University of Castilla-La Mancha

A publication of the
Department of Computer Science

Bases de Datos: Persistencia del Espacio-Tiempo

by

Elena Navarro, Jesús Damián García-Consuegra, Nikos Lorentzos

Technical Report **#DIAB-01-06-22** October 2001

DEPARTAMENTO DE INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete – 02071 – Spain
Phone +34.967.599200, Fax +34.967.599224

Este trabajo ha sido parcialmente financiado por el proyecto CICYT “METODOLOGIA DE ANALISIS DE SISTEMAS DE INFORMACION GEOGRAFICA”, TIC 2000-1106-C02-02.

NOTA PARA EL LECTOR:

A lo largo del presente trabajo y con objeto de facilitar su lectura, se han definido diferentes estilos para denotar:

- Conceptos introducidos: *bases de datos temporales*
- Datos usados en los ejemplos: desde el 1 de Enero de 1999 hasta hoy, etc.
- Operadores más relevantes: *UNFOLD* , *FOLD*, etc.
- Definición de sentencias:
  `SELECT P.Propietario, P.Parcela, FROM Propietarios P`
- Otras definiciones:
  $\forall$ `geo in GEO.` $\forall$ ext_1, ext_2 `in EXT`
  `geo x regions` $\rightarrow$ `bool inside`

ÍNDICE

1 Introducción

El papel tradicional de los sistemas gestores de base de datos ha sido el de almacenar información de forma segura, en cuanto a controles de integridad, control de usuario, etc. y recuperarla eficientemente, con unos mecanismos de indexación apropiados. Pero esta información normalmente se limita a números, cadenas de texto, etc, es decir, información cuya complejidad es limitada. La necesidad de introducir una referencia espacial asociada a esta información alfanumérica ha puesto en evidencia, con el paso de los años, su carencia para el tratamiento de este tipo de información. La respuesta ha sido lo que se ha dado en llamar bases de datos espaciales (SDBMS). Éstas han sido un área de investigación activa durante las últimas décadas, principalmente por las necesidades de aplicaciones como los Sistemas de Información Geográfica y los sistemas CAD, así como aplicaciones potenciales tales como Sistema de Información Multimedia, Data Warehouse, etc.

Pero el ritmo vertiginoso con el que cualquier tipo información cambia obliga, en numerosas ocasiones, a descartar información. Aunque su período de validez no coincide con el actual, su utilización podría contribuir bien en el procesamiento de información más reciente, o bien en el conocimiento de su evolución temporal o como referencia histórica. Con objeto de eliminar, esta problemática, se han desarrollado durante los últimos tiempos las llamadas bases de datos temporales, que ofrecen una solución formal a este tipo de problemas.

La integración de las diferentes aproximaciones que se han venido desarrollando durante las últimas décadas, tanto de las bases de datos espaciales como temporales han supuesto los pilares en la definición de las *bases de datos espaciotemporales*. La convivencia tanto del dominio espacial como temporal supone, sin lugar a dudas, un catalizador para cualquier sistema que se apoye en ellos, ya sea un Sistema de Información Geográfica, un CAD o cualquier otro.

La integración se concreta en:

- la definición de modelos de datos que permitan una definición lógica abstracta y autocontenida de los objetos que intervienen en ambos dominios,
- la introducción de lenguajes, que permitan su consulta y definición, y
- estructuras de datos que permitan su recuperación eficientemente tanto en el tiempo empleado como en el espacio ocupado por las mismas.

El objetivo del presente trabajo es presentar el estado del arte de las diferentes líneas, que se ven afectadas con la incorporación del dominio espacial, temporal y espaciotemporal, en el campo de las bases de datos, es decir, en los modelos de datos, los lenguajes de consulta y los algoritmos de indexación. Para ello, se presentará en primer lugar se introducen las bases de datos temporales, haciendo especial hincapié en una serie de conceptos que afectan directamente a la gestión del tiempo en las bases de datos espaciotemporales. A continuación, se

introducen algunos de los modelos de datos, lenguajes de consulta y algoritmos de indexación propuestos en el dominio espacial. El trabajo termina presentando las tendencias más difundidas en estos tres campos en el área de las bases de espaciotemporales, intentando dar aquellas referencias básicas que puedan ayudar en estudios posteriores de mayor profundidad.

2 Bases de Datos Temporales

Tradicionalmente, las bases de datos tratan de representar una visión de la información referente al mundo real. Dicha información tiene una validez determinada en un instante determinado. Por lo que si esta realidad se ve alterada en algún momento. El sistema gestor (DBMS) ha de reflejar este cambio mediante actualizaciones o borrados en la base de datos, según sea la naturaleza del cambio. Esta manera de tratar las modificaciones conlleva que el estado previo de la información se pierda o se desprecie. Este tipo de bases de datos se conoce como base de datos *snapshot*, debido a que ofrecen una fotografía de la realidad en un momento concreto**.**

La motivación de gestionar así los cambios en la información proviene de la capacidad, siempre limitada, de almacenamiento o de carencias en el modelo de datos que permitan capturar esa evolución de la información. La aparición de mejoras en la tecnología ha permitido utilizar mayores y más rápidos sistemas para almacenamiento de la información, erradicando de las limitaciones del sistema entre las restricciones que soportan tal decisión.

A estas mejoras en la tecnología se une la aparición de las *bases de datos temporales*, que se caracterizan, esencialmente, por incorporar la representación de aspectos temporales en el modelo de datos, con una semántica especial (que veremos en el siguiente apartado) junto con las facilidades en el sistema gestor necesarias para su manipulación.

Las bases de datos temporales han sido un área de investigación activa durante las últimas décadas, como podemos apreciar por la numerosa bibliografía que trata sobre ellas, tanto en temas relativos a los modelos de datos[45, 24], lenguajes[6, 20], indexación [4, 51] o implementación [23, 59, 62]. En el marco del modelo relacional, la aparición de TSQL2 [58], es un reflejo del trabajo que se ha estado realizando por aunar esfuerzos y que trata de integrar las diferentes alternativas propuestas. Éste consiste en una extensión consistente del lenguaje estándar SQL-92, que permite introducir esa dimensión temporal, desarrollado con la intención de ser incluido en el estándar SQL3.

Los tipos de datos temporales, sus operaciones asociadas y las diferentes aproximaciones que existen para llevar a cabo su integración en el sistema gestor, han supuesto la aparición de diferentes conceptos que han de ser estudiados [35].

2.1 Algunos conceptos básicos

Según Date [8], una base de datos temporal es aquella que contiene datos históricos en lugar de, o así como, datos actuales, es decir, situaciones en las que

es necesario reflejar la validez temporal de la información, durante qué intervalo o instante de tiempo se consideró o fue considerada como válida. Así, un ejemplo podría darse al intentar reflejar la información temporal referente a la propiedad de una parcela L. Podemos encontrar diferentes alternativas para registrar esa evolución:

1. La parcela L fue adquirida por O el 1 de Enero de 1999.
2. La parcela L ha pertenecido a O desde el 1 de Enero de 1999 hasta la fecha.

Cada una de las sentencias anteriores proporciona una interpretación diferente sobre el estado o la situación de propiedad en que se encuentra la parcela L. En la primera, nos informa acerca del momento concreto en que la parcela L pasó a ser propiedad de O; no proporciona información sobre si la parcela pertenece o no en la actualidad a O, solo indica un instante temporal. En la segunda, además, se representa el período o intervalo de tiempo durante el cual la parcela L ha pertenecido a O. En ambas, los datos tienen una referencia temporal, introducida mediante la fecha "*1 de Enero de 1999*", que recibe, en la literatura inglesa, el nombre de *timestamp*, que marca en ambos casos un instante temporal. El timestamp nos permite incluir una marca temporal sobre un momento de validez de la información. Podemos apreciar igualmente, que la información referente a la parcela, como por ejemplo su propietario, extensión, catalogación, etc, puede estar asociada a cualquier número de eventos o de intervalos a lo largo del tiempo.

