Universidad Nacional de Rosario

Monografía
Bases de Datos Avanzadas

Bases de Datos Espaciales y Temporales

Autores:
Federico Badaloni
David Giordana

Docentes:
Claudia Deco
Cristina Bender

Departamento de Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Av. Pellegrini 250, Rosario, Santa Fe, Argentina

13 de octubre de 2020

Resumen

El resumen de la monografía.

Índice general

Capítulo 1

Introducción

La idea general es presentar espaciales como algo practica-¿teoria, temporales como teoria-¿practica y despues espacio temporales como una mezcla de ambas çulturas".

Intro hablando sobre el marco general de la mono, detallando que va a tratar cada capitulo y como esta estructurada

En esta trabajo, se presentan conceptos generales y el estado actual de la investigación científica en el área de las Bases de Datos Espaciales, Temporales y Espacio-Temporales.

En el capitulo 1 se presentan conceptos sobre Bases de Datos Espaciales, antecedentes importantes en el procesamiento de datos espaciales como los sistemas GIS, los modelos teóricos que se usan actualmente, diversas aplicaciones y generalizaciones teóricas que nos serán útiles para combinarlas con modelos temporales.

En el capítulo 2 se presentan modelos de Bases de Datos Temporales, las distintas formas de modelarlas teóricamente, el consenso obtenido entre distintos modelos y posterior extensión al estándar SQL y se compara contra el proceso de desarrollo de las Espaciales.

En el capítulo 3 se examina como de la combinación de ambas ramas de investigación y desarrollo surgen las Bases de Datos Espacio-Temporales, las distintas tendencias heredadas de las diferencias historicas entre Temporales y Espaciales que derivan en una variedad de modelos especializados para aplicaciones especificas y se concluye con un análisis de como los autores de este trabajo ven la posible evolución de esta area

Capítulo 2

Bases de Datos Espaciales

2.1. Introducción

Los sistemas de bases de datos espaciales (SDBMS) son los DBMS que incorporan capacidades de representación y manipulación de datos geométricos en un marco de referencia dado.

La principal aplicación de los SDBMS son los Sistemas de Información Geográfica (GIS): software que provee mecanismos de análisis y visualización de de datos geográficos. Los datos geográficos son datos espaciales cuyo marco de referencia es la superficie terrestre. Los GIS implementan en sus herramientas un gran conjunto de técnicas desarrolladas por cartógrafos y que son previas al desarrollo de la informática. Este hecho dota a la investigación de las Bases de Datos Espaciales de un *carácter multidisciplinario* que puede señalarse como una de las causas del rápido avance de la misma.

A su vez, existen otras aplicaciones para los SDBMS tales como la representación de circuitos, datos astronómicos, moléculas y muchas otras. En general, los modelos teóricos son los mismos y lo que cambia es simplemente el marco de referencia en el que se representan los datos. En este trabajo haremos foco en las aplicaciones a los GIS, pero el lector no debe perder de vista que todos los conceptos discutidos son aplicables a estas otras áreas.

2.2. Modelos Teóricos

El desarrollo de modelos teóricos para las Bases de Datos Espaciales debe entenderse en el contexto histórico discutido en la sección anterior. Los primeros GIS implementaban todas las operaciones sobre datos espaciales a nivel de aplicación, guardando sus datos en bases de datos convencionales. A su vez, los primeros SDBMS intentaron simplemente mover esta lógica al nivel de base de datos, pero sin desarrollar un marco teórico. Posteriormente, algunos investigadores comenzaron a dar definiciones formales de **Tipos de Datos Espaciales** (SDTs).

En las secciones 2.2.1 y 2.2.2 analizaremos brevemente el Álgebra de RoSE [3] (Robust Spatial Extension), un modelo formal que consideramos uno de los más elegantes

y significativos del campo. Luego, en las secciones 2.2.3 y 2.2.4 veremos los principales paradigmas de modelado que se utilizan en los SDBMS actuales, el modelo funcional y el modelo basado en objetos.

2.2.1. Realms

Según Güting y Schneider [3], un modelo formal para Bases de Datos Espaciales debe ser:

- General: los objetos geométricos usados como valores de SDTs deben ser tan generales como sea posible. Por ejemplo, un valor región debe poder representar una colección de áreas disjuntas, cada una de las cuales puede tener agujeros. Mas precisamente, los dominios de los tipos de datos punto, línea y región deben ser cerrados respecto a la unión, intersección y diferencia de sus conjuntos de puntos subyacentes.

- Riguroso: la semántica de los SDTs, es decir, los posibles valores para los tipos y las funciones asociadas con las operaciones, deben estar definidas formalmente para evitar ambigüedades para el usuario y el implementador.

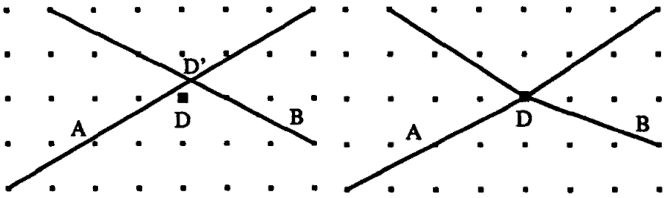
- De resolución finita: las definiciones formales deben tener en cuenta las capacidades de representación finitas de las computadoras Delegarle al programador la responsabilidad de cerrar la brecha entre teoría y práctica en este punto lleva a errores numéricos y topológicos.

