columna1 + columna2 en vez de columna1 = columna2.

  De igual manera, en el término <aggregate list>, donde sólo tienen sentido operaciones aritméticas, puede generarse una operación de comparación como columna1 > columna2 en vez de columna1 * columna2.

- Sólo se pueden realizar restricciones sobre el término <*base-table name*> por medio de conjunciones. Esto imposibilita operaciones de disyunción.

- El término <*aggregate list*> no permite especificar varias

funciones de agregación. Por ejemplo no es posible expresar 'SUM:columna1, COUNT:columna2'.

C. Extensión

Se propone una extensión al operador *map cube*, con el fin de cubrir casos no contemplados en la propuesta original y corregir los aspectos mencionados en la sección anterior.

A continuación se presenta la nueva gramática para el operador *map cube*.

Gramática Map Cube Extendido

Base Map <table list>
Base Table <table list>
(WHERE <condition>)?
Aggregate by <aggregate list>
Reclassify by <attribute list>
Data cube dimension <attribute list>
Cartographic preference <carto attribute list>

<table list> → <table name> (, < table name>)*
<table name> → <name>
<condition> → <join attribute pair> (<logical operator> <join attribute pair>)*
<join attribute pair> → <column name> <comparison operator> (<column name> | <value>)
<aggregate list> → <aggregate type> (, <aggregate type>)*
<aggregate type> → <simple aggregation> | <spatial aggregation>
<simple aggregation> → <simple aggregation unit> (<arithmetic operator> (<simple aggregation unit> | <number>))* (AS <name>)?
<simple aggregation unit> → (<simple aggregate function> | <special aggregate function>) : <column name>
<special aggregate function> → <numeric spatial function> : <spatial aggregation function>
<spatial aggregation> → <spatial aggregation unit> (AS <name>)?
<spatial aggregation unit> → <spatial aggregation function> : <column name>
<attribute list> → <column name> (, <column name>)*
<carto attribute list> → <carto-attribute-value pair> (, <carto-attribute-value pair>)*
<carto-attribute-value pair> → <carto-attribute> = <carto-value>
<carto-attribute> → Color | Thickness | Texture | Annotation | Text | Symbol | Layout | Legend | Title | No-of-map-per cuboid |

Normalize
<carto-value> → <name> | <num>
<column name> → <name> | <compound column>
<compound column> → <table name> . <name>
<user function> → <name>
<value> → '<name>' | <number>
<name> → <letter> (<letter> | <digit> | _)*
<number> → (-)? (<digit>)+ (. (<digit>)+)?
<letter> → A | B | C … | Z | a | b | c … | z
<digit> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<arithmetic operator> → + | - | * | / |
<comparison operator> → > | < | >= | <= | = | <>
<logical operator> → AND | OR
<simple aggregate function> → SUM | MAXN | MINN | COUNT | MEDIAN | AVG | MAX | MIN | <user function>
<numeric spatial function> → AREA | PERIMETER | LENGTH | <userfunction>
<spatial aggregation function> → GEOMETRIC_UNION | INTERSECTION | CONVEX_HULL | CENTROID | MBR | CENTER_OF_MASS | UNION_BY_CONTINUOUS_POLYGON | <user function>

Los cambios introducidos son los siguientes:

- Los elementos <*base-map name*> y <*base-table name*> se definen mediante el elemento <*table list*> para indicar explícitamente que consisten de una lista de nombres. <*table list*> se descompone en un conjunto de elementos que representan nombres que solo pueden contener letras, dígitos y el símbolo '_'.
- Se separan los operadores aritméticos y de comparación en los elementos <arithmetic operator> y <*comparison operator*> respectivamente. También se amplia la gama de operadores de comparación con >=, <= y <>; y se incluye un nuevo elemento terminal llamado <*logical operator*> para realizar operaciones de conjunción y disyunción.
- Se incluye el elemento <*condition*>, el cual está orientado a condiciones de reunión (join), aunque también permite comparaciones simples, es decir, comparar una columna con valores numéricos o alfanuméricos. Esto permite un mayor número de posibilidades de restricción sobre el término <*table list*> de la claúsula *Base Table*.
- Se modifica el término <*aggregate list*> para permitir una lista de agregados tanto simples como espaciales.

