

# Una comparación de dos lenguajes de consulta espacio-temporales: SQL<sup>ST</sup> y el de Güting

## Comparing two spatio-temporal query languages: SQL<sup>ST</sup> and Güting's language

Luis Miguel Pérez Montoya<sup>1</sup> y Francisco Javier Moreno Arboleda<sup>2</sup>

### RESUMEN

Las bases de datos espacio-temporales permiten representar objetos y fenómenos del mundo real que cambian de posición o forma a medida que transcurre el tiempo. En la última década se han propuesto diversos lenguajes de consulta para dicho tipo de bases de datos. En este artículo se comparan dos de estos lenguajes: SQL<sup>ST</sup> y el de Güting. La comparación se realiza por medio de criterios que se han aplicado a lenguajes de programación, pero que se han adaptado para evaluar lenguajes de consulta de bases de datos. Los resultados permiten concluir que el alto grado de expresividad evidenciado por ambos lenguajes puede afectar otros criterios, como la legibilidad y la simplicidad, especialmente en el caso del lenguaje de Güting.

**Palabras clave:** bases de datos espacio-temporales, lenguajes de consulta espacio-temporales, SQL, criterios de comparación.

### ABSTRACT

Spatio-temporal databases allow us to represent objects and phenomena from the real world which change position or shape as time elapses. Several query languages have been proposed during the last decade to deal with this type of database. Two of these languages have been compared in this paper: SQL<sup>ST</sup> and Güting's language. The comparison was based on criteria which have been applied to programming languages; however, they were adapted here to evaluate database query languages. The results led to concluding that both languages high degree of expressiveness may affect other criteria such as readability and simplicity, especially in the case of Güting's language.

**Keywords:** spatio-temporal database, spatio-temporal query language, SQL, comparison criteria.

Recibido: enero 21 de 2008

Aceptado: octubre 27 de 2008

### Introducción

Las bases de datos espacio-temporales tratan con objetos que cambian de posición o forma a medida que transcurre el tiempo (Worboys, 1994; Güting y Schneider, 2005). Estos objetos son adecuados para representar y registrar la evolución de planetas, propiedades urbanas, personas, animales, autos, barcos, aviones, huracanes, tumores, glaciares, entre otros.

Se han propuesto diversos lenguajes de consulta para este tipo de bases de datos, por ejemplo SQL<sup>ST</sup> (Chen y Zaniolo, 2000), el de Güting (Güting et al., 2000; Güting y Schneider, 2005), el de Chomicki (Chomicki y Revesz, 1999) y STQL (Erwig y Schneider, 2002). En el presente artículo se analizan los dos primeros mediante un conjunto de consultas y de criterios con el fin de evidenciar sus ventajas y desventajas. Los criterios empleados, véase sección 2, son aplicables, en general, a lenguajes de programación, sin embargo en este estudio se han orientado a lenguajes de consulta para bases de datos.

El artículo se estructura así: inicialmente se exponen los criterios para comparar los lenguajes; luego se describen los lenguajes de Güting y SQL<sup>ST</sup> y se presenta un caso de estudio en ambos lenguajes; finalmente se despliegan los resultados, se concluye el artículo y se plantean trabajos futuros.

### Criterios para comparar lenguajes de consulta

Para comparar ambos lenguajes se considera la siguiente lista de criterios (Haase et al., 2004; Sebesta, 2007). Sin embargo, tal lista es necesariamente controversial, ya que algunos criterios son vagos o no exactamente medibles; además, algunos son complementarios y otros pueden estar en conflicto.

- Expresividad (*expressiveness*): se refiere a si el lenguaje ofrece operaciones "poderosas", es decir, que realicen procesos complejos (Gilman y Rose, 1984; Turbak y Gifford, 2008) y que por lo tanto generan consultas concisas.
- Verbosidad (*writability*): trata de qué tan rápido, conciso y claramente se pueden expresar los procesos deseados en el lenguaje. "El lenguaje debe ofrecer un conjunto de abstracciones y construcciones que estén apropiadamente adaptadas a las habilidades mentales, hábitos y limitaciones del usuario" (Wirth, 1974).
- Legibilidad (*readability*): alude a qué tan fácil es comprender el significado de una consulta escrita en el lenguaje (Bonifati y Ceri, 2000). Se debe considerar si el lenguaje se acerca al lenguaje natural, si la consulta fue escrita en un lenguaje no diseñado para un propósito determinado (por ejemplo, programar operaciones matriciales en SQL). Además, hay aspectos que no dependen del lenguaje sino del programador: su grado de conocimiento del lenguaje y la posibilidad de escribir mal en un lenguaje (axioma de Flon (Flon, 1975)).