- Geométricamente consistente: distintos objetos espaciales pueden estar pueden estar relacionados mediante restricciones geométricas. Las definiciones de los SDTs ayudar a mantener esa consistencia.

El Álgebra de RoSE se basa en la incorporación a los DMBSs del concepto de realm, un conjunto finito de puntos y segmentos sin intersecciones sobre los cuales se posicionan todos los datos espaciales de una base de datos. Con este enfoque, los datos no se crean al darle valor a un atributo, sino que se *seleccionan* del conjunto de valores existentes en el realm. Este modelo permite asegurar que el álgebra espacial es cerrada respecto a un realm. Es decir, que la primitivas geométricas y las operaciones sobre realms se definen sobre aritmética entera, libre de errores de redondeo que podrían aparecer si el modelo se definiera sobre un espacio euclídeo.

Otra ventaja de los realms es que permiten aislar todo el computo de puntos en las operaciones de actualización del realm. No se computan intersecciones durante consultas de búsqueda.

Para mapear los segmentos con intersecciones de una aplicación a un conjunto de segmentos sin intersecciones de un realm, se utiliza redibujado y geometría de resolución finita [2].



(a) (b)

Figura 2.1: Redibujado de la intersección de dos segmentos.

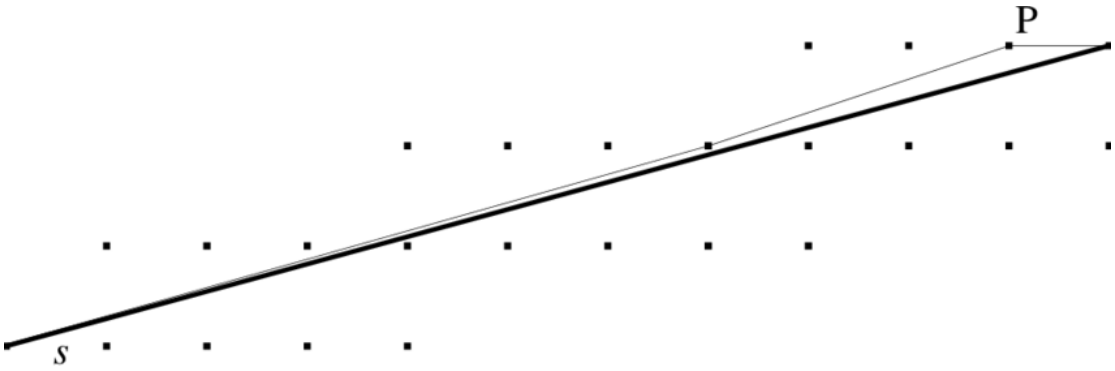


Figura 2.2: Envoltura de un segmento.

2.2.2. Actualizaciones en realms

Veamos mas en detalle que ocurre en este modelo cuando se realiza una actualización que requiere la insercion de un nuevo segmento. Esto es importante porque los datos geométricos provenientes de la aplicación no son no-intersecantes como requiere el modelo. El problema fundamental es que usualmente los puntos de intersección no se corresponden con puntos de la grilla del realm. La solución es aplicar redibujado: se modifican los segmentos de forma tal que la intersección se mueva al punto mas cercano que sí esté en la grilla.

Esto puede verse en las figuras 2.1a y 2.1b: la intersección entre los segmentos A y B (D') no se corresponde con un punto de la grilla, pero podemos partir a cada uno de ellos en dos, de manera que todos los segmentos resultantes tengan a Dcomo uno de sus extremos.

Pero ahora nos surge un nuevo problema que es la preocupación de que aplicando esta operación sucesivamente los segmentos se modifiquen cada vez mas, introduciendo cada vez mas error en la representacion. Podemos acotar este error que se introduce, usando el concepto de envoltura. La envoltura de un segmento es el conjunto de puntos de la grilla que se encuentran inmediatamente arriba, debajo o sobre el segmento. Si establecemos entonces la restriccion de que los puntos a usar para el redibujado deben caer sobre la envoltura del segmento, tenemos una cota para el error: la distancia entre puntos en la grilla. La figura 2.2 muestra un ejemplo de envoltura de un segmento.

De esta forma, Güting y Schneider definen un conjunto de operaciones geométricas primitivas robustas con error acotado que son la base para, a través de la composición de las mismas, definir de forma rigurosa todos los SDTs y las operaciones geométricas que nos pueden resultar interesantes para la representación de datos espaciales. Si bien el Álgebra de RoSE fue un hito en términos de rigurosidad teórica en los modelos de bases de datos espaciales, no es utilizada actualmente en ninguno de los SDBMSs actualmente en uso por la industria del software. De todas formas, su influencia puede verse en los modelos que se utilizan actualmente y que veremos a continuación.

2.2.3. Modelo Funcional

En esta sección, veremos el modelo funcional o modelo orientado a campos. Comenzaremos presentando un ejemplo de un problema espacial a modelar. Supongamos un parque nacional que cuenta con un conjunto de bosques, cada uno con una especie de árbol dominante.

En el modelo funcional, se pueden modelar los bosques del parque nacional como una función cuyo dominio es el conjunto de puntos en el mapa del parque nacional y el rango es un conjunto de especies de árboles. En este caso, se trata de una funcion escalonada (constante dentro en un mismo bosque y con saltos en los bordes entre bosques).

En general, para modelar un problema usando el paradigma funcional, se deben definir tres componentes [12]:

- Un sistema de referencia espacial: una grilla finita sobre el espacio subyacente, similar al concepto que presentamos en la sección 2.2.1 sobre el Álgebra de RoSE. El ejemplo mas conocido es un sistema de referencia de la tierra por latitud y longitud.