Un agregado simple retorna un valor numérico o alfanumérico, mientras que un agregado espacial retorna una figura geométrica.

Un agregado simple puede ser de dos tipos: una agregación mediante COUNT, SUM, entre otras, o una combinación de una función <*numeric spatial function*> como AREA, LENGTH y PERIMETER, junto con una operación de agregación espacial, por ejemplo: AREA:GEOMETRIC_UNION:columnaespacial. Estas dos posibilidades se representan en el elemento <*simple aggregation*>.

Las funciones incluidas en <*numeric spatial function*> sólo se deben aplicar a operaciones de agregación espaciales que retornen una geometría distinta de un punto, para evitar expresiones como AREA:CENTROID:ConjuntoPuntos, ya que el área de un punto no tiene sentido.

Además, se puede renombrar el resultado de un agregado mediante la palabra clave 'AS'.

- El elemento <*user function*> permite la inclusión de funciones definidas por el usuario. Éstas se pueden adicionar a las funciones de agregación simple, a las funciones numéricas espaciales y a las de agregación espacial. De acuerdo con el tipo de función se debe tener en cuenta:
  - Agregación simple: la función debe recibir como parámetros valores alfanuméricos y retornar valores alfanuméricos.
  - Funciones numéricas espaciales: la función debe recibir como parámetros figuras geométricas y retornar un valor numérico.
  - Agregaciones espaciales: la función debe recibir como parámetros figuras geométricas y retornar figuras geométricas.

La posibilidad de elegir la función para realizar la agregación espacial es el aspecto más destacado dentro de la nueva definición del operador *map cube*, ya que en el operador original la operación *geometric union* está predeterminada[1].

III. EJEMPLOS MAP CUBE EXTENDIDO

En esta sección se presentan dos casos de estudio que ilustran las nuevas características del operador.

A. Geometric Union

Considérese un sector agrícola compuesto por cuatro zonas de cultivo. Se producen cuatro tipos de alimentos: cebolla, cilantro, tomate y repollo.

La Tabla 1 muestra la distribución de los cultivos en el sector.

TABLA 1. SECTORAGRICOLA

Zona	*Cultivo*	*FormaCultivo*
Z_1	Cebolla	F_1
Z_1	Cilantro	F_2
Z_1	Tomate	F_3
Z_1	Repollo	F_4
Z_1	Cebolla	F_5
Z_1	Tomate	F_6
Z_2	Cilantro	F_7
Z_2	Repollo	F_8
Z_2	Cebolla	F_9
Z_2	Tomate	F_{10}
Z_2	Repollo	F_{11}
Z_2	Cebolla	F_{12}
Z_2	Tomate	F_{13}
Z_2	Cebolla	F_{14}
Z_3	Tomate	F_{15}
Z_3	Cebolla	F_{16}
Z_3	Cilantro	F_{17}
Z_3	Cebolla	F_{18}
Z_3	Cilantro	F_{19}
Z_3	Tomate	F_{20}
Z_4	Tomate	F_{21}
Z_4	Cebolla	F_{22}

La columna FormaCultivo representa la forma y la localización del cultivo dentro de la zona en el sector agrícola, como se muestra en la Figura 1.