<sup>1</sup> Ingeniero de sistemas, Universidad Nacional de Colombia, Medellín. Imperezm@unal.edu.co

<sup>2</sup> M.Sc. y estudiante de doctorado en Ingeniería - Sistemas, Universidad Nacional de Colombia, Medellín. Profesor, Universidad Nacional de Colombia, fjmoreno@unal.edu.co

- d. Cierre (*closure*): se cumple cuando todas las operaciones retornan un elemento propio del modelo de datos que soporta al lenguaje. Por ejemplo, el álgebra relacional es cerrada porque todas sus operaciones retornan y trabajan con relaciones. Este aspecto facilita la escritura de expresiones anidadas y evita funciones de conversión (*casting*) y comportamientos inesperados.
- e. Seguridad (*safety*): implica que toda consulta retorna una cantidad finita de resultados (Korth *et al.*, 2005). Este criterio se relaciona con la confiabilidad (*reliability*): estrictamente una consulta debe hacer sólo lo que se le pide y no generar comportamientos inesperados ni ambigüedades. Los resultados han de ser correctos de acuerdo con la información disponible y las condiciones impuestas por el programador.
- f. Simplicidad (*simplicity*): se refiere a que las construcciones y la sintaxis del lenguaje deben ser fáciles de entender y no haber ambigüedades. Este criterio se relaciona con la uniformidad (*uniformity*): las construcciones similares del lenguaje exigen escribirse en forma similar. La simplicidad también se afecta por el número de construcciones esenciales que tiene un lenguaje y por el número de las que posee la misma función, lo que podría confundir a pesar de su utilidad (McIver y Conway, 1996). Por ejemplo, la condición de una reunión (*join*) en SQL se puede escribir usando las cláusulas INNER JOIN o WHERE.
- g. Portabilidad (*portability*): se refiere a qué tan independiente es el lenguaje de un producto (Oracle, SQL Server, DB2), es decir, el lenguaje se debe poder implantar en cualquier sistema de gestión de bases de datos (SGBD) y carecer de aspectos propietarios.

Con el fin de evaluar los lenguajes seleccionados frente a los criterios descritos, se proponen los siguientes mecanismos de medición:

-Encuesta: se pregunta a un programador, preferiblemente familiarizado con el lenguaje, su opinión o calificación sobre un criterio determinado,

-Conteo de casos: se ejecuta una consulta con diversas muestras de datos y se cuentan los comportamientos inesperados, ambiguos o que generen resultados infinitos; también se pueden contar las construcciones no uniformes, propietarias, expresivas, y las que violen el criterio de cierre, y

-Prueba de tiempo: se mide el tiempo que requiere un programador para realizar una actividad relacionada con el criterio a medir, tal como informar su comprensión de una consulta (en el caso de la legibilidad) o resolver un problema con el lenguaje (en el caso de la verbosidad).

La Tabla 1 presenta los mecanismos de medición que son aplicables a cada criterio de comparación.

Tabla 1. Mecanismos de medición vs. Criterios de comparación.

| Criterio de comparación | Mecanismo de medición |                 |                  |
|-------------------------|-----------------------|-----------------|------------------|
|                         | Encuesta              | Conteo de casos | Prueba de tiempo |
| Expresividad            | ✓                     | ✓               |                  |
| Verbosidad              | ✓                     |                 | ✓                |
| Legibilidad             | ✓                     |                 | ✓                |
| Cierre                  |                       | ✓               |                  |
| Seguridad               |                       | ✓               |                  |
| Simplicidad             | ✓                     | ✓               |                  |
| Portabilidad            |                       | ✓               |                  |

Para la comparación de lenguajes de consulta se pueden considerar también las siguientes observaciones: Darwen (Darwen, 2007) afirman que "la simple longitud de una expresión en caracteres puede ser muy significativa"; Mendelzon (Mendelzon y Vaisman, 2000) expresan que las consultas deben ser intuitivas, concisas y elegantes, y Date (Date, 2003) admite que la mayoría de las consultas temporales son muy difíciles de expresar en un lenguaje no especializado como SQL.

## Lenguajes de consulta espacio-temporales

### SQL<sup>ST</sup>

SQL<sup>ST</sup> (Chen y Zaniolo, 2000) es un lenguaje para el manejo de información espacio-temporal. Es una extensión minimalista del SQL estándar (Eisenberg *et al.*, 2004), ya que preserva su estructura y sólo añade tipos de datos temporales (DAY), espaciales (POINT, LINE y REGION) y operadores (ÁREA, OVERLAP, MOVING\_DISTANCE). SQL<sup>ST</sup> es una extensión de SQL<sup>T</sup> (Chen y Zaniolo, 1999), un lenguaje de consulta para el manejo de información temporal. El tiempo se representa discretamente, así los cambios de las geometrías, ya sea en forma o posición, son discretos. El lenguaje se ejemplifica más adelante.