- Campos simples: un conjunto finito de funciones computables $\{f_i : SF \rightarrow A_i, 1 \leq i \leq n\}$ donde SF es el sistema de referencia espacial y los A_i son dominios de atributos. La elecciones de estos dominios de atributos y las funciones que asignan elementos de los mismos a cada punto del espacio dependerá de la aplicación en cuestión.

- Operaciones de campos: especifican las relaciones e interacciones entre los distintos campos. Estas se dividen en:

    - Operaciones locales: operaciones en las que el valor del campo resultante solo depende de los valores de los campos de entrada en ese punto. La composición de campos es un ejemplo de operación local.
    - Operaciones focales: operaciones en las que el valor del campo resultante depende de los valores de los campos en un pequeña vecindad alrededor del punto. Este es el caso por ejemplo de operaciones que involucran gradientes.
    - Operaciones zonales: operaciones que involucran operadores agregados como el promedio o la integración. También son operaciones zonales aquellas que particionan el espacio en zonas.

Los modelos funcionales son buenos para representar problemas en los que las datos continuos como elevación, temperatura y variación del suelo. Tienen la ventaja de poder representar fenómenos espaciales amorfos o con contornos fluidos.

2.2.4. Modelos Basados en Objetos

Volviendo al ejemplo del parque nacional, si consideramos los lugares donde los bosques cambian, en un entorno idealizado donde los bordes entre bosques están claramente definidos, obtendremos los bordes de polígonos. A cada polígono se le puede asignar un identificador y un atributo no espacial (la especie de árbol dominante). Los bosques del parque pueden modelarse entonces como un conjunto de polígonos.

En el modelado basado en objetos, se abstrae información espacial en conjuntos de entidades únicas, distinguibles y relevantes llamadas objetos. Cada objeto tiene un conjunto de atributos que lo caracterizan, los cuales se dividen en atributos espaciales y no espaciales. En el ejemplo del parque nacional, un objeto bosque tiene un atributo espacial, un polígono, que representa su extensión espacial, y un atributo no espacial, su espacie dominante, representada como un valor alfanumérico. Es interesante notar que un objeto puede tener varios atributos espaciales. Por ejemplo, un camino puede tener un polígono y una línea, para elegir cuál usar según la escala del mapa.

Los modelos de objetos son los mas comunes en problemas relacionados a representar redes de transportes o parcelas de tierra. Tienen la ventaja de ser mas intuitivos y cercanos a los modelos Entidad-Relación ya muy presentes en los DBMS más usados. En definitiva, la decisión de usar el paradigma funcional o de en objetos se basará en los requerimientos de la aplicación.

En las próximas secciones, detallaremos más los SDTs y operaciones espaciales en el modelo basado en objetos.

2.2.5. SDTs

Se han propuesto muchos conjuntos básicos de tipos espaciales para la representación de formas comunes en mapas. Actualmente, la más adoptada es el estándar OGIS [6] y es en la que nos basaremos. El estándar OGIS propone los siguientes tipos de datos:

- Geometría: es un tipo abstracto (no puede ser instanciado) del cual se derivan todos los otros tipos. Tiene asociado un sistema de referencia.

- Punto: describe la posicion de objetos de cero dimensiones, como una oficina o una zona de campamento, en el caso del parque nacional.

- Cadena de Lineas: describe un objeto de una dimension a través de una sucesión de de dos o mas puntos, que pueden representar un segmento o aproximar una curva. Por ejemplo, caminos o ríos.

- Polígono: objetos de 2 dimensiones, como un bosque.

- Colección de Geometrías: formas complejas formadas por conjuntos de otros tipos. Se dividen en multipuntos, multilineas y multipoligonos. Estos tipos son necesarios para asegurar que las operaciones geométricas sean cerradas. Por ejemplo, la intersección de un río con un bosque de forma cóncava pueden ser representado por un conjunto de líneas.

2.2.6. Operaciones Espaciales

En el modelo funcional, las relaciones disponibles quedaban definidas por las funciones de campos del modelo. Pero en un modelo basado en objetos, el espacio subyacente será el que determine las operaciones disponsibles y las relaciones que pueden existir entre objetos. La siguiente tipología de espacios resume la presentada por [9]:

- Espacios Orientados a Conjuntos: son los espacios mas simples y generales. Permiten todas las relaciones usuales de conjuntos como unión, intersección, contención y pertenencia.
- Espacios Topológicos: los espacios topológicos son aquellos en los que las relaciones no se ven afectadas por transformaciones elásticas del mismo.
- Espacios Direccionales: las reaciones direccionales de un espacio pueden ser absolutas (en referencia a un sistema de referencia global), relativas (basadas en la orientación de un objeto) o basadas en un observador (un objeto especial designado como tal).
- Espacios Métricos: son aquellos espacios en los que la noción de distancia está bien definida. La función de distancia puede usarse para definir una topología y por lo tanto todo espacio métrico es también un espacio topológico.
- Espacios Euclideanos: Son espacios en los que los puntos pueden expresarse como la combinación lineal de los elementos de un sistema de referencia. Sobre espacios euclideanos pueden definirse todas las operaciones de las categorías anteriores.