Basándonos en el ejemplo anterior, podemos introducir dos conceptos claves en el campo de las bases de datos temporales: tiempo de validez(valid time) y tiempo de transacción(transaction time). El primero de ellos expresa el tiempo durante el cual una cierta proposición es cierta, es decir, según el caso anterior deberíamos almacenar que la validez de la información relativa a la propiedad de la parcela es: "desde el 1 de Enero de 1999 hasta hoy". Mientras que el tiempo de transacción indica el tiempo en que una proposición aparece reflejada en la base de datos como cierta, el momento en que incorporamos esa información en la base de datos. El primero de ellos puede ser actualizado por el usuario para reflejar un cambio en la proposición, en cambio, el segundo puede ser únicamente modificado por el sistema gestor, cualquier cambio que un usuario realice sobre la información quedaría reflejado mediante un nuevo tiempo de transacción que marcaría el momento en que hizo la modificación.

El tiempo de validez y de transacción no debe confundirse con el *tiempo de usuario*(*user time*). Éste representa el empleo, por parte del usuario de un tipo de datos estándar (date) de un modo similar al empleo de los enteros o los reales, para representar alguna información temporal. Por ejemplo, en una base de datos de personal, al introducir una información como la fecha de alta de un usuario o su fecha de nacimiento se está reflejando una información temporal, no su período o instante de validez. En este caso, no se ofrece un soporte especial del lenguaje de consulta como en el caso del tiempo de validez o de transacción.

La introducción del tiempo de validez y de transacción, ha permitido la definición de los operadores: *valid timeslice* y *transaction timeslice*. Ambos toman como argumento una relación, con una referencia temporal ya sea de validez o de transacción, dependiendo de si se trata del primer o del segundo operador, y un valor temporal que indica un momento o instante sobre el que queremos realizar la operación. De esta forma el operador valid timeslice devolverá el estado de la relación válida en ese instante, es decir, toda aquella información que contuviera esa relación valida en ese momento. El uso del operador transaction timeslice, permitirá obtener aquella información que era actual en ese momento.

De igual forma que el tiempo de transacción y de validez son aplicables sobre una información almacenada en la base de datos, también es posible aplicar estos tiempos a una relación. Es decir, permitir que pueda reflejarse cuando una relación ha sido incluida en el esquema o cuando se han efectuado modificaciones sobre ella en el primer caso, o delimitar, en el caso de tiempo de validez, en que instante o durante que intervalo de tiempo la relación ha sido válida. Marcando por tanto una diferencia sobre las *relaciones snapshot*, que podemos encontrar en los sistemas de gestión tradicionales, en las que una relación no viene cualificada por ningún atributo temporal.

Snodgrass [58] clasifica las bases de datos en *estáticas*, *históricas*, *rollback* o *bitemporales*, según como se contemplen, o no, los conceptos anteriores dentro del esquema. Las bases de datos históricas no contemplan ni el tiempo de validez ni de transacción, no almacenan ninguna de estas referencias temporales en el modelo que emplean. Las bases de datos históricas contemplan únicamente el tiempo de validez, almacenan la información que conocemos como válida, tanto presente como pasada o futura, pero los cambios producidos en la información (como ha sido la evolución de las actualizaciones que se han realizado) no quedan almacenados. Las bases de datos rollback, en cambio, contienen exclusivamente tiempo de transacción, nos permiten devolver la base de datos a un estado anterior en caso de que sea necesario, pero no permiten conocer durante que intervalo de tiempo ha permanecido como valida una determinada información. Por último, las bitemporales soportan tanto el tiempo de transacción como el de validez, por lo que nos dan una visión más precisa de la evolución que ha sufrido la información tanto sobre sus intervalos de validez como de sus actualizaciones, al contemplar los diferentes estados por los que pasó la información, y permitir su recuperación.

Otro concepto, que afecta especialmente a la semántica de las operaciones y que ha de ser incluido es la definición de Crono *o* quantum temporal. Éste hace referencia a la duración de tiempo más corta, al instante temporal más breve, soportado por el DBMS. Un quantum determina un subintervalo de tiempo, con una duración fija a lo largo de la línea temporal.

Junto a las definiciones anteriores es de especial importancia una de las características de los datos temporales: la *granularidad temporal*. La granularidad es una medida de tiempo que puede ser expresada en días, minutos, segundos, etc. Su importancia deriva en como influye en la semántica de las operaciones temporales. Tomando como ejemplo el caso anterior de la propiedad de una parcela, se puede apreciar en la Tabla 1 como se ha incluido esa referencia temporal, mediante los años de comienzo y de fin en que una persona fue propietario, pero, ¿qué ocurriría al tratar de realizar una consulta sobre esa relación para conocer "quién fue propietario de la parcela L1 el 2 de Enero de 1991"?

Tabla 1 Relación de propietarios

Propietario	Parcela	Desde	Hasta
O1	L1	1990	1998
O2	L1	1999	2000
O3	L2	1980	1982
O4	L2	1983	1984

```
SELECT P.Propietario, P.Parcela
FROM Propietarios P
WHERE '1-01-1991' OVERLAPS (P.Desde, P.Hasta)
```

En la setencia SELECT anterior, escrita en SQL92, estamos realizando una intersección del intervalo que marcan esas dos referencias temporales, Desde y Hasta, con la fecha "1-01-1991" mediante el operador de intersección temporal OVERLAPS. La granularidad que tienen es diferente, la primera de ellas es de años y la segunda de días. Si no contempláramos este hecho esa intersección daría como resultado un error, dado que el procesador de consultas consideraría que son tipos incomparables entre sí. Varias soluciones han sido propuestas en las que se facilitan diferentes semánticas para las operaciones de este tipo, como la presentada en [11]. En todas ellas se trata de realizar una transformación de uno de los operandos a la granularidad elegida como final. El problema radica en cuál de ellas es la más idónea: aquella que presenta una granularidad más fina, es decir, la de días en nuestro ejemplo, o más gruesa, como sería la de años. La necesidad de contemplar la granularidad temporal conlleva otros problemas fuera del alcance de este trabajo. Para más información consultar [11].

Para las anteriores definiciones, se hace necesario la descripción de un dominio temporal que permita especificar la escala de las anteriores primitivas. Dos alternativas se han presentado para el dominio temporal. La primera de ellas lo define como un dominio discreto, es decir, isomórfico al conjunto de los enteros de forma que para cualquier primitiva temporal existe un único sucesor y predecesor. La segunda alternativa, define el dominio temporal como un dominio continuo, isomórfico al conjunto de los números reales. Esta alternativa implica que entre cualesquiera dos primitivas temporales existe otra primitiva temporal.

La definición de un dominio temporal discreto supone ventajas como la mayor proximidad a la comprensión humana, la facilidad para modelar sucesos con una duración determinada y la mayor sencillez de implementación, pero presenta dificultades para modelar determinados sucesos en el campo de la física o la matemática.

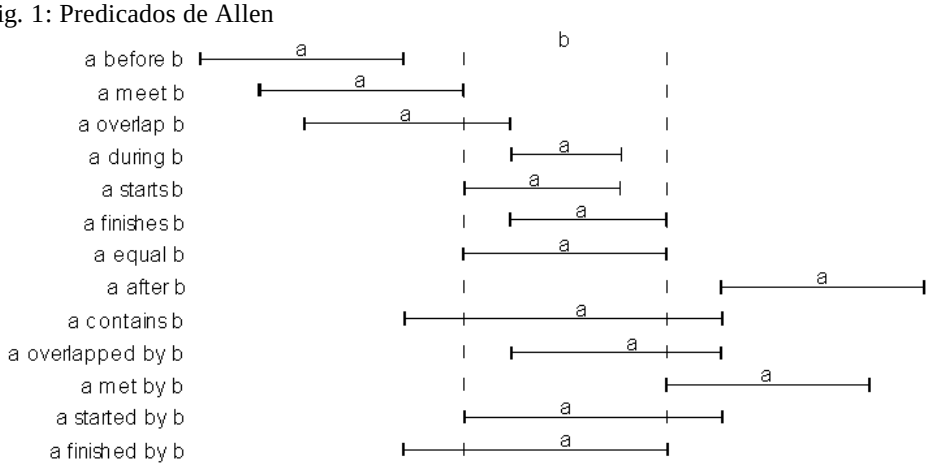
2.2 Un tipo de datos especial: Intervalos

Según podemos observar en la Tabla 1 para poder almacenar el período de tiempo de propiedad de una parcela se hace necesaria la incorporación de dos atributos (desde y hasta). Esta solución supone diferentes problemas para el usuario de la base de datos, como la exigencia de declarar, de forma separada, este par de atributos y por tanto realizar un control de la integridad de la información a la que éstos hacen referencia. Integridad que debe preservase evitando estados no válidos, como los que se presentarían, por ejemplo, si se consintiera una fecha de comienzo(en la Tabla 1 campo Desde) de la propiedad posterior a la fecha de finalización (en la Tabla 1 campo Hasta). A la necesidad de evitar este tipo de situaciones se une la ausencia de operaciones específicas para la gestión de este tipo de información.