### El lenguaje de Güting

En Güting *et al.* (2000) y Güting y Schneider (2005) se propone un lenguaje de consulta para el manejo de objetos móviles (se soportan cambios continuos de las geometrías, ya sea en forma o posición). Se proponen tipos de datos móviles como MBOOL, MINTEGER, MPOINT, MLINE y MREGION, que combinan los tipos de datos primitivos (BOOLEAN, INTEGER) o espaciales (POINT, LINE y REGION) con información temporal. El lenguaje es de tipo SQL, aunque presenta un carácter procedural ya que posee construcciones como la asignación de variables. El lenguaje se ejemplifica en la más adelante.

### Ejemplos

A continuación se exhibe un caso de estudio y cuatro consultas en ambos lenguajes.

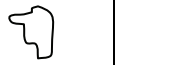

#### Caso de estudio

Considérese un escenario de incendios forestales (Güting *et al.*, 2000; Chen y Zaniolo, 2000). Se maneja información sobre bosques, incendios y bomberos. Sean las relaciones bosque, incendio y bombero. Una muestra de datos se despliega en las Tablas 2, 3 y 4 para SQL<sup>ST</sup> y en las tablas 2, 5, y 6 para el lenguaje de Güting.

En SQL<sup>ST</sup>:

-CREATE TABLE bosque (nombre CHAR(30), territorio REGION)







Tabla 2. Muestra de datos de la relación bosque para SQL<sup>ST</sup> y para el lenguaje de Güting

| Bosque      |   |
|-------------|---|
| Nombre      | Territorio  |
| Verde       |  |
| Santa Elena |  |

-CREATE TABLE incendio (nombre CHAR(30), extensión REGIÓN, día DAY)

En la Tabla 3 se muestra el valor del área de cada incendio (en km<sup>2</sup>). Este valor, aunque no es un atributo de la relación bosque, se puede obtener por medio de la función AREA de SQL<sup>ST</sup>.

Tabla 3. Muestra de datos de la relación incendio para SQL<sup>ST</sup>

| Incendio |  |         |
|----------|--|---------|
| Nombre   | Extensión  | Día     |
| Gran L   | <br>(Área: 315) | 5/7/06  |
| Gran L   | <br>(Área: 503) | 6/7/06  |
| Gran L   | <br>(Área: 503) | 7/7/06  |
| Gran L   | <br>(Área: 114) | 7/7/06  |
| Azul     | <br>(Área: 710) | 24/3/07 |
| Azul     | <br>(Área: 480) | 25/3/07 |

-CREATE TABLE bombero (nombre CHAR(30), ubicación POINT, día DAY)

El atributo ubicación representa la localización puntual del bombero durante un día en el plano bidimensional. Dichos puntos se representan con letras en la Tabla 4 para facilitar su referencia posterior.

Tabla 4. Muestra de datos de la relación bombero para SQL<sup>ST</sup>

| Bombero    |            |        |
|------------|------------|--------|
| Nombre     | Ubicación  | Día    |
| T. Montoya | A (31,52)  | 4/7/06 |
| T. Montoya | B (40,47)  | 6/7/06 |
| T. Montoya | C (34, 24) | 7/7/06 |
| J. Vélez   | D (21,75)  | 6/8/07 |
| J. Vélez   | E (30, 66) | 7/8/07 |
| J. Vélez   | F (34, 41) | 9/8/07 |

En el lenguaje de Güting:

-Bosque (nombre: STRING, territorio: REGION)

La muestra de datos de la relación bosque es la misma que la de la Tabla 2.

-Incendio (nombre: STRING, extensión: MREGION)

-Bombero (nombre: STRING, ubicación: MPOINT)

Tabla 5. Muestra de datos de la relación incendio para el lenguaje de Güting







| Incendio |   |
|----------|---|
| Nombre   | Extensión   |
| Gran L   | {(5/7/06,  ) , (6/7/06,  ) ,<br>(7/7/06,  ) , (7/7/06,  ) } |
| Azul     | {(24/3/07 ,  ) , (25/3/07 ,  ) }  |

Tabla 6. Muestra de datos de la relación bombero para el lenguaje de Güting

| Bombero    |   |
|------------|---|
| Nombre     | Ubicación                                     |
| T. Montoya | {( 4/7/06, A ), (6/7/06 , B ), (7/7/06 , C )} |
| J. Vélez   | {( 6/8/07, D ), (7/8/07, E ), (9/8/07, F )}   |

En la Tabla 6 se representa la ubicación puntual de cada bombero mediante letras. Sus coordenadas son las mismas de las de la Tabla 4.