Teniendo en cuenta esto, veamos ahora algunos ejemplos de las operaciones que define el estándar OGIS. Consideremos una base de datos espacial con las siguientes tablas que definen países, ciudades y ríos.

```
CREATE TABLE Pais(
    nombre varchar(30),
    continente varcahr(30),
    pbi integer,
    geometria Polygon
);

CREATE TABLE Ciudad(
    nombre varchar(30),
```

```
    pais varchar(30),
    poblacion integer,
    geometria Point
);

CREATE TABLE Rio(
    nombre varchar(30),
    largo Number,
    geometria LineString
);
```

Podemos usar el predicado `Touch` para encontrar los países limítrofes de Argentina.

```
SELECT P1.nombre AS "Limítrofes de Argentina"
FROM Pais P1, Pais P2
WHERE Touch(P1.geometria, P2.geometria) = 1
      AND P2.nombre = 'Argentina';
```

Podemos usar el predicado `Cross` para encontrar todos los paises por los cuales pasa cada río.

```
SELECT R.nombre, P.nombre
FROM Rio R, Pais P
WHERE Cross(R.forma, P.forma) = 1
```

Por último veamos una consulta mas compleja que combina dos operaciones nuevas. Supongamos que el Río Paraná puede proveer de agua a todas las ciudades a menos de 300km de distancia. Primero, podemos usar `Buffer` para encontrar el área a 300km de dicho río y luego, `Overlap` para encontrar aquellas ciudades que tienen intersección con esta área.

```
SELECT C.nombre
FROM Ciudad C, Rio R
WHERE Rio.nombre = "Paraná"
      AND Overlap(C.forma Buffer(R.forma, 300)) = 1
```

Para un listado completo de los operadores y predicados espaciales presentes en el estandar SQL, sugerimos referirse a [6].

Capítulo 3

Bases de Datos Temporales

3.1. Introducción

Estudiaremos ahora el problema de representar **datos temporales** y **relaciones temporales** entre datos, ambos aspectos muy importantes de los fenómenos del mundo real. La habilidad de modelar esta dimensión temporal es esencial para aplicaciones informáticas como econometría, bancos, control de inventario, contabilidad, leyes, registros médicos, sistemas de reservas de vuelos y muchas otras.

Podemos introducir la idea de bases de datos temporales como una generalización a la siguiente intuición. Una base de datos tradicional representa el estado del sistema en un momento específico: el presente. La existencia de datos que existieron en el pasado o relaciones que fueron verdaderas previamente pero ya no lo son, quedan relegadas a la lógica de la aplicación del usuario y no poseen soporte del motor de base de datos.

Además, existe otro problema con notables similitudes. A medida que los datos se modifican, las versiones anteriores se pierden. Los datos desactualizados son borrados de la base de datos.

Surge entonces la problemática de las aplicaciones para las cuales nos sería útil almacenar y razonar sobre estos estados que si bien no son una representación del estado actual de la realidad modelada, representan igualmente información valiosa.

A lo largo de este capítulo exploraremos estas cuestiones. En la sección 3.2, introduciremos algunos conceptos lógica temporal. Luego, en la sección 3.3 exploraremos en mas detalle los dos tiempos que acabamos de introducir y que llamaremos tiempo de validez y tiempo de transacción. Por último, en la sección 3.4 veremos como estos conceptos fueron introducidos en el estándar SQL 2011.

3.2. El dominio del tiempo

Para poder plantear una implementación de una base de datos temporal es necesario comprender primero cómo se comporta y cómo se observa el tiempo.

En el capítulo anterior vimos como la investigación de bases de datos espaciales se vio motivada por la necesidad práctica de mejorar el manejo de datos de los sistemas

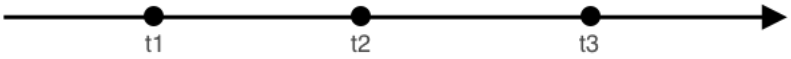


Figura 3.1: Ejemplo de modelo lineal del tiempo

GIS. Las bases de datos temporales por su parte, fueron fuertemente impactadas por la investigación previa en lógica temporal desde de los años 70[11] y que presentaremos a continuación.

3.2.1. La estructura del tiempo

Los trabajos fundacionales de la lógica temporal han dividido el estudio del tiempo en dos modelos estructurales principales: **lineal** y **ramificado**[10]. Comenzaremos entonces por estudiar estos modelos y luego mencionaremos brevemente otras posibilidades.

Modelo Lineal

Es el modelo más simple, donde el tiempo avanza del pasado al futuro de forma ordenada formando así una secuencia tal y como se ve en la figura 3.1.

Generalmente esta estructura es útil para mantener información histórica centrada en eventos del pasado. Por ejemplo: registros de transacciones de un banco, sistemas de control de inventario, etc.

Modelo ramificado

A diferencia del modelo anterior, el tiempo se presenta en dos formas: se extiende de forma lineal desde el pasado hacia el presente formando una secuencia y se ramifica del presente al futuro creando un árbol con raíz en el *ahora* cuyas ramas se extienden hacia el futuro. Podemos visualizar este modelo en la figura 3.2.

A esta forma de organizar el tiempo también se la conoce con el nombre de "modelo de los posibles futuros". Debido a sus características es útil para representar y trabajar con información predicativa.

Elección entre estructuras temporales

Los modelos más generales del tiempo en la lógica temporal representan el tiempo como un conjunto arbitrario que cumple con la restricción de ser un *orden parcial*. Otros axiomas introducen otros modelos más refinados del tiempo. Por ejemplo: el tiempo lineal puede ser simplificado agregando un axioma que imponga un *orden total* en este conjunto. Un modelo recurrente, por otra parte, puede trabajarse con un *modelo cíclico* del tiempo.