La conjunción de estos factores motivó la aparición de los *intervalos* como un tipo de datos que ofrecen una solución a este tipo de situaciones. El empleo de este tipo de datos supone, entre otras ventajas, una formulación más sencilla de las consultas que en el caso de tener que tratar, por separado, ambos extremos del intervalo o imponer restricciones, como en el ejemplo de la Tabla 1, para tratar de evitar que dos parcelas puedan tener dos propietarios en el mismo momento. Por ello Lorentzos et al. [39] identifican como una de las propiedades que han de cumplir los modelos de validez temporal, la incorporación de los intervalos temporales como un tipo de datos primitivo, a fin de solucionar los problemas que se mencionaron anteriormente.

Su definición consiste de un punto de inicio y otro de fin, así como un tipo sobre el que se construye el intervalo, es decir, si se trata de un intervalo de enteros, fechas, etc. El tipo de datos sobre el que se construye el tipo intervalo ha de cumplir una relación de orden total, que permita establecer una *relación de precedencia* entre los elementos que lo componen. La exigencia de esta relación viene dada por la necesidad de determinar cada elemento del intervalo a partir exclusivamente del punto de inicio y de fin. En el caso de las bases de datos temporales, obviamente, el tipo sobre el que se construye el intervalo es el tiempo, que ha de ser considerado como totalmente ordenado.

Fig. 1: Predicados de Allen



Otra característica inherente a la definición de los intervalos es su consideración como abiertos o cerrados con respecto a los puntos que determinan su inicio y fin. Consideramos que un intervalo es abierto con respecto a su punto de inicio, si éste no está incluido dentro de él. En otro caso diremos que es cerrado con respecto a ese punto. En el ejemplo de la Tabla 2, podemos apreciar como los intervalos son cerrados con respecto a su punto de inicio, pero abiertos con respecto a su punto de fin.

Junto con la definición de los intervalos encontramos una serie de predicados que podemos ver en la Fig. 1, comúnmente aceptados, conocidos como los *Predicados de Allen* [1]. Estos definen hasta un total de 13 relaciones distintas posibles entre dos intervalos que permiten la manipulación de datos temporales y la definición de nuevas operaciones sobre ellos.

Además de los anteriores predicados, se han propuesto otros operadores entre los que cabe destacar *UNFOLD* y *FOLD* [40], aunque pueden ser encontrados en la literatura con otros nombres. Estos dos operadores son de particular importancia debido a que permiten una sencilla formalización del resto de las operaciones definidas en el álgebra relacional que ha permitido la definición de *IXSQL* como una extensión consistente de SQL2 para la gestión de intervalos de datos. El primero de ellos, al aplicar *UNFOLD* sobre una relación R con respecto a un atributo A, reemplaza cada tupla por un conjunto de tuplas que mantienen los mismos valores para todos los atributos de la tupla excepto para el atributo A, para el cual aparecerá cada uno de los elementos simples que pertenecen a ese intervalo. El segundo, realiza la operación contraria, es decir, reúne en una tupla todas aquellas tuplas que son idénticas para todos los atributos distintos de A y que representan valores contiguos o solapados para A. Basándose en ellas, tal y como se ha comentado, se pueden definir el resto de las operaciones del álgebra, como se muestra en el siguiente ejemplo.

Tabla 2 Relación Cultivos

Parcela	Cultivo	Tiempo
L1	Maíz	[1985, 1989)
L1	Trigo	[1989, 1997)
L2	Maíz	[1997, 2000)
L3	Maíz	[1988, 1992)

Tabla 3 Relación de Propietarios

Propietario	Parcela	Tiempo
Pedro	L1	[1980, 1995)
Juan	L1	[1995, 2000)
Luis	L2	[1972, 1990)
Mario	L2	[1990, 2000)
José	L3	[1998, 2000)

Dadas las relaciones de las Tabla 2 y Tabla 3, si se quisiera obtener en qué momento "qué propietarios han cultivado maíz", se podría abordar mediante la operación *INTERVSECT*[40], que devuelve el resultado de la intersección de dos intervalos. Esta operación ha sido definida a partir de fold y unfold. En este ejemplo , el proceso sería:

1. Aplicar unfold a la relación de *Cultivos* y de *Propietarios* sobre el atributo *Tiempo*, generando dos nuevas relaciones en las que se repetirían los diferentes valores del resto de atributos, para cada uno de los elementos de esos intervalos definidos sobre el atributo *Tiempo*.

Tabla 4 UNFOLD de Cultivos sobre el atributo Tiempo

Parcela	Cultivo	Tiempo
L1	Maíz	1985
L1	Maíz	1986
L1	Maíz	1987
L1	Maíz	1988
L1	Trigo	1989
...	...	...
L1	Trigo	1996
L2	Maíz	1997
L2	Maíz	1998
L2	Maíz	1999
L3	Maíz	1988
L3	Maíz	1989
L3	Maíz	1990
L3	Maíz	1991

Tabla 5 Unfold de Propietarios sobre el atributo Tiempo

Propietario	Parcela	Tiempo
Pedro	L1	1980
Pedro	L1	1981
...	...	...
Pedro	L1	1994
Juan	L1	1995
...	...	...
Juan	L1	1999
Luis	L2	1972
...	...	...
Luis	L2	1989
Mario	L2	1990
...	...	...
Mario	L2	1999
José	L3	1998
José	L3	1999

2. Se aplicaría la intersección de ambas relaciones, buscando la coincidencia entre los instantes (timestamp) en que se ha cultivado trigo en una parcela y aquellos en los que un propietario tenía esa parcela (ver Tabla 6).

Tabla 6 Join de ambas tablas sobre el atributo tiempo

Parcela	Cultivo	Tiempo	Propietario
L1	Maíz	1985	Pedro
L1	Maíz	1986	Pedro
L1	Maíz	1987	Pedro
L1	Maíz	1988	Pedro
L2	Maíz	1997	Mario
L2	Maíz	1998	Mario
L2	Maíz	1999	Mario

3. Sobre el resultado de esa intersección(Tabla 6) se emplearía la operación fold, que tomaría conjuntos de tuplas en las que todos los atributos, excepto el Tiempo, son iguales y las reuniría generando nuevos intervalos con respecto a ese atributo. La Tabla 7 muestra resultado de la siguiente consulta SQL:

```
SELECT P.Propietario, P.Parcela
FROM Propietarios P
WHERE (P.Parcela = C.Parcela) AND (P.Time intervsect
   (SELECT C.Time
   FROM Cultivos  C
   WHERE Cultivo = Maíz))
```

Tabla 7 Resultado de la consulta

Propietario	Parcela
Pedro	L1
Mario	L2

Esa consulta sería equivalente a realizar una operación unfold sobre el atributo tiempo de ambas tablas. De esta forma sería posible calcular la intersección entre los instantes en que un propietario poseía una parcela y aquellos en que se cultivaba maíz. En el siguiente ejemplo vemos como se realizaría la operación en el álgebra relacional extendida:

$\pi_{propietario,\ parcela}(\sigma_{cultivo='Maiz'}$
(FOLD Tiempo,((UNFOLD PROPIEDAD, TIEMPO) INTERSECT
(UNFOLD PROPIEDAD, TIEMPO))))

De forma similar, otras funciones (*MININTERV*, *MAXINTERV*, *START*, *STOP*, etc) han sido definidas empleando *FOLD* y *UNFOLD*, como así ha quedado reflejado en la extensión del lenguaje SQL[40]..

2.3 Alternativas en el modelo relacional: incorporación del tiempo

En el modelo relacional, se han presentado dos alternativas sobre el nivel de introducción de la referencia temporal, que han permitido establecer una clasificación de los modelos de validez temporal. La primera de ellas contempla la inclusión del tiempo de validez y/o el de transacción (dependiendo del tipo de base de datos temporal) *a nivel de tupla*, es decir, se indica que la validez temporal de una tupla viene determinada por un atributo temporal, dos en nuestro ejemplo (Desde y Hasta), para reflejar el instante de comienzo y de fin de ese período de validez(Tabla 8). Otra alternativa sería emplear el tipo intervalo definido en el apartado anterior. Esta forma de introducir la información temporal implica que la evolución temporal de una información conlleva la inclusión de varias tuplas. En el ejemplo anterior, la información referente a la propiedad de la parcela L_1 aparece en varias tuplas, siendo el conjunto de todas ellas el que permite ver quién ha sido el propietario de esa parcela a lo largo del tiempo.