**Consultas**

A partir de las tablas de la sección anterior se plantean las siguientes consultas; se resuelven primero en SQL<sup>ST</sup> y luego en el lenguaje de Güting.

Consulta 1: ¿Cuándo y dónde alcanzó el incendio “Gran L” su máxima extensión?

Consulta 2: ¿Cuándo y dónde se expandieron los incendios más de 500 km<sup>2</sup>?

Consulta 3: ¿Cuánto tiempo estuvo el bombero “T. Montoya” dentro del incendio “Gran L” y cuánta distancia recorrió dentro del mismo?

Consulta 4: Determinar el momento y el lugar donde se inició el incendio “Gran L”.

**Consulta 1. Una respuesta en SQLST tomada textualmente de (Chen y Zaniolo, 2000) es:**



```
SELECT F1.día, F2.extensión, AREA(F1.extensión)
FROM incendio AS F1 F2
WHERE F1.nombre = "Gran L" AND F2.nombre = "Gran L" AND
F1.día = F2.día
GROUP BY F1.día
HAVING AREA (F1.extensión) = (SELECT MAX(AREA(extensión))
FROM incendio
WHERE nombre = "Gran L")
```

Nótese que el atributo F2.extensión aparece en la cláusula SELECT pero no en la cláusula GROUP BY. Esto es válido porque el atributo de agrupamiento F1.día lo determina funcionalmente. Por otra parte, en la cláusula HAVING aparece la función ÁREA, que no es una función de grupo, lo que es inválido en un lenguaje tipo SQL.

En la cláusula FROM se reúne la relación incendio consigo misma (self-join). En la cláusula WHERE se eliminan las tuplas que no tie-

nen la misma fecha en el atributo día y las que no tienen el valor "Gran L" en el atributo nombre (SQL<sup>ST</sup> y el lenguaje de Güting usan comillas dobles para las cadenas de caracteres, a diferencia de SQL, que utiliza comillas simples). El resultado de la consulta se muestra en la Tabla 7.

Tabla 7. Resultado de la consulta 1 en SQL<sup>ST</sup>

| F1.día | F2.extensión  | AREA |
|--------|---|------|
| 6/7/06 |  | 503  |
| 7/7/06 |  | 503  |

La solución presentada en (Chen y Zaniolo, 2000) es más compleja de lo necesario. Se pueden evitar el *self-join* y las cláusulas GROUP BY y HAVING, así:

```
SELECT día, extensión, ÁREA(extensión)
FROM incendio
WHERE nombre = "Gran L" AND AREA(extensión) = (SELECT
MAX(ÁREA(extensión))
FROM incendio
WHERE nombre = "Gran L")
```

#### Consulta 1. En el lenguaje de Güting:

- i) LET GranL = ELEMENT(SELECT extensión
FROM incendio
WHERE nombre = "Gran L");
- ii) LET max\_area = INITIAL(ATMAX(ÁREA(GranL)));
- iii) ATINSTANT(GranL, INST(max\_area)); VAL(max\_area)

La consulta se compone de tres partes. En la primera, la función ELEMENT convierte el resultado en un valor atómico y lo asigna a la variable GranL mediante el operador LET; por lo tanto, la variable GranL contiene el atributo extensión, de tipo MREGION, del incendio "Gran L". ELEMENT sólo se puede usar en una consulta que devuelva una tupla y un atributo. El contenido de GranL se muestra en la Figura 1.





|  |
|--|
| {(5/7/06,  ) , (6/7/06,  ) , |
| (7/7/06,  ) , (7/7/06,  ) }  |

Figura 1. Contenido de GranL

En la segunda parte se busca la mayor área en GranL. Si hay empates en el valor máximo, se escoge el de fecha más antigua usando la función INITIAL. El resultado se asigna a la variable max\_area. La consulta 1 presentada en SQL<sup>ST</sup>, en contraste, retorna todos los casos (empates) en los que el área fue la máxima (véase la Tabla 7).

En la tercera parte se retornan los resultados:

-ATINSTANT(GranL, INST(max\_area)): retorna el día en que el área del incendio fue máxima. Se retorna el 6/7/06 (aunque el 7/7/06 también se alcanzó el área máxima).

-VAL(max\_area): retorna el valor (área) de max\_area (503 km<sup>2</sup>).

El resultado de la consulta se muestra en la Tabla 8.