A lo largo de este trabajo se utilizará un modelo lineal del tiempo ya que es el más sencillo y brinda todas las características necesarias en las bases de datos temporales.

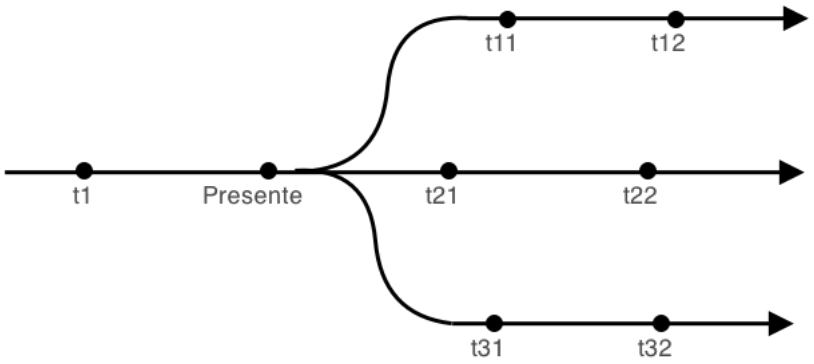


Figura 3.2: Ejemplo de modelo ramificado del tiempo

3.2.2. La densidad del tiempo

Siguiendo con el modelo lineal, la densidad del tiempo puede variar en la línea. Esta se puede clasificar en dos grupos: **discretos** y **densos**.

Modelos Discretos

Estas representaciones son isomorfas a los números naturales, lo cual implica que cada punto en el tiempo tiene un único sucesor. Cada número natural corresponde a una unidad no descomponible de tiempo de duración arbitraria. Estas reciben el nombre de **chronons**.

Modelos Densos

Por su parte, los modelos densos presentan dos variaciones: pueden ser isomorfos tanto a los números racionales como a los reales. Como consecuencia, dados dos momentos cualquiera en el tiempo, existe otro momento entre ellos.

Los modelos isomorfos a los reales también son conocidos como continuos.

Chronons y la elección del modelo de densidad

Un **chronon** es la duración de tiempo más pequeña que puede ser representada. Cabe destacar que no se trata de un punto sino un segmento en la línea del tiempo. A pesar de que el tiempo en sí mismo suele ser percibido como algo continuo, la mayoría de las propuestas para agregar una dimensión temporal a los modelos de datos relacionales están basados en un modelo discreto del tiempo. Existen varias razones para esta elección entre las que se destacan cuatro:

- Las medidas del tiempo son inherentemente imprecisas. Los instrumentos de medición temporal invariablemente reportan la ocurrencia de eventos en términos de

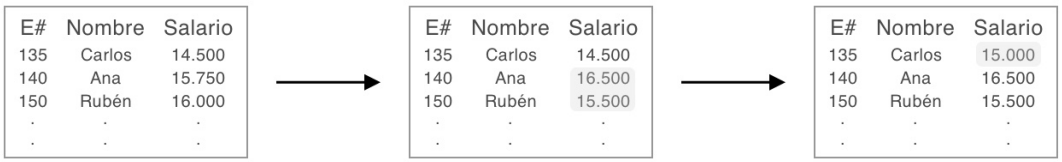


Figura 3.3: Base de Datos Snapshot como una maquina de estos finita.

chronons, no en "puntos" de tiempo. Por lo tanto, incluso los eventos así llamados "instantáneos" pueden ser medidos como si hubieran ocurrido durante un chronon, en el mejor de los casos.

- La mayor parte de las referencias temporales del lenguaje natural son compatibles con el modelo discreto del tiempo. Por ejemplo, cuando se dice que un evento ocurrió a las 4:30 PM, en realidad no se hace referencia a que el evento ocurrió en el "punto" del tiempo asociado a las 4:30 PM, sino a algún momento cercano a las 4:30 PM aunque exista una variación de uno o dos minutos.
- Los conceptos de chronon y de intervalo permiten modelar naturalmente eventos que no son instantáneos sino que tienen duración.
- Cualquier implementación de un modelo de datos con dimensión temporal tendrá la necesidad de codificar el tiempo de forma discreta, Es decir que, es inevitable discretizar el tiempo en algún punto. Este problema es análogo al que analizamos en el Álgebra de Rose en el capítulo anterior.

3.3. *Tiempo de Validez* y *Tiempo de Transacción*

Ahora que hemos introducido los conceptos de la lógica temporal, pasaremos a la cuestión de integrar estos conceptos a un modelo de bases de datos relacionales. Como veremos, es útil definir dos categorías fundamentales de datos temporales: **tiempo de validez** y **tiempo de transacción**.

Volviendo sobre la intuición que planteamos en la introducción de este capitulo, llamaremos **bases de datos snapshot** a cualquier base de datos que no soporta capacidades temporales (independientemente de si tiene datos temporales en un sentido definido por el usuario). Como se aprecia en la figura 3.3, podemos modelar estas bases de datos como maquinas de estado finitas, en la que cada estado contiene solo los datos del presente y un cambio destruye los datos anteriores.

Si nos interesa modelar los aspectos temporales propios del dominio del problema, podemos ampliar este modelo introduciendo el concepto de tiempo de validez, es decir el tiempo cuando el dato es verdadero en la realidad modelada. Un dato puede tener asociados múltiples instantes e intervalos de tiempo de validez que representen diferentes atributos temporales del mismo. A este tiempo también se lo suele conocer como tiempo del mundo real o tiempo lógico.