Tabla 8 Timestamp a nivel de tupla

Propietario	Parcela	Desde	Hasta
O_1	L_1	1990	1998
O_2	L_1	1999	2000
O_3	L_2	1980	1982
O_4	L_2	1983	1984

La segunda alternativa recoge la información temporal *a nivel de atributo* (Tabla 9). Tanto el valor de timestamp como la característica a la que éste hace referencia se almacenan de forma anidada en un mismo atributo. Al englobar en un mismo atributo la información temporal(tiempo de transacción y/o de validez), se disminuye la redundancia de la información. Sin embargo, esto no evita las anomalías de actualización (*INSERT*, *DELETE*, *UPDATE*) que pueden aparecer en estos casos, debido a como son afectados los valores anidados por la propagación de las modificaciones. Por lo tanto, se trata se trata de un modelo que incumple por tanto la primera forma normal, con los inconvenientes mencionados.

Tabla 9 Timestamp a nivel de atributo

Parcela	**Propiedad**
L_1	$\{[1990, 1998) O_1, [1998, 2000) O_2\}$
L_2	$\{[1980, 1982) O_1, [1982, 1990) O_2\}$

Hasta la fecha, diferentes modelos han sido presentados, ofreciendo diferentes aproximaciones, en función de la alternativa elegida. En [39], se presenta una clasificación de los modelos junto con las propiedades referidas tanto al tipo de datos, las operaciones, etc, como aquellas que son relativas al tiempo, que deberían identificarse en cualquier modelo de datos temporal.

2.4 Bases de Datos Orientadas a Objetos: otra perspectiva

Los diferentes conceptos que se han presentado, ponen de manifiesto la complejidad semántica del "tiempo". En el caso de las bases de datos orientadas a objetos (OODB) y objeto-relacionales(ORDB), se ha tratado de incluir en el modelo de datos esta nueva dimensión pero siempre desde las facilidades que aporta la orientación a objetos. Aunque hay determinados conceptos similares, existen diferentes alternativas, que han de ser tenidas en cuenta al incluirlas dentro del modelo de datos, como por ejemplo el uso de un modelo lineal o con múltiples ramificaciones, discreto o continuo, con simple o múltiples granularidades, etc. Es, en este punto, donde las características de herencia y tipado de la orientación a objetos presentan una solución con la que contemplar las diferentes aproximaciones.

El empleo del sistema de tipos en la orientación a objetos para estructurar el espacio de diseño de los modelos de objetos temporales e identificar las dependencias dentro, y entre, las diferentes dimensiones permite simplificar la representación del dominio temporal. Se trata de definir un *tipo abstracto de dato tiempo* para modelar las características más generales y en cada uno de los subtipos que se definan a partir de él, especializar aquellas semánticas que sean necesarias para cada una de las aplicaciones que se intenten abordar.

El problema de las aproximaciones OODB y ORDB es la carencia de un estándar de facto, como ocurre en el caso del modelo relacional. Las diferentes alternativas de modelos de datos que se están empleando actualmente, son definidas normalmente por los propios desarrolladores del modelo temporal. Aportaciones similares a las comentadas, ha sido la presentada por Bertino et al.[3] en la que, tras la definición del modelo de datos, se propone una extensión del mismo para incorporar esa dimensión temporal. Aunque actualmente se están planteando otras opciones que emplean el estándar ODMG, todavía no existen trabajos con una base formal fuerte en este campo, como los relativos al chequeo y sistema de tipos.

3 Bases de Datos Espaciales

Para los intereses de este trabajo se pueden considerar las *bases de datos espaciales* como un DBMS con capacidades adicionales para la representación y manipulación de datos con una referencia en el espacio [29]. Éste puede ser considerado como la superficie terrestre, al intentar capturar la información proporcionada por la geografía o la cartografía; los objetos hechos por el hombre mediante diseño VLSI, o cualquier otro tipo de información que necesite de una referencia espacial. La característica común a todos ellos es la necesidad de tratar con grandes colecciones de objetos geométricos relativamente simples.

Los datos espaciales poseen algunas características que hacen inapropiada la utilización de los DBMS tradicionales, motivando la aparición de las bases de datos espaciales con objeto de permitir su gestión de una forma eficiente. Algunas de esas características son:

- La complejidad de las estructuras de datos necesarias para su almacenamiento, como es el caso de un polígono, hace inviable la posibilidad de emplear los tipos de datos tradicionales, dado que se precisa registrar un conjunto de puntos que lo delimiten, puntos que pueden ser d-dimensionales. Esto hace las DBMS tradicionales, como pueden ser las relacionales con un tamaño de tupla fijo, inapropiadas en este dominio. Además la necesidad de contemplar toda una serie de operaciones, como la intersección o la unión que posteriormente (ver apartado 3.2), que necesitan una implementación eficiente en el DBMS, hace inevitable la extensión o definición de un modelo que les de soporte.
- La necesidad de ofrecer estructuras de datos que permitan tanto la inserción, como el borrado y actualización de una forma robusta y con un rendimiento elevado, sin que el coste de procesamiento sea demasiado elevado.
- El elevado volumen de datos con el que se suele trabajar en aplicaciones que tratan con sistemas VLSI o con información geográfica como los Sistemas de Información Geográfica, pone de manifiesto la necesidad de emplear sistemas de almacenamiento secundario que permitan su almacenamiento y recuperación de forma eficiente.
- La carencia de un álgebra espacial estándar, implica la imposibilidad de utilización de un conjunto de operaciones cuya semántica esté claramente definida. En su lugar, se encuentran diversas propuestas en este campo que en muchas ocasiones dependen demasiado del dominio de aplicación.
- La dimensionalidad de los datos espaciales supone a su vez otro problema, debido a la inexistencia de un orden total que permita aplicar los predicados existentes en las bases de datos tradicionales. Este orden se refiere a la imposibilidad de, por ejemplo, realizar una consulta según los objetos espaciales sean mayores que uno dado. No existe una definición del concepto mayor sobre los datos espaciales.

– El problema de cierre de las operaciones espaciales aparece con demasiada frecuencia, debido a los diferentes tipos sobre los que se pueden aplicar. Tomando como ejemplo el caso de la intersección entre dos objetos espaciales, el resultado podría ser tanto un polígono, como una línea o un punto, o una serie de estos objetos. En muchas aproximaciones se desprecia parte del conjunto de posibles soluciones debido a deficiencias en el modelo en el que se basa.

Independientemente del modelo de DBMS que se emplee sea orientado a objetos, relacional u objeto-relacional, se trata de realizar una extensión de en su arquitectura de forma que soporte la representación de estos objetos geométricos. Esta extensión significa, en primer lugar, la modificación del modelo de datos para permitir la inclusión de nuevos *tipos abstractos de datos espaciales*, tales como puntos, líneas o regiones, como detallaremos posteriormente, que permitan recoger la geometría de las entidades que queremos almacenar y, en segundo lugar, el conjunto de operaciones que definen su comportamiento. Junto a la inclusión de estos tipos de datos es imprescindible la extensión de los lenguajes tanto de consulta como de definición, que permitan manipular datos espaciales, así como un conjunto de técnicas de indexación adaptadas al tipo de datos con el que se esté trabajando, que permitan obtener un alto rendimiento del sistema.

Extensiones que comprenden los puntos antes mencionadas se encuentran en diferentes bases de datos comerciales como el cartucho espacial de Oracle [47], la extensión de Informix [34], DB2 [9] o el SDE [12] propuesto por ESRI. Junto a los anteriores, existen otros desarrollos académicos como Geo++ [65], Paradise [14] o Dedale [26]. Se trata en todos ellos de ofrecer persistencia a información espacialmente referenciada, con las características proporcionadas por un DBMS en su gestión en aspectos como integridad, control de accesos, concurrencia, etc, pero con una diferencia apreciable: la mayor generalidad de las segundas frente a las primeras, les permitiendo su aplicación a una gran cantidad de campos.

En los siguientes apartados, se muestran las aproximaciones más representativas sobre cada uno de las áreas de interés antes mencionadas, para la extensión de un sistema gestor de estas características.