Tabla 8. Resultado de la consulta 1 en el lenguaje de Güting.




| ATINSTANT | VAL |
|-----------|-----|
| 6/7/06    | 503 |

#### Consulta 2. Una respuesta en SQL<sup>ST</sup> tomada textualmente de (Chen y Zaniolo, 2000) es:

```
SELECT F1.día, F2.extensión
FROM incendio AS F1 F2
WHERE F1.día = F2.día AND F1.nombre = F2.nombre
GROUP BY F1.día, F2.extensión, F1.nombre
HAVING AREA(F1.extensión) > 500
```

De nuevo se efectúa un *self-join* de la relación incendio. La cláusula WHERE elimina las tuplas que no cumplen la condición F1.día = F2.día y F1.nombre = F2.nombre y la cláusula HAVING selecciona las tuplas cuya área de F1.extensión es mayor que 500 km<sup>2</sup>. El resultado de la consulta se muestra en la Tabla 9.

Tabla 9. Resultado de la consulta 2 en SQL<sup>ST</sup>

| F1.día  | F2.extensión  |
|---------|---|
| 6/7/06  |  (Área: 503)   |
| 7/7/06  |  (Área: 503)  |
| 24/3/07 |  (Área: 710) |

Esta consulta también se puede simplificar. Se evitan el *self-join* y las cláusulas GROUP BY y HAVING, así:

```
SELECT día, extensión
FROM incendio
WHERE AREA(extensión) > 500
```

#### Consulta 2. En el lenguaje de Güting:

- i) LET reg\_grande = SELECT a\_grande AS extensión WHEN(FUN(r:region) AREA(r) > 500)
FROM incendio;
- ii) SELECT \*
FROM reg\_grande
WHERE NOT(ISEMPTY(DEFTIME(a\_grande)));




La primera parte de la consulta reduce la MREGION (atributo extensión) de cada incendio a las parejas (tiempo, región) en las que el área de la región es mayor que 500 km<sup>2</sup>. Para ello se usa el operador WHEN, que restringe un valor que dependa del tiempo a los periodos en que cumple cierta condición. El argumento de WHEN es un predicado que expresa la restricción. Aquí el predicado se construye mediante el operador FUN así: FUN <argumentos> <expresión>. Por ejemplo, la siguiente función calcula el cuadrado de un número:

```
LET square = FUN (m:integer) m * m;
```

Así, square(5) es igual a 25 (FUN es similar a la directiva #DEFINE del lenguaje C).

Nótese que en el lenguaje de Güting el orden de los argumentos del operador de renombrado AS es contrario a como se realiza en SQL. Por lo tanto, en el atributo a\_grande, se tienen las MREGION de los incendios luego de realizar la restricción. Este resultado se guarda en la variable reg\_grande, como se muestra en la Tabla 10.

Tabla 10. Contenido de reg\_grande

| reg_grande  |
|---|
| a_grande  |
| {(6/7/06,  , (7/7/06,  )} |
| {(24/3/07,  )}   |





La segunda parte de la consulta considera si un incendio nunca alcanzó más de 500 km<sup>2</sup>. Para detectar estos casos, la cláusula WHERE revisa las fechas de cada incendio. Para ello se usa la función DEFTIME; esta retorna los tiempos en los que un objeto móvil está definido. En otras palabras, al aplicar DEFTIME a a\_grande, la función recorre la MREGION de cada incendio y extrae el tiempo de cada pareja (tiempo, región). El resultado de la consulta corresponde al de la Tabla 10.

**Consulta 3. Una respuesta en SQL<sup>ST</sup> tomada textualmente de (Chen y Zaniolo, 2000) es:**

```
SELECT DURATION (bombero. día),
        MOVING_DISTANCE (bombero.ubicación,
                          bombero.día)
FROM incendio, bombero
WHERE incendio.día = bombero.día AND
      incendio.nombre = "Gran L" AND
      bombero.nombre = "T. Montoya"
GROUP BY incendio.día
HAVING INSIDE (bombero.ubicación, incendio.extensión)
```

A partir del producto cartesiano entre las relaciones incendio y bombero, la cláusula WHERE selecciona las tuplas en las que el nombre del bombero es "T. Montoya", el nombre del incendio es "Gran L" y las fechas del incendio y del bombero coinciden. Hasta aquí se obtiene el resultado de la Tabla 11.

Tabla 11. Resultado parcial de la consulta 3 en SQL<sup>ST</sup>

| Incendio |   |        | Bombero    |           |        |
|----------|---|--------|------------|-----------|--------|
| Nombre   | Extensión   | Día    | Nombre     | Ubicación | Día    |
| Gran L   |  | 5/7/06 | T. Montoya | A         | 5/7/06 |
| Gran L   |  | 6/7/06 | T. Montoya | B         | 6/7/06 |
| Gran L   |  | 7/7/06 | T. Montoya | C         | 7/7/06 |
| Gran L   |  | 7/7/06 | T. Montoya | C         | 7/7/06 |

La cláusula HAVING elimina la tupla sombreada de la Tabla 11, así: se usa la función INSIDE, que toma como argumentos un punto y una región y retorna un valor lógico que indica si el punto se encuentra en la región.