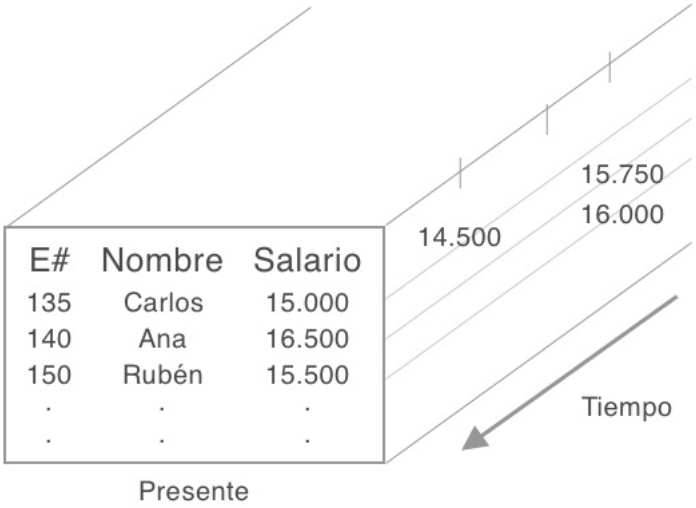


Figura 3.4: Base de datos con tiempo de transacción.

Por otro lado, si nos interesa mantener un registro histórico de los estados que recorrió la base de datos, introduciremos el tiempo de transacción, es decir el tiempo en el cual el hecho fue modelado por el usuario. Llamaremos **consultas de viaje en el tiempo** a aquellas que especifiquen un tiempo de transacción distinto al presente.

Es importante notar que estos dos tiempos no solo cumplen roles distintos y son independientes el uno del uno del otro, sino que ademas se comportaran distinto. Generalmente, el tiempo de validez será ingresado por el usuario mientras que el tiempo de transacción es administrado por el motor de bases de datos. Como consecuencia de esto, un usuario no puede modificar los estados del pasado. Llamaremos además **base de datos bitemporal** a una base de datos que soporta ambos tiempos y **base de datos unitemporal** a aquellas que solo soportan uno de ellos.

3.4. Funcionalidades Temporales en SQL 2011

La versión 2011 de SQL[4], publicada en diciembre de 2011 extiende el estándar con funcionalidades temporales. Esta extensión se basa en el agregado de **períodos temporales** a las tablas. La misma define dos variantes de períodos: de **aplicación de usuario** (para tiempos de validez) y de **versionado del sistema** (para tiempos de transacción). Los períodos se definen con dos columnas temporales (tipos *Date* o *Timestamp*) de la tabla. Se prefirió esta forma en lugar de un tipo nuevo ya que es la que permite migrar más fácilmente sistemas previos al estándar. Además, como es común en el álgebra temporal, los periodos son **cerrado-abierto**, es decir que contienen el momento de su inicio pero no de su finalización.

3.4.1. Períodos de Aplicación de Usuario

Veamos ahora un ejemplo de una tabla que define un periodo de aplicación de usuario:

```
CREATE TABLE Estudiantes (
    Legajo INTEGEER,
    EStart DATE,
    EEnd DATE,
    ECarrera INTEGER,
    PERIOD FOR EPeriod (EStart, EEnd)
);
```

Como vemos, las columnas `EStart` y `EEnd` son respectivamente el comienzo y fin del período `EPeriod`. Como mencionamos en la sección anteior, al tratarse de un tiempo de validez debe poder ser establecido por el usuario. Efectivamente, podemos especificar los valores de este período y de hecho, la sintaxis para hacerlo no cambia en nada respecto a un `INSERT` tradicional.

```
INSERT INTO Estudiantes
VALUES (1234, DATE '2013-03-01', DATE '2018-12-20', 1);
```

Consideremos ahora la tabla con esta fila y veamos una query un poco mas interesante.

```
UPDATE Estudiantes
    FOR PORTION OF EPeriod
        FROM DATE  DATE '2015-04-01'
        TO DATE '2015-05-01'
SET Carrera = 2
WHERE Legajo = 1234;
```

Lo primero que notamos en este `UPDATE` es que introducimos una sintaxis nueva que nos permite establecer en que tiempo de validez debe ser actualizado el valor. En este caso, el período especificado está contenido en el período de la fila presente en la tabla.

Pero entonces, ¿Cuál debería ser el nuevo período de validez de la fila actualizada? La respuesta es que en realidad, un `UPDATE` temporal se comporta un poco distinto uno tradicional. O, mas precizamente, es una generalización de este. Si consideramos que un `UPDATE` clásico inserta una fila en la tabla por cada una que remueve, un `UPDATE` temporal puede crear varias filas (hasta tres) por cada una que borra. En este caso, por ejemplo, el resultado seran tres filas, como vemos en la tabla 3.1. De todas formas, si la tabla tuviera definidos triggers para las actualizaciones, la fila con `carrera = 2` se considera como la actualizada.

Similarmente, un `DELETE` puede borrar un dato en un subperíodo de validez.

Legajo	EStart	EEnd	ECarrera
1234	2013-03-01	2015-04-01	1
1234	2015-04-01	2015-05-01	2
1234	2015-05-01	2018-12-20	1

Cuadro 3.1: la tabla `Estudiantes` luego del `UPDATE`.