3.1 Modelos de datos espaciales

El propósito de cualquier *modelo de datos*, es proporcionar un medio formal con el que representar un determinado tipo de información, así como un conjunto de operaciones que permitan su manipulación. Esa representación trata de abstraer un determinado aspecto de la realidad. En este caso, el espacio d-dimensional que es por naturaleza infinito y más concretamente, el espacio Euclídeo en la mayoría de las aproximaciones. Desafortunadamente, los modelos de datos que se presentarán seguidamente, dan una representación finita de esos

infinitos e incluso innumerables conjuntos de puntos que componen los objetos espaciales registrados en las bases de datos, llevando en ocasiones a una pérdida inevitable de información. No hemos de olvidar que cualquiera de las máquinas existentes en la actualidad, tienen una capacidad de representación limitada(apartado 3.1.4).

Un modelo de datos es considerado generalmente como una abstracción, que incorpora solamente aquellas propiedades pensadas como relevantes a la aplicación o aplicaciones con las que se está tratando, usualmente una conceptualización humana de la realidad sin connotaciones tecnológicas en su mayor nivel de abstracción, es decir, sin considerar donde será implementado. Pero, la implementación del mismo conlleva generalmente a tomar decisiones en las que se prima el rendimiento frente a la precisión en la representación. Esta decisión impacta directamente en las características de almacenamiento, manipulación y recuperación de las estructuras tanto física como de los datos en sí mismos. Se hace necesario examinar esos temas utilizando un conjunto de criterios basados en el uso, de forma que la calidad total o la idoneidad de un modelo de datos específico puedan ser evaluados dentro de un contexto particular. Los criterios generales son:

- *completitud*, que puede plantearse en términos de la proporción de todas las entidades y relaciones de un fenómeno particular, existentes en la realidad y que son representadas en el modelo.
- *robustez*, como el grado con que el modelo de datos puede acomodarse a circunstancias especiales o poco usuales, tales como un polígono como un agujero en su interior.
- *versatilidad*, permitiendo contemplar situaciones no recogidas inicialmente en el modelo.
- *eficiencia*, incluye tanto la compactación, es decir, eficiencia de almacenamiento, como la velocidad de uso entendida como eficiencia temporal.
- *facilidad de generación*, es la cantidad de esfuerzo necesario para convertir datos de un formato a algún otro requerido por el modelo de datos.

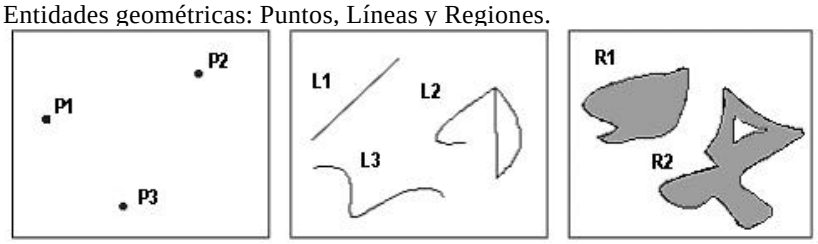
El grado de variación de cada uno de estos factores ha de tomarse en consideración para cualquier campo de aplicación. La importancia relativa de cada factor es una función de los tipos de datos particulares a ser empleados y de los requisitos operacionales totales del sistema. Por ejemplo, si la base de datos a ser generada tiene un volumen elevado y tiene que ofrecer un rendimiento en un contexto interactivo, sería adecuado un compromiso entre los tres primeros factores, dado que la eficiencia total y la facilidad de generación predominarían.

Es posible medir cuantitativamente el rendimiento de varios de estos factores como la velocidad y la eficiencia en espacio para un modelo de datos particular. No sin embargo, proporcionar medidas cuantitativas para los factores más abstractos como completitud, robustez o versatilidad. Este hecho combinado con

el escaso conocimiento sobre las características de rendimiento de un amplio rango algoritmos de procesamiento espacial así como su interacción con otros algoritmos y modelos de datos, indica que el proceso de modelado de datos es mucho mas un arte que una ciencia. Para ellos, la experiencia y la intuición se mantendrán como factores clave en la interpretación de las especificaciones de requisitos de sistemas poco definidos y la construcción de modelos de datos satisfactorios, particularmente para sistemas de información geográfica integrados y completos.

En la definición de estos modelos de datos encontramos dos alternativas claramente diferenciadas. La primera de ellas realiza una definición abstracta de los tipos de datos de forma que permita su posterior integración en el DBMS con independencia de si éste es relacional, objetual o de cualquier otro tipo. En cambio, la segunda alternativa hace uso del modelo utilizado por el sistema gestor que pretende extender para realizar su definición. Existe un punto en común a todas las alternativas que es la semántica asociada a los tipos de datos que se pretenden incorporar.

Fig. 2: Entidades geométricas: Puntos, Líneas y Regiones.



Según podemos apreciar en la Fig. 2, los tipos de datos espaciales que, normalmente, solemos encontrar en las diferentes aproximaciones [22, 31, 55, 41], para describir la geometría de diferentes objetos en el espacio, son *puntos* (*P1, P2, P3*), *líneas* (*L1, L2, L3*) y *regiones* (*R1, R2*). Generalmente, la definición de estas entidades se realiza en el espacio bidimensional aunque su extensión a la n-dimensión es relativamente sencilla.

Un punto representa un objeto del cual sólo nos interesa conocer su localización en el espacio, no así su extensión; por ejemplo, la localización de una ciudad, siendo su forma de representación más habitual mediante dos coordenadas (x, y) en el espacio. Una línea es comprendida como una secuencia de puntos y por tanto como una secuencia de coordenadas (x, y), normalmente representa conexiones en el espacio como podrían ser carreteras, ríos, etc. Una región, en cambio, es entendida como una superficie encerrada por una serie de líneas conectadas, podría pensarse como una abstracción de una extensión bidimensional, tal como un municipio o una provincia. En la definición de las regiones, puede contemplarse la posibilidad de tener agujeros en su interior (R2), lo cual se conoce como *región cóncava*. Otras aproximaciones que permiten

considerar diferentes regiones inconexas como una única región, es decir, según el ejemplo de la Fig. 2, R1 y R2 podrían pertenecer a la misma región.

Las *álgebras espaciales*, capturan junto con los tipos de datos espaciales, un conjunto de operaciones que permiten su manipulación y una serie de predicados que ayudan a establecer sus relaciones entre ellos, permitiendo su posterior incorporación al DBMS. Este tipo de operaciones, varían dependiendo del modelo con el que se trate, no sólo sintáctica, sino también semánticamente. Aportaciones que tratan de dar una definición más homogénea de los diferentes predicados, han sido los trabajos sobre relaciones espaciales que veremos posteriormente (apartado 3.2).

3.1.1 El modelo ráster

En este tipo de modelo la información geométrica es intencionalmente descrita por un número finito de puntos ráster. La semántica en este modelo es que el número infinito de puntos en el entorno de un punto ráster p tienen sus mismas propiedades. Los puntos ráster están uniformemente distribuidos sobre el objeto que está siendo representado. Aunque el modelo es bastante intuitivo, presenta algunos problemas relacionados con la distribución, en ocasiones heterogénea, de las propiedades relevantes, por ejemplo una recta que no se ajusta a los puntos que conforman el ráster, dando lugar a problemas con el cierre de las operaciones.

Un ejemplo, es la conocida como *ROSE álgebra* [31]. Se trata en este caso de un álgebra en la que se definen, con independencia de modelo del DMBS, tres tipos de datos, points, lines y regions cuyos valores están fundados en una base geométrica discreta conocida como *realm*[28]. Un realm representa la geometría subyacente de un espacio de aplicación particular. Para ello se define como un conjunto finito de puntos y líneas de segmento sobre un grid discreto tal que:

- cada punto o punto final de una línea segmento es un punto del realm;
- cada punto final de una línea segmento es también un punto del realm;
- ningún punto del realm cae dentro de una línea segmento si no es un punto final;
- nunca dos segmentos intersectan salvo en sus puntos finales.