La función INSIDE no es una función de grupo, por lo tanto, su uso en una cláusula HAVING es inválida en un lenguaje tipo SQL. La consulta se puede corregir, así:

```
SELECT DURATION (bombero. día),
        MOVING_DISTANCE (bombero.ubicación,
                          bombero.día)
FROM incendio, bombero
WHERE incendio.día = bombero.día AND
      incendio.nombre = "Gran L" AND
      bombero.nombre = "T. Montoya" AND
      INSIDE (bombero.ubicación, incendio.extensión)
```

De acuerdo con la muestra de datos, Tablas 3 y 4, supóngase que los datos espaciales se encuentran distribuidos como se muestra en la Figura 2, para el bombero "T. Montoya" y el incendio "Gran L".

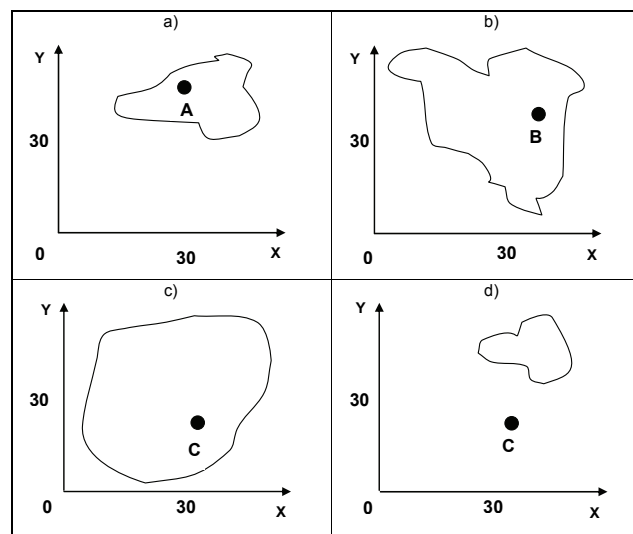


Figura 2. Ubicación del bombero "T. Montoya" y del incendio "Gran L": a) punto A y región de 315 km<sup>2</sup> el 5/7/06, b) punto B y región de 503 km<sup>2</sup> el 6/7/06, c) punto C y región de 503 km<sup>2</sup> el 7/7/06, y d) punto D y región de 114 km<sup>2</sup> el 7/7/06.

La cláusula SELECT retorna dos valores:

-DURATION: a partir de los datos que recibe como argumento, días en el ejemplo, retorna el tiempo total transcurrido entre ellos.

-MOVING\_DISTANCE: calcula la distancia entre los puntos que recibe como argumento. El cálculo se realiza en orden de acuerdo con el atributo temporal para dar la distancia total recorrida al sumar estas distancias; es decir, la distancia entre el primer punto (cuya fecha es menor) y el segundo (cuya fecha es la segunda menor) más la distancia entre el segundo y el tercer puntos y así sucesivamente, como se muestra en la Figura 3.

El resultado de la consulta se muestra en la Tabla 12.

Tabla 12. Resultado de la consulta 3 en SQL<sup>ST</sup>

| DURATION | MOVING_DISTANCE |
|----------|-----------------|
| 3        | 34.1            |

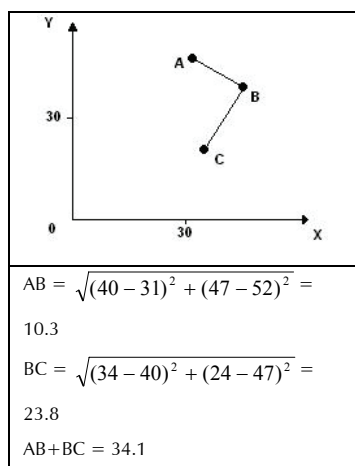


Figura 3. Ejemplo de la función MOVING\_DISTANCE

**Consulta 3. En el lenguaje de Güting:**

```

SELECT tiempo AS
DURATION(DEFTIME(INTERSECTION(ubicación, GranL))),
      distancia AS LENGTH (TRAJECTORY
      (INTERSECTION(ubicación, GranL)))
FROM bombero
WHERE nombre = "T. Montoya";

```

De la relación bombero, se seleccionan las tuplas en las que el nombre del bombero es "T. Montoya" (se usa la variable GranL definida en la consulta 1).