3.4.2. Claves Primarias

Esta nueva semántica de actualizaciones que y borrados nos introduce por otro lado un problema. En el ejemplo anterior, si la clave primaria de la tabla era el legajo, se produce una duplicación. Es evidente que tenemos que incluir el período en la PK.

Comunmente querremos además evitar una forma particular de duplicación en los datos temporales: filas iguales en la parte no temporal de la clave primaria y con una superposicion de periodos. Para evitar este caso claramente no alcanza con prohibir la igualdad en todas las columnas como en un dato no temporal. Por este motivo, se introduce la directiva `WITHOUT OVERLAPS`. Podemos volver a definir entonces nuestra tabla `Estudiantes` de la siguiente manera:

```
CREATE TABLE Estudiantes (
    Legajo INTEGEER,
    EStart DATE,
    EEnd DATE,
    ECarrera INTEGER,
    PERIOD FOR EPeriod (EStart, EEnd),
    PRIMARY KEY (
        Legajo,
        EPeriod WITHOUT OVERLAPS
    )
);
```

3.4.3. Períodos de Versionado del Sistema

Como mencionamos previamente, el otro tipo de períodos es de versionado de sistema. Las tablas declaradas con versionado de sistema tienen campos temporales de nombre fijo (`sys_start` y `sys_end`), que el usuario no puede modificar, sino que sus valores son definidos por el sistema al momento de realizar otras operaciones. El uso de esta funcionalidad es opcional y se declara con la sintaxis WITH SYSTEM VERSIONING.

```
CREATE TABLE Estudiantes (
    Legajo INTEGEER,
    EStart DATE,
    EEnd DATE,
    ECarrera INTEGER,
    PERIOD FOR EPeriod (EStart, EEnd),
```

```
    PRIMARY KEY (
        Legajo,
        EPeriod WITHOUT OVERLAPS
    )
)
WITH SYSTEM VERSIONING;
```

Como mencionamos en la sección anterior, a las consultas por datos en un tiempo del sistema diferente al presente se las conoce como **consultas de viaje en el tiempo**. Los siguintes son algunos ejemplos de este tipo de consultas.

```
SELECT Legajo, Carrera, sys_start, sys\_end
FROM Estudiante
FOR SYSTEM TIME AS OF TIMESTAMP '2017-05-01 00:00:00';

SELECT Legajo, Carrera, sys_start, sys_end
FROM Estudiante
FOR SYSTEM TIME
    FROM TIMESTAMP '2017-05-01 00:00:00'
    TO TIMESTAMP '2017-06-01 00:00:00';

SELECT Legajo, Carrera, sys_start, sys_end
FROM Estudiante
FOR SYSTEM TIME BETWEEN
    TIMESTAMP '2017-05-01 00:00:00'
    AND TIMESTAMP '2017-06-01 00:00:00';
```

Vale la pena mencionar brevemente algunos detalles de esta sintaxis. Por un lado, tenemos dos formas de especificar períodos ya que `FROM ... TO ...` es Cerrado-Abierto mientras que `BETWEEN ... AND ...` es Cerrado-Cerrado. Por otro, se definen algunas constantes: una query sin especificar tiempo es equivalente a `FOR SYSTEM TIME AS OF CURRENT TIMESTAMP`.

Cuando está activada esta funcionalidad, cualquier query `UPDATE` o DELETE automáticamente preserva el estado anterior en filas nuevas. Podemos pensar que las filas son de dos tipos, de **sistema actual** cuando el presente se encuentra dentro de su período de versionado de sistema e **históricas** en otro caso. Obviamente, no pueden realizarse queries `UPDATE` o `DELETE` sobre filas históricas.

Las restricciones de integridad, por otro lado, son mas sencillas que en el caso de los períodos de aplicación de usuario: cualquier restriccion que es chequeada y se cumple en el sistema actual, heredará la propiedad cuando pase a ser una versión histórica.

El estándar SQL 2011 soporta además bases de datos bitemporales en el sentido definido en al sección 3.3, permitiendo combinar libremente estas funcionalidades con las de tiempo de aplicación de usuario vistas anteriormente.

Capítulo 4

Bases de Datos Espacio-temporales

4.1. Introducción

En este capitulo aplicaremos un enfoque distintos a los capitulos anteriores. Para el estudio de las bases de espaciales y temporales fue necesario presentar varios conceptos teóricos, la mayoría provenientes de otras ramas científicas, que construyen los cimientos para la implementación de las mismas. Luego, presentamos los estándares ya afirmados en cada área, justificando con estos marcos teóricos las decisiones de diseño que se tomo en cada uno.

En el campo de las **bases de datos espacio-temporales**, que es como llamaremos a las bases de datos que combinan de alguna forma ambas funcionalidades, veremos que el panorama es un poco distinto. Debido a la gran amplitud en requerimientos de las distintas aplicaciones, existen muchos y variados modelos, cada uno caracterizado por qué elementos toma de las funcionalidades espaciales, cuáles de las temporales y como los combina. La mayoría de estos modelos se focalizan en una o pocas aplicaciones. Si bien existen modelos mas generales, son muy complejos conceptualmente y no muy fáciles de optimizar para problemas específicos.

Por lo tanto, a lo largo de este capítulo iremos presentando brevemente, en orden incremental de complejidad, diferentes modelos y sus aplicaciones, intercalándolos con conceptos y definiciones cuando sea necesario. Para un estudio comparativo mas en profundidad de los distintos modelos, sugerimos recurrir a [8].