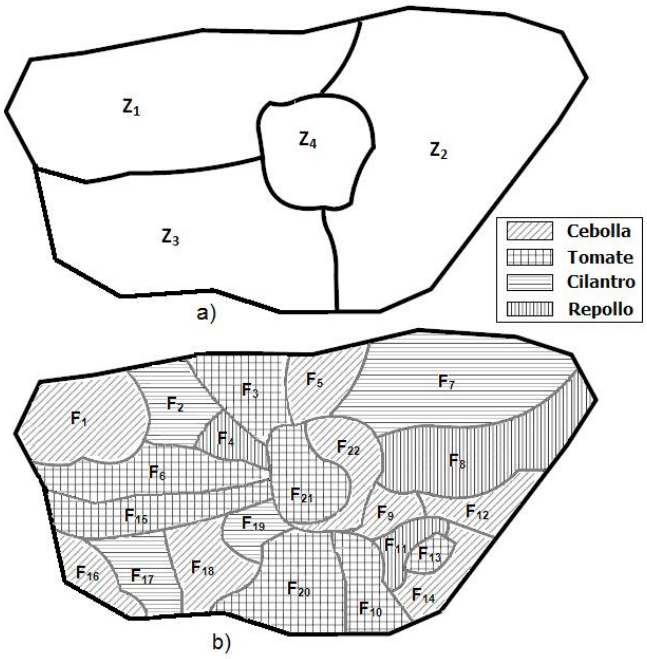

Figura 1. MapaSectorAgricola: a) distribución de zonas y b) distribución de cultivos.

[1] En [1] se utiliza en algunos ejemplos, también de forma predeterminada, un tipo de unión geométrica por adyacencia (geometric union by continuos polygon).

Se desea visualizar la forma y el área que abarcan los cultivos de tomate y cebolla, dentro de cada zona y para todo el sector agrícola. También se desea visualizar que tan dispersos están dichos cultivos, ya que su control se facilitaría si estuvieran distribuidos de manera contigua.

Para lograr lo anterior se puede utilizar el operador map cube:

Base Map MapaSectorAgricola
Base Table SectorAgricola
WHERE Cultivo = 'Tomate' OR
Cultivo = 'Cebolla'
Aggregate by GEOMETRIC_UNION: FormaCultivo AS FormaCultivo,
AREA: GEOMETRIC_UNION: FormaCultivo AS AreaCultivada_m2
Reclassify by Zona, Cultivo
Data cube dimension Zona, Cultivo
Cartographic Preference No-of-map-per-cuboid = 1,
Symbol = FormaCultivo

La consulta SQL generada implícitamente por el operador es:

```
SELECT   Zona, Cultivo ,
         GEOMETRIC_UNION(FormaCultivo) AS
         'FormaCultivo',
         AREA(GEOMETRIC_UNION(FormaCultivo)) AS
         'AreaCultivada_m2'
FROM     SectorAgricola
WHERE  Cultivo = 'Tomate' OR
         Cultivo = 'Cebolla'
GROUP BY CUBE Zona, Cultivo.
```

Map cube genera resultados para los cuboides de la Figura 2.

Figura 2. Cuboides generados por map cube para Zona y Cultivo.

La Tabla 2 muestra para cada zona y cultivo su forma y localización, así como el área total cultivada. En la Figura 3 se observa la distribución espacial correspondiente. Esto corresponde al cuboide (Zona, Cultivo). Nótese que el resultado de *map cube* para este cuboide está conformado tanto por la Tabla 2 como por la Figura 3.

TABLA 2. GRUPO POR ZONA Y CULTIVO

Zona	*Cultivo*	*FormaCultivo*	*AreaCultivada_m2*
Z_1	Cebolla	Q_1	280
Z_1	Tomate	Q_2	420
Z_2	Cebolla	Q_3	315
Z_2	Tomate	Q_4	210
Z_3	Cebolla	Q_5	245
Z_3	Tomate	Q_6	455
Z_4	Cebolla	Q_7	140
Z_4	Tomate	Q_8	210

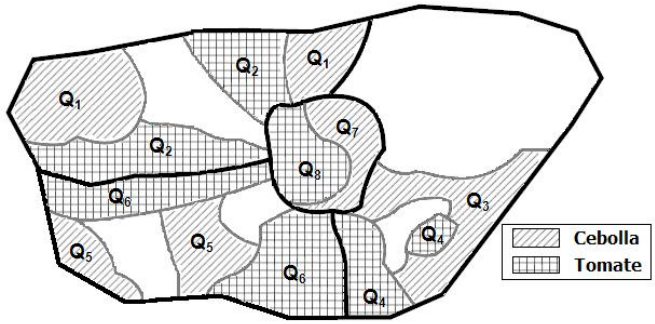

Figura 3. Resultados por zona y cultivo.

Con fin de facilitar la visualización en este caso de estudio, a los mapas generados por el operador *map cube* se les agrega la delimitación de la zonas dentro del sector agrícola y a cada tipo de cultivo se le asigna una convención.