En la cláusula SELECT, se calcula el tiempo durante el cual el MPOINT del bombero estuvo dentro de la MREGION del incendio, así: INTERSECTION(ubicación, GranL) halla las intersecciones espaciales entre el MPOINT ubicación y la MREGION GranL. El resultado es un MPOINT. Esta intersección se muestra en la Figura 2 y el MPOINT resultante en la Figura 4. Luego, la función DEFTIME extrae los tiempos para los que existe intersección. Estos se obtienen a partir de las parejas que conforman el MPOINT resultante. Finalmente, se calcula el tiempo que estuvo el bombero dentro del incendio con la función DURATION, que calcula el tiempo transcurrido a partir del resultado de DEFTIME.

{(5/7/06, A), (6/7/06, B), (7/7/06, C)}

Figura 4. MPOINT resultante de INTERSECTION(ubicación, GranL).

En la cláusula SELECT también se calcula la longitud de la trayectoria que recorrió el bombero dentro del incendio. Para lograrlo, la función TRAJECTORY proyecta el MPOINT de la Figura 4 sobre el plano bidimensional y retorna una polilínea que une los puntos A, B y C. Por último, la función LENGHT calcula la longitud de la polilínea, como se muestra en la Figura 3.

El resultado es igual al de la Tabla 12, excepto por el nombre de los atributos.

**Consulta 4. En SQL<sup>ST</sup>:**

```

SELECT día, extensión
FROM incendio
WHERE nombre = "Gran L" AND día = (SELECT MIN(día)
FROM incendio
WHERE nombre = "Gran L")

```

Nótese que si se encuentra más de una tupla con la fecha más antigua para el incendio "Gran L", se retornan todas estas. El resultado de la consulta se muestra en la Tabla 13.

Tabla 13. Resultado de la consulta 4 en SQL<sup>ST</sup>

| Día    | Extensión |
|--------|-----------|
| 5/7/06 |           |

**Consulta 4. En el lenguaje de Güting:**

Para responder a esta consulta se procede así:

-INST(INITIAL(GranL)): retorna la fecha más antigua de la MREGION GranL.

-VAL(INITIAL(GranL)): retorna el valor (es decir la región) de la MREGION GranL que está asociado con la fecha más antigua.

El resultado de la consulta es igual al de Tabla 13.

**Resultados**

Se encuestó a un grupo de veinte programadores familiarizados con SQL, al que se le presentaron las características esenciales de los dos lenguajes, las consultas realizadas y los criterios. Los criterios se midieron con un puntaje entre 0 y 5. Los resultados de la comparación se muestran en la Tabla 14.

Tabla 14. Comparación entre SQL<sup>ST</sup> y el lenguaje de Güting

| Criterio     | SQL <sup>ST</sup>   | Lenguaje de Güting  |
|--------------|---|---|
| Expresividad | Existen operadores "poderosos" como DURATION, INSIDE y MOVING_DISTANCE. Puntaje: 4                | Existen operadores "poderosos" como ATINSTANT, VAL, DEFTIME y TRAJECTORY. Puntaje: 4.3  |
| Verbosidad   | Similar a la del SQL estándar. Puntaje: 4.5   | Aunque algunas consultas simples son más largas que sus correspondientes en SQL <sup>ST</sup> , la asignación de variables simplifica el planteamiento de otras. Puntaje: 4.2 |
| Legibilidad  | Se acerca bastante al SQL estándar. Los nuevos operadores son fáciles de comprender. Puntaje: 4.4 | La asignación de variables dificulta la comprensión de algunas consultas. Además, posee muchas funciones y a veces su significado no es evidente. Puntaje: 3.8                |
| Cierre       | Es consecuente con su modelo de datos dado que sólo retorna relaciones. Puntaje: 4.7              | Tiene algunos operadores, como ELEMENT, que tienden a violar el cierre. Puntaje: 4.5  |
| Seguridad    | No genera resultados infinitos. Puntaje: 4.9  | No genera resultados infinitos. Puntaje: 4.9  |
| Simplicidad  | Similar a la del SQL estándar. Puntaje: 4.6   | Su aspecto procedural le hace menos simple que SQL <sup>ST</sup> . Puntaje: 4   |
| Portabilidad | Carece de aspectos propietarios. Puntaje: 4.8   | Carece de aspectos propietarios. Puntaje: 4.8   |

**Conclusiones y trabajo futuro**

En este artículo se compararon dos lenguajes de consulta espacio-temporales: SQL<sup>ST</sup> y el de Güting. Para hacer la comparación se adaptaron criterios que usualmente se han aplicado a lenguajes de

programación. Los resultados evidenciaron, por ejemplo, que el alto grado de expresividad de ambos lenguajes puede afectar otros criterios como la legibilidad y la simplicidad, especialmente en el caso del lenguaje de Güting.