4.2. Modelo de Snapshots

Comenzaremos por el **modelo de snapshots**[5], por ser uno de los mas sencillos conceptualmente. Este modelo divide los datos en capas temporalmente homogéneas, con un timestamp único. De esta manera, muestra el estado de la distribución espacial en diferentes tiempos sin relaciones explícitas entre capas.

Es un modelo que si bien representa aspectos espaciales y temporales, ambos se encuentran muy separados y no agrega comportamientos complejos. Desde el punto de vista espacial, podemos pensar el tiempo como un atributo no espacial (en el sentido definido en la sección 2.2.4). Desde el punto de vista temporal, a las consultas espaciales podemos agregarle condiciones de tiempo de validez similares a las que vimos en la sección 3.4.1.

Si bien como una primera aproximación es sencilla e intuitiva, la división en capas que propone este modelo tiene varias desventajas.

- El modelo no es apropiado para describir cambios en el espacio a través del tiempo. Es difícil determinar cambios entre dos momentos ya que hay que comparar los snapshots.
- Todo cambio produce una copia completa en cada porción de tiempo, generando una importante duplicación de datos. Este problema se ve incrementado por la tendencia de los datos espaciales de por sí a ser muy pesados.
- Es muy difícil diseñar o aplicar reglas para la lógica interna o la integridad. El modelo no proporciona una comprensión de las restricciones en la estructura temporal.

4.3. MADS

El modelo de snapshots y muchos otros modelos posteriores desarrollaron la idea de representar datos espaciales que pueden cambiar. Pero un enfoque diferente, propuesto por primera vez por [1], propone *representar los procesos de cambio que actuan sobre los atributos geométricos de una entidad.*

Un importante exponente de este modelo es **MADS**[7] (sigla en inglés de "Modeling Application Data with Spatio-temporal features"). MADS se propone incorporar los conceptos básicos de modelado de espacio y tiempo en el modelo objeto-relación. Los procesos son representados como relaciones entre objetos espacio-temporales.

```
OBJECT Parcel
    TEMPORAL Day
    GEOMETRY AREA TEMPORAL DAY
    ATTRIBUTES
        Number: INTEGER
        Name: string [1:N] TEMPORAL DAY
        Owner: [1:N] TEMPORAL DAY
END Parcel
```

4.4. MOST

El último paradigma que veremos es el de los modelos que se centran en representar objetos en movimiento.

Capítulo 5

Conclusiones

Primero ponemos que espacio-temporales tiene problemas propios que no son la simple conbinacion de resolver al mismo tiempo problemas espaciales y temporales, sino que emergen propiedades y problemas nuevos.

Terminamos con la conclusion del final de las slides. Que aun no hay consenso cientifico de si vamos a seguir ramificando o se va a encontrar un modelo general para todos los usos.

Bibliografía

[1] C. Claramunt, C. Parent, S. Spaccapietra y M. Thériault. «Database Modelling for Environmental and Land Use Changes». En: *Geographical Information and Planning: European Perspectives*. Ed. por J. Stillwell, S. Geertman y S. Openshaw. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, págs. 181-202. ISBN: 978-3-662-03954-0. DOI: 10.1007/978-3-662-03954-0_10. URL: https://doi.org/10.1007/978-3-662-03954-0_10.

[2] D. H. Greene y F. F. Yao. «Finite-resolution computational geometry». En: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. 1986, págs. 143-152.

[3] R. Gting, M. Schneider, P. Iv y F. Hagen. «Realm-Based Spatial Data Types: The ROSE Algebra». En: (dic. de 1995).

[4] K. Kulkarni y J.-E. Michels. «Temporal features in SQL:2011». En: *ACM SIGMOD Record* 41 (oct. de 2012), págs. 34-43. DOI: 10.1145/2380776.2380786.

[5] G. Langran y N. Chrisman. «A Framework For Temporal Geographic Information». En: *Cartographica: The International Journal for Geographic Information and Geovisualization* 25 (ene. de 1992). DOI: 10.3138/K877-7273-2238-5Q6V.

[6] OGC. *Open GIS Consortium, Inc. OpenGIS ® Simple Features Specification For SQL Revision 1.1*. 1999.

[7] C. Parent, S. Spaccapietra y E. Zimanyi. «Spatio-Temporal Conceptual Models: Data Structures + Space + Time». En: *Proceedings of the 7th ACM Symposium on Advances in GIS* (1999). URL: http://infoscience.epfl.ch/record/99112.

[8] N. Pelekis, B. Theodoulidis, I. Kopanakis e Y. Theodoridis. «Literature review of spatio-temporal database models». En: *The Knowledge Engineering Review journal, Volume 19, Issue 3, Pages: 235-274* 19 (jun. de 2005), págs. 235-274. DOI: 10.1017/S026988890400013X.

[9] S. Shekhar y S. Chawla. *Spatial databases - a tour*. Ene. de 2003. ISBN: 978-0-13-017480-2. DOI: 10.1016/B0-12-369398-5/00337-6.

[10] R. Snodgrass. «Temporal Databases». En: 19 (oct. de 1986), págs. 35-42. DOI: 10.1109/MC.1986.1663327.

[11] R. Snodgrass. «Temporal Databases - Status and Research Directions.» En: *SIGMOD Record* 19 (dic. de 1990), págs. 83-89. DOI: 10.1145/122058.122068.

[12] M. Worboys y M. Duckham. «GIS, a Computing Perspective». En: (ene. de 2004).