De la Figura 3 se observa que los cultivos de cilantro y repollo, representados por las regiones blancas, intervienen en la continuidad de los cultivos de tomate y cebolla, lo que dificulta su unificación.

La Tabla 3 y la Figura 4 corresponden al cuboide por (Zona) y la Tabla 4 y la Figura 5 corresponden al cuboide por (Cultivo).

TABLA 3. GRUPO POR ZONA

Zona	*Cultivo*	*FormaCultivo*	*AreaCultivada_m2*
Z_1	Todos	R_1	1400
Z_2	Todos	R_2	1050
Z_3	Todos	R_3	1400
Z_4	Todos	R_4	700

La Tabla 3 muestra la agrupación de cultivos para cada zona y el tamaño del área cultivada. La forma y localización de cada zona se observa en la Figura 4.

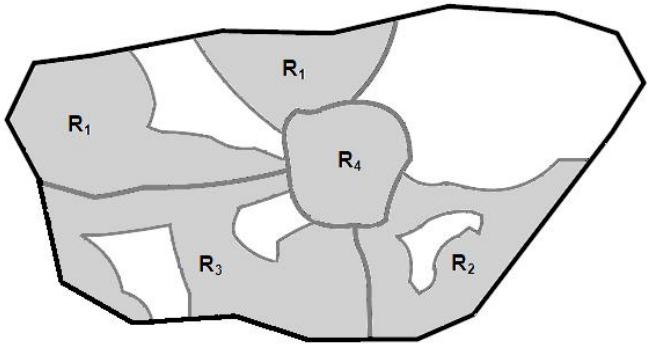

Figura 4. Resultados por zona.

En la Figura 4 las regiones blancas corresponden a cultivos de otros productos o a regiones no cultivadas.

TABLA 4. GRUPO POR CULTIVO

Zona	*Cultivo*	*FormaCultivo*	*AreaCultivada_m2*
Todos	Cebolla	S_1	1960
Todos	Tomate	S_2	2590

La Tabla 4 muestra la agrupación por cada cultivo para todo el sector con el área cultivada correspondiente; su forma y localización se representa en la Figura 5.

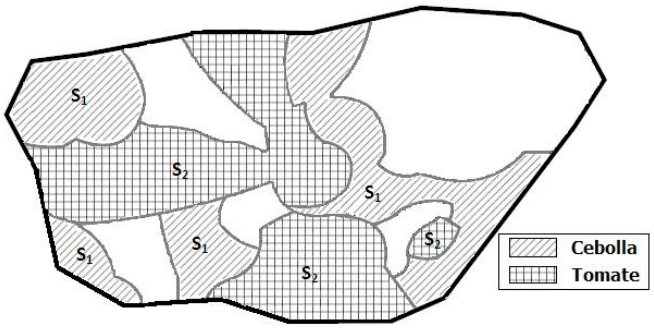

Figura 5. Resultados por cultivo.

A partir de la Figura 5 se concluye que los cultivos de cebolla están más dispersos que los cultivos de tomate.

Además en la parte central del mapa, se observa una pequeña porción de cultivo de cebolla que divide dos grandes áreas de cultivo de tomate. Dicha porción podría reemplazarse por tomate para facilitar el control.

Finalmente, la Tabla 5 y la Figura 6, muestran el área cultivada de los dos productos en todo el sector agrícola. Esto hace referencia al cuboide (Todos).

TABLA 5. GRUPO TODOS

Zona	*Cultivo*	*FormaCultivo*	*AreaCultivada_m2*
Todos	Todos	**T_1**	4550

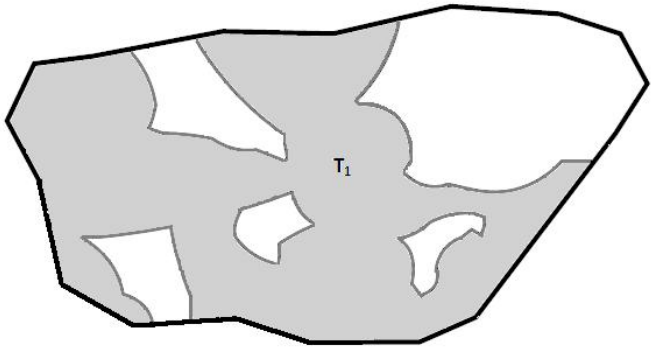

Figura 6. Resultados de los cultivos en todo el sector agrícola.