Adicionalmente, se corrigieron y simplificaron algunas consultas en SQL<sup>ST</sup> planteadas en la fuente original (Chen y Zaniolo, 2000). Las simplificaciones fueron esenciales para no crear en el grupo de programadores encuestados una falsa impresión de complejidad de dicho lenguaje.

Se planea identificar tipos de consultas espacio-temporales que sean difíciles o imposibles de expresar en SQL<sup>ST</sup> y en el lenguaje de Güting, con el fin de concebir operadores que faciliten su planteamiento. También se pretende incorporar en SGBD como Oracle y DB2, los lenguajes analizados. Para ello se espera aprovechar las características espaciales y objeto-relacionales de estos SGBD, analizar las dificultades de dicha implantación y realizar pruebas de rendimiento.

## Agradecimientos

Este artículo se desarrolló en el marco del proyecto "Modelo multi-dimensional espacio-temporal y su correspondiente lenguaje de consulta y del doctorado en Ingeniería-Sistemas de la Universidad Nacional de Colombia, Medellín, del cual el segundo autor es becario de Colciencias.

## Bibliografía

- Bonifati, A., Ceri, S., Comparative Analysis of Five XML Query Languages., SIGMOD Record, Vol. 29, No.1, 2000, pp. 68-79.
- Chen, C., Zaniolo, C., Universal Temporal Extensions for Database Languages., 15th International Conference on Data Engineering, IEEE Computer Society Press, May, 1999, pp. 428-437.
- Chen, C., Zaniolo, C., SQLST: A Spatio-Temporal Data Model and Query Language., Conceptual Modeling - ER 2000, 19th International Conference on Conceptual Modeling., October, 2000, pp. 96-111.
- Chomicki, J., Revesz, P. Z., A Geometric Framework for Specifying Spatiotemporal Objects, 6th International Workshop on Temporal Representation and Reasoning., Orlando, IEEE Computer Society, May, 1999, pp. 41-46.
- Darwen, H., Comunicación privada, Diciembre, 2007.
- Date, C. J., An Introduction to Database Systems., 8a ed., Nueva York, Addison-Wesley, 2003.
- Eisemberg, A., Melton, J., Kulkarni, K. G., Michels, J. E., Zemke, F., SQL: 2003., SIGMOD Record., Vol. 33, No. 1, 2004, pp. 119-126.
- Erwig, M., Schneider, M., STQL: A Spatio-Temporal Query Language., En: Mining Spatio-Temporal Information Systems (eds. R. Ladner, K. Shaw, M. Abdelguerfi), Kluwer Academic Publishers, 2002, pp. 105-126.
- Flon, L., On Research in Structured Programming., ACM SIGPLAN Notices, Vol. 10, No. 10, 1975, pp. 16-17.
- Gilman, L., Rose, A. J., APL: an Interactive Approach., 3a ed., Washington, Wiley, 1984.
- Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., Varzirgiannis, M., A Foundation for Representing and Querying Moving Objects., ACM Transactions on Databases Systems, Vol. 25, No. 1, 2000, pp. 1-42.
- Güting, R. H., Schneider, M., Moving Objects Databases., Oxford, Morgan Kaufmann, 2005.
- Haase., P., Broekstra, J., Eberhart, A., Volz, R., A Comparison of RDF Query Languages., The Semantic Web – ISWC, 3rd International Semantic Web Conference, November, 2004, pp. 502-517.
- Korth, H. F., Silberschatz, A., Sudarshan, S., Database Systems Concepts., 5a ed., Nueva York, McGraw-Hill, 2005.
- McIver, L., Conway, D., Seven Deadly Sins of Introductory Programming Language Design., SEEP '96, International Conference on Software Engineering: Education and Practice, January, 1996, pp. 1-8.
- Mendelzon, A., Vaisman, A. Temporal Queries in OLAP., 26th International Conference on Very Large Data Bases, Morgan Kaufmann, September, 2000, pp. 242-253.
- Sebesta, R. W., Concepts of Programming Languages., 7a ed., Washington, Addison Wesley, 2007.
- Turbak, F. A., Gifford, D. K., Design Concepts in Programming Languages., 1a ed., Boston, The MIT Press, 2008.
- Wirth, N., On the Design of Programming Languages., Information Processing 74, North-Holland Publishing Co., 1974, pp. 386-393.
- Worboys, M. F., A Unified Model for Spatial and Temporal Information., The Computer Journal, Vol. 37, No.1, 1994, pp. 26-34.