Emplear este tipo de formalismo ofrece diversas ventajas al tratar los problemas relativos a la robustez numérica a nivel de la capa geométrica, sin que estos afecten a las álgebras o tipos de datos definidos sobre ellos, cumpliendo así con propiedades de cierre no solo en la teoría sino también en la implementación. Además permite, de una forma sencilla, representar colecciones de objetos que se encuentren espacialmente relacionadas. Al mantenerse los bordes comunes entre objetos, se proporcionan ventajas en cuanto al cierre de aquellas operaciones basadas en líneas del realm y a la consistencia geométrica de objetos

espacialmente relacionados. En cambio, aunque evita los problemas relativos a la corrección y robustez numérica al garantizar que ningún nuevo punto de intersección es calculado en el procesamiento de consultas, si plantea problemas de consistencia entre consultas.

Fig. 3: Realm (a) R-point, (b) R-face, (c) R-block

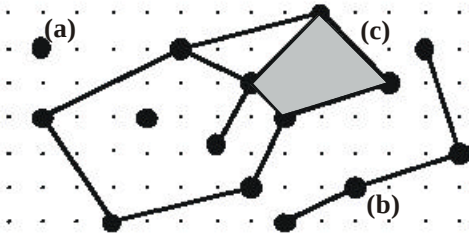
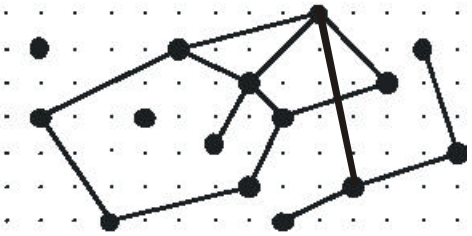
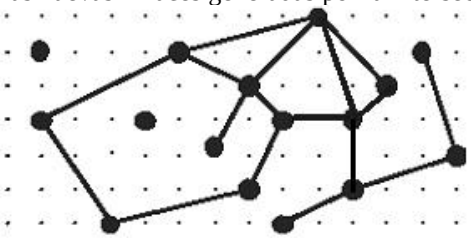


Fig. 4: (a) Realm tras la inserción de una nueva R-face intersectando con otras ya existentes.



(b) Realm con las nuevas R-faces generadas por la intersección



El procedimiento de inserción de un nuevo objeto depende de las líneas que existan en el realm, de tal manera que si existe la necesidad de incorporar nuevas líneas que intersectan con las existentes(Fig. 4. (a)), nuevos puntos de intersección tienen que ser incorporados. Esto puede llegar a provocar que algunas líneas del realm se vean modificadas(Fig. 4. (b)), afectando por tanto a aquellos objetos que están formados por ellas. Esto significa que puede haber ocasiones en las que una determinada consulta produzca resultados diferentes en dos momentos distintos.

Mediante esta estructura se definen los conceptos:

– *R-point*, cualquiera de los puntos definidos en el realm(Fig. 3 (a));

- *R-block*, conjunto conectado de segmentos de línea(Fig. 3 (b));
- *R-face*, un polígono con agujeros construido con segmentos del realm(Fig. 3 (c));

que permiten introducir los diferentes tipos espaciales:

- *points*, conjunto de R-points;
- *lines*, conjunto disjunto de R-blocks;
- *regions*, conjunto de R-faces cuyos bordes son disjuntos aunque tengan puntos comunes.

Otra característica de este álgebra es la definición de operaciones polimórficas mediante la cuantificación de tipos de datos, es decir, en la ROSE Algebra se distinguen dos conjuntos de tipos sobre los que se aplican esas operaciones, EXT={*lines, regions*} y GEO={*points, lines, regions*}. En base a esos dos conjuntos se definen las signaturas de cuatro clases de operaciones que a continuación se enuncian [28]:

- Predicados espaciales sobre los que se definen relaciones topológicas. Por ejemplo:

  $\forall$ geo in GEO. $\forall$ ext_1, ext_2 in EXT
  geo x regions $\rightarrow$ bool inside
  ext_1 x ext_2 $\rightarrow$ bool intersects, meets

  Los conjuntos de tipos se refieren a los objetos que soportan esas operaciones. Así, carece de sentido definir la operación *incide* sobre el tipo punto, dado lo ilógico de averiguar si una región está dentro de un punto.
- Operaciones que devuelven tipos de datos espaciales atómicos, como podría ser:

  $\forall$ geo in GEO
  lines x lines $\rightarrow$ points intersection
- Operadores espaciales que devuelven valores numéricos:

  $\forall$ geo_1, geo_2 in GEO
  geo_1 x geo_2 $\rightarrow$ real dist
- Operaciones espaciales sobre conjuntos de objetos, que recogen operaciones agregadas. Por ejemplo para realizar la unión de todos los objetos geométricos y calcular la suma de valores de un determinado atributo:

  $\forall$ obj in OBJ, $\forall$geo,geo_1, geo_2 in GEO
  set (obj) x (obj$\rightarrow$geo) $\rightarrow$ points sum

Una de las características de este modelo es la *extensibilidad* de la arquitectura del sistema, dada la posibilidad de poder definir nuevos tipos de acuerdo a las necesidades del dominio concreto donde se vayan a aplicar. Incluye también diferentes operaciones que permiten la conversión entre diferentes tipos, así como completitud en cuanto al grado de objetos espaciales que son representables mediante este modelo. Así mismo, es eficiente tanto en términos de utilización de almacenamiento, debido a la escasa duplicación de información,

como temporal, gracias al cálculo de las intersecciones en el momento de inserción de los objetos espaciales.

Otra aproximación es la conocida como *Quantum Álgebra* [41] al igual que la anterior se trata de un álgebra definida para el espacio bidimensional, aunque sus autores aseguran su validez para el espacio n-dimensional. Es especificada como una extensión al álgebra relacional, basándose en un formalismo conocido como *spatial quanta* define los tipos de datos espaciales primitivos. Para ello, dado un subconjunto de los números enteros se definen tres tipos primitivos(Fig. 5):

- *Quantum points*. Conjunto de puntos que cumplen:
  $Q_{POINT} = \{\{p\} \mid p = (i, j) \in I_m^{\,2}\}$
- *Quantum lines*. (Fig. 5(a y b)) Dados dos puntos p, q $\in Q_{POINT}$ cuyas coordenadas son, respectivamente, (i,j) y (k,l), entonces se definen las quantum lines como el conjunto de líneas tal que:
  $ql_{pq} = \{(x, y) \in R^2 \mid i \le x \le i+1 \wedge y = j\}$, conocida como pure horizontal quantum line, o
  $ql_{p,q} = \{(x, y) \in R\ 2 \mid x = i \wedge j \le y \le j+1\}$, conocida pure vertical quantum line.

  Dadas *pure quantum lines* (Q_{PL}) como el conjunto de todas las pure horizontal quantum line y pure vertical quantum line, se define $Q_{LINE} = Q_{PL} \cup Q_{POINT}$, como el conjunto de todas las quantum lines. De ello, se deduce que el conjunto Q_{POINT} se consideran quantum lines degeneradas.
- *Quantum surfaces*. Dados p, q, r, s $\in Q_{POINT}$, ordenados en sentido horario como cuantro puntos esquina, cuyas coordenadas son (i, j), (i+1, j), (i+1, j+1) y (i, j+1), respectivamente, podemos definir una *pure quantum surface* como:
  $qs_{p,q,r,s} = \{\{x,y\} \in R^2 \mid i \le x \le i+1 \wedge j \le y \le j+1\}$

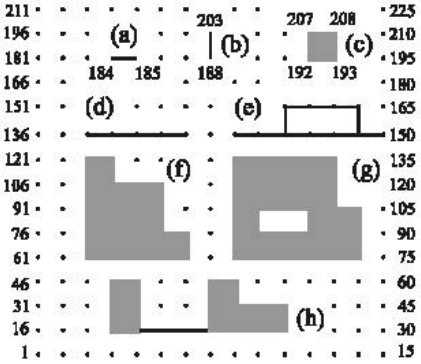
  Una quantum surface consiste de un número infinito de elementos R^2, que puede ser geométricamente interpretada como un cuadrado cuyas cuatro esquinas son los puntos p, q, r, s $\in Q_{POINT}$ y cuyos vértices son las cuatro quantum lines $ql_{p,q}$, $ql_{q,r}$, $ql_{r,s}$, $ql_{s,p}$. En la Fig. 5 se puede apreciar como se forma una quantum surface, $qs_{192, 193, 208, 207}$, para el objeto etiquetado como (c).

  Si Q_{PS} denota el conjunto de todas las quantum pure surface, podemos definir el conjunto de todas las quantum surfaces como $Q_{SURFACE} = Q_{PS} \cup Q_{LINE}$, de manera que se consideran las Q_{PS} y Q_{LINE} como superficies degeneradas.