B. Múltiple agregación espacial, Center of Mass y Geometric Union

Considérese una ciudad conformada por seis barrios, para los cuales se tienen registros de tres tipos de crímenes: asesinato, robo y golpiza.

La Tabla 6 describe cada barrio junto con su delimitación espacial (columna Demarcación). La Tabla 7 contiene los datos de los crímenes que se presentan en los barrios, la columna localización indica el punto donde ocurrió el crimen, la columna NroPer muestra el número de víctimas.

TABLA 6. TABLA BARRIO

IdBarrio	*Nombre*	*Demarcación*
B_0	Barrio 0	D_0
B_1	Barrio 1	D_1
B_5	Barrio 5	D_5

TABLA 7. TABLA CRIMEN

IdBarrio	*Crimen*	*Localización*	*NroPer*
B_0	Robo	P_1	2
B_1	Asesinato	P_8	1
B_5	Golpiza	P_{32}	10

A diferencia del ejemplo anterior, donde a los mapas generados por el operador se les agregaba las delimitaciones de las zonas del sector agrícola, en este caso las delimitaciones de los barrios son incluidas explícitamente, gracias a la múltiple agregación espacial que permite el operador.

Lo que sí se adiciona a los mapas, son las convenciones para identificar cada crimen.

La demarcación de los barrios se muestra en la Figura 7 a) y la localizacion de los crimenes en la ciudad se muestra en la Figura 7 b).

Supóngase que se desea analizar cuáles son los puntos con más densidad criminal en la ciudad, así:

- en cada barrio,
- según el tipo de crimen,
- según el barrio y el tipo de crimen y
- en toda la ciudad.

También se desea determinar el promedio de víctimas de acuerdo con los grupos anteriores.

Para lograr lo anterior se puede utilizar el operador *map cube*:

Base Map MapaCrimenes
Base Table TablaBarrio, TablaCrimen
WHERE TablaBarrio.idBarrio = TablaCrimen.idBarrio
Aggregate by GEOMETRIC_UNION:TablaBarrio.Demarcacion AS Demarcacion
CENTER_OF_MASS: TablaCrimen.Localizacion AS Localizacion,
AVG: TablaCrimen.NroPer AS PromVictimas
Reclassify by Barrio, Crimen
Data cube dimension Barrio, Crimen
Cartographic preference No-of-map-per-cuboid = 1,
Symbol = Localizacion

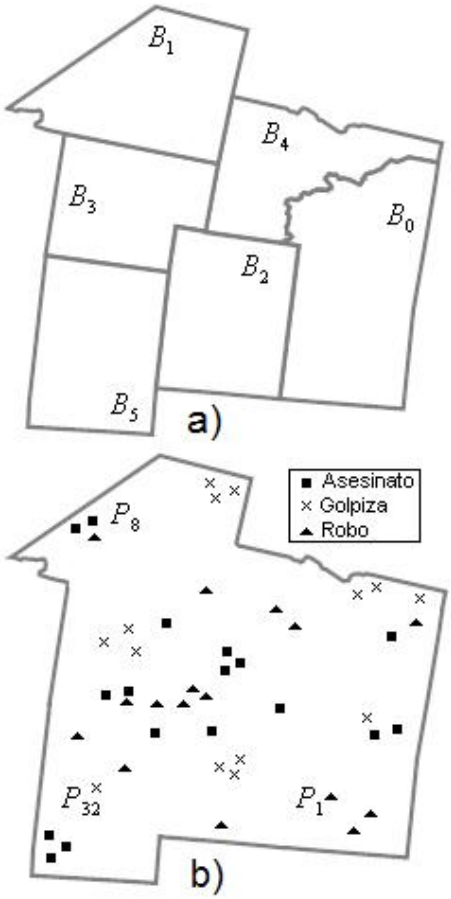

Figura 7. Mapa Crimenes: a) delimitación de los barrios y b) localización de los crimenes en la ciudad.