Mediante la definición de estos tipos primitivos es posible la introducción de los tipos de datos espaciales *POINT*, *LINE* y *SURFACE*, a los que se conoce en su conjunto como *GEO* en la Quantum Algebra[41]. Para ello se hace uso de la conectividad entre los tipos simples, de manera que una LINE se considera la unión de todas las quantum lines que se encuentren conectadas, tal y como se aprecia en la Fig. 5, los objetos etiquetados como a, b, c y e. No se puede olvidar

que los puntos se consideran líneas degeneradas. De manera similar, las superficies (Fig. 5) se forman mediante la unión no solamente de las pure quantum surface, sino también con las pure quantum lines(objeto h), con el único requisito de existir conectividad entre las diferentes entidades.

Fig. 5: Quantum spatial objects. (a) pure horizontal quantum line, (b) pure vertical quantum line, (c) pure quantum surface, (d, e) line, (f, g, h) surface.



Junto a las definiciones de esos tipos simples también se definen diferentes predicados y funciones espaciales, como: *RANK* que devuelve el rango de un tipo GEO (0 para un punto, 1 para una línea y 2 para una superficie); *OVERLAP* que indica si un objeto solapa a otro (ver semántica en el apartado 3.2); etc.

En la extensión al álgebra relacional, una vez introducidos los anteriores tipos, podemos realizar la definición de los atributos de cualquier relación del esquema, y la definición de las operaciones sobre ellas. Para ello define el resultado de las operaciones del álgebra relacional:

UNION:	U = R_1 Union R_2	
EXCEPT:	E = R_1 Except R_2	
INTERSECT:	I = R_1 Intersect R_2	
PROJECT:	P = R_1 Project R_2	
PRODUCT:	P = R_1 Product R_2	
SELECT:	S = Select [F] R	donde F es una fórmula bien formada
JOIN:	J = R_1 Join[F] R_2	similar a Select

Esta definición sería similar a la extensión realizada en el álgebra relacional para intervalos [39]. Por ejemplo, para aplicar la operación Intersect sobre dos relaciones $R_1(G_1, A_1)$ y $R_2(G_2, A_2)$, donde G_1 y G_2 son de tipo GEO aplicaría el operador unfold para descomponer cada una de los objetos espaciales en sus formas quantum básicas y sobre ellas realizaría la operación intersect, es decir, se

simplifica enormemente la operación. De igual forma, se realizaría para el resto de las operaciones. Además, se aduce en el mencionado trabajo[41], la posibilidad de definir cualquier otra de las operaciones del álgebra en base a este conjunto básico.

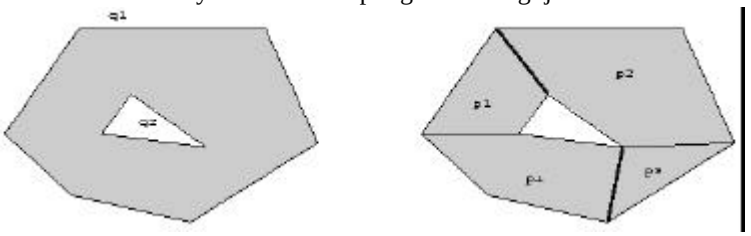
Podemos apreciar que al igual que en el caso de la Rose Algebra[28] se fundamenta este modelo en una base geométrica discreta. La principal diferencia estriba en no permitir la definición de líneas oblicuas, solucionando de esta manera los problemas derivados del cierre de las operaciones, dado que cualquier intersección siempre se encontrará formada por los quantum points o quantum lines. No tiene en consideración los problemas, mencionados anteriormente, relacionados con la granularidad o pérdida de información derivada de la definición de los objetos en función de líneas rectas. Si en cambio posee características relativas a la completitud, robustez, versatilidad y facilidad de generación. La eficiencia temporal quedaría en función de la capacidad de aplicación de las operaciones fold y unfold, operaciones básicas para la definición del resto de operaciones del álgebra.

3.1.2 El modelo polinomial

El modelo de datos con restricciones introducido por Kanellalis et al.[37] trata de mostrar un paradigma para la representación de todos los tipos de datos desde un marco unificado. Las restricciones lineales sobre los números racionales han demostrado, mediante la representación modo vector, su idoneidad para la representación de datos espaciales. Este formato de representación se enfrenta con una representación en la que se emplean los bordes de un objeto para representar los infinitos puntos que lo componen, y que conlleva la inexistencia de un estándar que permita aportar una solución única a este problema.

La idea básica de este modelo es representar los objetos espaciales como colecciones infinitas de puntos en el espacio de los racionales (Q^2) que satisfacen una formula de primer orden. Por ejemplo, un polígono convexo, que es la intersección de un conjunto de *medios planos*, es definido mediante la conjunción de las inigualdades que definen cada uno de ellos. Un polígono no convexo es definido mediante la unión de un conjunto de polígonos convexos. Aunque no restringe el tipo de objetos que pueden ser almacenados en la base de datos, si en cambio obliga a una descomposición del espacio en componentes convexos en el momento de la carga de la información. Aunque esto conlleva un mayor coste de procesamiento inicial, simplifica enormemente cualquier consulta posterior. Este tipo de representación se conoce como DNF (Disjunctive Normal Form, Fig. 6). En cambio, ese procesamiento puede ser evitado si se permite la definición de polígonos con agujeros, es decir, la representación conocida como CHNF (Convex with Holes Normal Form, Fig. 6).

Fig. 6: Representación DNF y CHNF de un polígono con agujeros



De esta forma y empleando un lenguaje de primer orden, se definen las restricciones lineales mediante el lenguaje $L=\{\leq, +\} \cup Q$, sobre la estructura del conjunto ordenado de los números racionales con suma y constantes racionales. Las restricciones se definen como ecuaciones lineales e inigualdades de la forma $\sum_{i=i}^{p} a_i x_i q\, a_0$, donde θ es uno de los predicados = ó ≤, las x_i denotan variables y los a_i son constantes enteras, siendo el número de variables libres la *aridad* de la fórmula. Así, dado $\sigma = \{ R_1, ..., R_n \}$ un esquema de la base de datos tal que $L \cap \sigma = \varnothing$ y siendo $R_1, ..., R_n$ símbolos de relación, podríamos se definen las diferente formulas DNF y CHNF como:

– La definición de $S \subseteq Q^k$ sea una relación de aridad k, la relación S es una relación de restricciones lineales si existe una fórmula $\rho(x_1, ..., x_k)$en L con k distinto de las variables libres $x_1, ..., x_k$ tal que:

  $Q \mid \forall\ x_1, \ldots, x_k\ (S\ (x_1, \ldots, x_k) \leftrightarrow \rho(\ x_1, \ldots, x_k))$

  De tal manera que una formula DNF $\rho = \bigvee^{k}_{i=1} \bigwedge_{j=1}$ es vista como una fórmula generalizada, es decir, un conjunto infinito de tuplas generalizadas, que son conjunciones de formulas atómicas $\rho_{i,j}$

  $\{t_i \mid 1 \leq i \leq k_i\ t_i = \bigwedge_{j=1} \rho_{i,j}\}$

– Una fórmula está en forma CHNF si está libre de cuantificadores de libertad en formato DNF, o bien está en la forma F-F', donde F y F' son dos fórmulas CHNF de la misma aridad y – es la diferencia de conjunto, (es decir $\wedge\neg$).

Una instancia de la base de datos asocia a cada símbolo R de relación de aridad k en σ, una relación de restricciones lineales de aridad k. Un ejemplo de una relación GO de ocupación del terreno, se encuentra en la Fig. 7(a), donde (b) y (c) presentan la información asociada y la interpretación geométrica, respectivamente.