La consulta SQL generada implícitamente por el operador es:

```
SELECT Barrio, Crimen,
GEOMETRIC_UNION(TablaBarrio.Demarcacion) AS
'Demarcacion',
CENTER_OF_MASS(TablaCrimen.Localizacion)
AS 'Localizacion',
AVG(TablaCrimen.NroPer) AS 'PromVictimas'
FROM TablaBarrio, TablaCrimen
WHERE TablaBarrio.idBarrio = TablaCrimen.idBarrio
GROUP BY CUBE Barrio, Crimen
```

En la Figura 8 se muestran los grupos generados por el operador.

Figura 8. Cuboides generados por *map cube* para barrio y crimen.

En la Tabla 8 se presenta la agrupación para cada barrio y tipo de crimen junto con el promedio de víctimas, la localización puntual de la agrupación de los crímenes se presenta en la Figura 9. La Tabla 8 y la Figura 9 hacen parte del cuboide (Barrio y Crimen).

TABLA 8. GRUPO POR BARRIO Y CRIMEN

Barrio	*Crimen*	*Demarcación*	*Localización*	*PromVictimas*
B_0	Asesinato	DB_0	Q_1	1
B_0	Golpiza	DB_0	Q_2	5
B_0	Robo	DB_0	Q_3	1.5
B_1	Asesinato	DB_1	Q_4	1.5
B_1	Golpiza	DB_1	Q_5	2.33
B_1	Robo	DB_1	Q_6	2
B_2	Asesinato	DB_2	Q_7	2
B_2	Golpiza	DB_2	Q_8	5
B_2	Robo	DB_2	Q_9	1
B_3	Asesinato	DB_3	Q_{10}	2
B_3	Golpiza	DB_3	Q_{11}	6
B_3	Robo	DB_3	Q_{12}	1
B_4	Asesinato	DB_4	Q_{13}	3
B_4	Golpiza	DB_4	Q_{14}	3.5
B_4	Robo	DB_4	Q_{15}	6
B_5	Asesinato	DB_5	Q_{16}	2
B_5	Golpiza	DB_5	Q_{17}	10
B_5	Robo	DB_5	Q_{18}	1

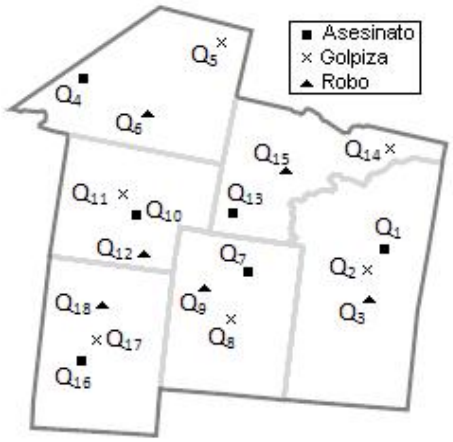

Figura 9. Resultados por Barrio y Crimen.

En la Figura 9 se observa que los crímenes en los barrios B_0, B_2, B_3 y B_5 tienden a concentrarse en un mismo lugar, mientras que en los barrios B_4 y B_1 están más dispersos.

La Tabla 9 y la Figura 10 corresponden al cuboide por (Barrio) y la Tabla 10 y la Figura 11 corresponden al cuboide por (Crimen).

TABLA 9. GRUPO BARRIO

Barrio	*Crimen*	*Demarcación*	*Localización*	*PromVictimas*
B_0	Todos	FB_0	R_0	2.25
B_1	Todos	FB_1	R_1	1.95
B_2	Todos	FB_2	R_2	2.67
B_3	Todos	FB_3	R_3	3
B_4	Todos	FB_4	R_4	4.17
B_5	Todos	FB_5	R_5	4,33

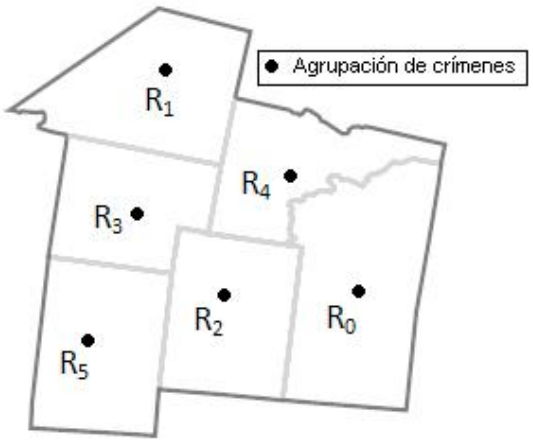

Figura 10. Resultado por barrio.

En la Figura 10 cada agrupación representada por R_x, simboliza un punto de partida para la implementación de políticas de seguridad en cada barrio.

TABLA 10. GRUPO CRIMEN

Barrio	*Crimen*	*Demarcación*	*Localización*	*PromVictimas*
Todos	Asesinato	GB_1	S_1	1.92
Todos	Golpiza	GB_2	S_2	5.31
Todos	Robo	GB_3	S_3	2.08

La Tabla 10 muestra la agrupación de cada tipo de crimen para toda la ciudad, representados en los puntos S_1, S_2 y S_3, los cuales se presentan en la Figura 11.

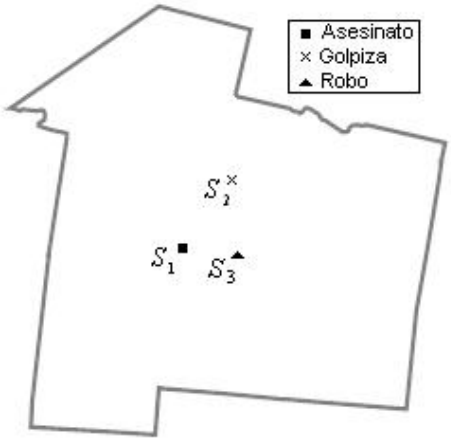

Figura 11. Resultado por crimen.

Los puntos en la Figura 11 indican el lugar con más densidad criminal para cada tipo de crimen dentro de la ciudad.

Finalmente en la Tabla 11 y en la Figura 12 se muestra el punto T_1 donde tienden a concentrarse los crímenes en la ciudad. Este resultado corresponde al cuboide (Todos).

TABLA 11. GRUPO TODOS

Barrio	*Crimen*	*Demarcación*	*Localización*	*PromPersonas*
Todos	Todos	DT_1	T_1	3.09

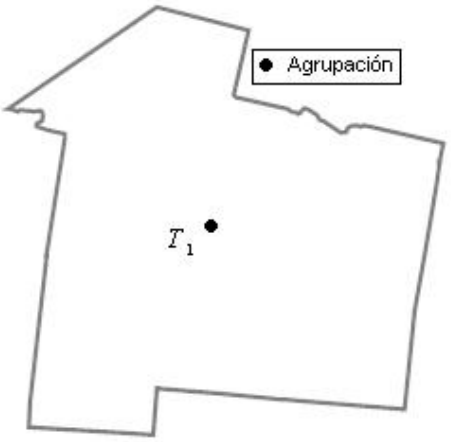

Figura 12. Resultado todos.

IV. CONCLUSIONES Y TRABAJO FUTURO

En este artículo se corrigieron algunas inconsistencias del operador *map cube* y se extendió su funcionalidad.

El principal aporte es permitir al usuario que elija las funciones de agregación espacial apropiadas para su contexto específico.

Se presentaron dos casos de estudio que demuestran la utilidad de las mejoras incorporadas. Los resultados permiten identificar comportamientos en los datos desde un *punto de*

vista espacial que de lo contrario serían difíciles de obtener.

Aunque la versión original del *map cube* ofrece facilidades de visualización (como color, grosor de las líneas, manejo de escala de grises, etc.) hacia el futuro se podrían incorporar más características. Por ejemplo adicionar un sistema de convenciones que permita al usuario especificar cómo desea que se dibujen determinadas áreas espaciales (manejo de entramados como en el ejemplo de los cultivos). Debe igualmente buscarse un punto de balance y no "saturar" el operador con muchas características de este tipo ya que éstas podrían ser especificadas por el usuario en una herramienta de visualización acoplada al operador.

Otro trabajo sería la incorporación de elementos temporales. Retomando el caso de estudio de los cultivos, supóngase que se tiene un registro histórico de las formas espaciales que han adoptado los cultivos año a año. Podría ser interesante para los administradores agrícolas observar los resultados generados por el operador *map cube* en diferentes años. Igualmente en este mismo caso de estudio, el mapa de las zonas también podría registrar una evolución e incidir en los resultados.

En la actualidad se está desarrollando un prototipo del operador. Se usa Oracle 10g Release 2 con su extensión espacial [9] Java y JSP. Aunque Oracle posee funciones para realizar agregación espacial y soporta el operador data cube (GROUP BY CUBE) [10], si se intentan usar estos dos aspectos en una misma sentencia SQL, se genera un error. Igualmente todo el trabajo relativo a la visualización y a las interfaces debe ser programado.

AGRADECIMIENTOS

Este trabajo es parte del proyecto "Modelo Multidimensional Espacio-Temporal y su correspondiente Lenguaje de Consulta", de la Convocatoria Nacional de Investigación 2007, Universidad Nacional de Colombia Sede Medellín y se desarrolla en el marco del Doctorado en Ingeniería de Sistemas de la misma universidad, auspiciado por Colciencias, del que el segundo autor es becario.

REFERENCIAS

[1] S. Shekhar, C. T. Lu, X. Tan, y S. Chawla. Map Cube: A Visualization Tool for Spatial Data Warehouses, 2002.

[2] http://www.spatial.cs.umn.edu/mapcube.htm

[3] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, H. Pirahesh, *Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub totals*, Data Mining and Knowledge Discovery 1(1), 1997, pp. 29-53.

[4] Peter Gulutzan y Trudy Pelzer, SQL-99 Complete Really, Primera Edición, R & B Books, Lawrence, Kansas, 1999.

[5] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In Proc. 1996 Int. Conf. Very Large Data Bases, 506-521, Bombay, India, Sept. 1996.

[6] S. Shekhar, S.Chawla, *Spatial databases: a tour*, Prentice Hall, 2003.

[7] J.R. Levine, T. Manson, y D. Brown. Lex and Yacc. O'Reilly And Associates, 1992.

[8] R. Sebesta, *Concepts of programming languages*, Pearson & Addison Wesley, 2005.

[9] Oracle®Spatial, User's Guide Reference 10g Release 2 (10.2) B14255-01, Junio 2005. Puede ser descargado de http://www.oracle.com/technology/products/spatial/index.html

[10] Oracle®Database Data Warehousing Guide 10g Release 2 (10.2) B14223-02. Diciembre 2005.

Francisco J. Moreno. Es MSc en Ingeniería de Sistemas de la Universidad Nacional de Colombia, Sede Medellín. Es candidato a Dr. en Ingeniería-Sistemas de la misma Universidad y es becario del programa de doctorados nacionales de Colciencias. Sus áreas de interés son las bodegas de datos.

Andrés F. Urrea. Es Ingeniero de Sistemas de la Universidad Nacional de Colombia, Sede Medellín. Sus áreas de interés son las bases de datos y la ingeniería de software.

Universidad Nacional de Colombia Sede Medellín
Facultad de Minas

Escuela de Ingeniería de Sistemas

Pregrado

- Ingeniería de Sistemas e Informática.

Posgrado

- Doctorado en Ingeniería-Sistemas.
- Maestría en Ingeniería de Sistemas.
- Especialización en Sistemas con énfasis en:
  - Ingeniería de Software.
  - Investigación de Operaciones.
  - Inteligencia Artificial.
- Especialización en Mercados de Energía.

Áreas de Investigación

- Ingeniería de Software.
- Investigación de Operaciones.
- Inteligencia Artificial.

Escuela de Ingeniería de Sistemas
Dirección Postal:
Carrera 80 No. 65 - 223 Bloque M8A
Facultad de Minas. Medellín - Colombia
Tel: (574) 4255350 Fax: (574) 4255365
Email: esistema@unalmed.edu.co
http://pisis.unalmed.edu.co/