Cuya interpretación geométrica podría ser la que podemos observar en la Fig. 7:

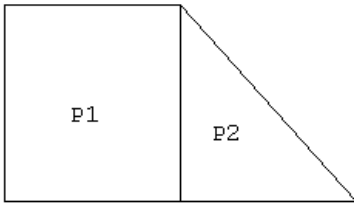
Fig. 7: (a) Relación GO de ocupación del terreno en un modelo con restricción.

```
GO={[nombre:String, tipo-suelo:string,
       propietario:string, geometria: {Q²}]}
```

(b) representación simbólica de la relación previa, con la geometría incluida

nombre	tipo-suelo	propietario	geometria
p1	bosque	Pablo	$\{x \leq 1 \wedge x \geq -1 \wedge y \leq 1 \wedge y \geq -1\}$
p2	gramíneas	María	$\{x \geq 1 \wedge y \geq -1 \wedge x + y - 2 \leq 0\}$

(c) Interpretación geométrica del atributo "geometría"



El álgebra quedaría definida mediante una extensión del álgebra relacional, con operadores específicos para los tipos anteriormente introducidos, es decir, sobre los operadores de conjunto, de selección, de proyección, reestructuracion, anidamiento y desanidamiento, que se encuentran definidos dentro del álgebra relacional y que se aplicarían sobre relaciones con restricciones, en función de los cuales se podrían definir el resto de operadores necesarios[26].

Aunque este álgebra permite tratar el espacio como un conjunto infinito de puntos, algo no contemplado por el resto de álgebras antes introducidas, es indudable la complejidad inherente no solo a la definición de cualquier objeto espacial, sino a la propia comprensión del modelo que está empleando, es decir, no es próxima a la intuición.

3.1.3 Modelo de datos topológico

El interés principal de este tipo de modelo es ofrecer una base forma para aquellas consultas en la base de datos con referencia a las relaciones topológicas de los objetos (ver apartado 3.2), tales como adyacencia o solapamiento. Una característica inherente de las bases de datos que soportan este tipo de relaciones es la equivalencia entre las bases de datos que son generadas mediante deformaciones topológicas, es decir, son equivalentes topológicamente.

Este tipo de modelo se aplica generalmente en dominios enfocados a modelar relaciones entre los objetos. Un caso de un dominio de aplicación es un sistema de control de tráfico donde, una de sus necesidades básicas es registrar los enlaces entre diferentes carreteras. Estas relaciones permiten, por ejemplo, determinar las posibles rutas alternativas.

Una de las alternativas, conocida como GaphDB [30], presenta un modelo de datos en el que se incluyen:

- *Tipos de datos*, que están parcialmente ordenados e incluyen los tipos numéricos NUM={INTEGER, REAL} y los tipos geométricos GEO={POINTS, EXT} donde EXT={LINES, REGIONS}. La introducción de esta ordenación permite la definición de funciones polimórficas, al igual que ocurría en la Rose Algebra (ver apartado 3.1.1).
- *Tipos de objetos*, que se identifican con las clases en el modelo de objetos, quedan referenciados mediante un identificador, siendo su relación de orden parcial la determinada por la jerarquía de clases. Mediante ese identificador, OID, podemos determinar de que clase es la instancia y por tanto su situación en la jerarquía.
- *Tipos de tuplas* son conjuntos de atributos cuyo tipo puede ser uno de los anteriormente mencionados, es decir, tipos de datos o tipos de objetos.

La utilización de tipos de objetos, permite la definición de una clase especial denominada *Link*(enlace) como una cuadrupla (*c, T, d, e*). Donde *c* y *T* son una lista de atributos de algunos de los tipos de datos y de tuplas, respectivamente; y *d* y *e* son nombres de clases entre las cuales se desea establecer el enlace. Es decir, es un conjunto de cuadruplas que son subconjunto de *oids(c) x valores (T) x oids(d) x oids(e).*

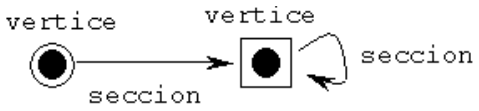
En este modelo, una base de datos es un par (C,≤). Donde C es un conjunto finito de clases definidas mediante un par (ctype(c), extensión(c)); mientras que ≤ es el operador que define un orden parcial sobre el árbol de clases. El conjunto de clases C se encuentra fraccionado en tres subconjuntos C=SC∪LC∪PC denominados *clases simple*, *clases enlace*(link) y *clases camino* (path), respectivamente.

Mediante estas cuadruplas es posible definir grafos de instancia, formados por dos conjuntos, nodos y bordes, que permiten realizar dos mappings entre *origen* y *destino* (source y target) desde los bordes a los nodos, dado que es posible la definición de múltiples bordes entre los dos mismos nodos. De esta forma, un esquema de la base de datos para la representación de un grafo se define como *SG= (S, L, source, target)* donde:

- S={c|((c, T), ext) ∈ SC}
- L={c|((c, T, d, e), etx) ∈ LC}
- source: L→S se define como source(c)= d ⇔ ((c, T, d, e), etx) ∈ LC
- target: L→S se define como target (c)= e ⇔ ((c, T, d, e), etx) ∈ LC

Por ello, para cada clase simple y cada clase de enlace hay un nodo y un borde, respectivamente, en el grafo de esquema.

Fig. 8: Grafo



Un tipo camino se considera como un autómata finito, perteneciente a una expresión regular sobre los nombre de clase, con *s* como su estado inicial y *F* su estado final. Se puede definir mediante la cuádrupla *(G, m, S, F)* donde:

- G=(V, E, source, target) es un grafo conectado
- *m: VÈE ®OT* es un función de etiquetado de nodos y borde de G con tipos de objetos tal que:
- $v \in V \Rightarrow \exists((c, T), ext) \in SC : \mu(v)=c$
- $e \in E \Rightarrow \exists((c, T), a, b) \in LC : \mu(e)=c \wedge \mu(source(e))=a \wedge \mu(target(e))=b$
- *s Î V* el nodo de comienzo
- *F Í V* el nodo final

La aplicación de este modelo al ejemplo anterior de la red de carreteras sería:

```
create class vertice= pos: POINT;
create vertice class union = name: STRING;
create vertice class salida = nr: INTEGER;
create link class seccion = ruta: LINE,
   no_caminos: INTEGER, max_velocidad: INTEGER
   from vertice to vertice;
create path class class carretera = name: STRING as section+;
```

Las clases salida y union son modeladas como subclases de la clase vértice, es decir , heredan su atributo pos, por lo que ambas clases pueden ser empleadas como origen y destino de los bordes de seccion. Mediante carretera se definiendo un conjunto de caminos en la base de datos formados mediante secciones. La Fig. 8 muestra un ejemplo de los grafos que se podría generar.

Obviamente, la posición exacta de los objetos, así como su geometría son registrados gracias al empleo de los tipos GEO. Se emplea, un entorno de modelado orientado a objetos, para facilitar su posterior adecuación a los diferentes dominios de aplicación. Su definición se aleja bastante de los modelos polinomial y ráster, dado que se centra específicamente en recoger esas relaciones entre los diferente objetos que la componen, en vez de registrar la información espacial como ocurría en los casos anteriores.

3.1.4 Un problema acerca de la granularidad

La imprecisión está directamente relacionada con la *granularidad* o resolución con que los datos son observados y con el medio de representación y de procesamiento, es decir, la medida en que son discernibles los elementos de un fenómeno que está siendo representado por un conjunto de datos, tal y como

ocurría para las bases de datos temporales. Su importancia deriva de su influencia directa en la heterogeneidad de los diferentes componentes de las bases de datos. Heterogeneidad entendida en diferentes dimensiones y contextos, como las diferencias geométricas y semánticas, así como la heterogeneidad de la calidad de los datos y metadatos. La cada vez mayor distribución de la información, tanto por necesidades de procesamiento como de almacenamiento de los volúmenes que suelen tratar este tipo de sistemas, hace necesario la definición de una semántica común que permita su integración entre diferentes conjuntos de datos sin pérdidas excesivas en el nivel de representación. Se trata de dotar al modelo de datos de capacidades que permitan mitigar la influencia de la multiresolución en la calidad de la información. Por ejemplo, al fusionar dos conjuntos de datos con distinta resolución espacial, se deberían unificar las resoluciones para su posterior procesamiento. La pregunta que surge es ¿cuál es la resolución más apropiada?

Worboys [68], presenta una alternativa que podría facilitar estas consideraciones, para el dominio de la información geográfica,, al proporcionar una base matemática que permita la conversión de diferentes conjuntos de datos de diferentes resoluciones o granularidades sin una pérdida significativa de información. Para ello define una estructura piramidal de granularidades que nos permitan una conversión directa entre una y otra.

3.2 Relaciones espaciales

En el tratamiento de la información espacial es necesario el establecimiento de una definición formal de las relaciones espaciales entre los objetos geométricos. Las *relaciones topológicas* suponen un conjunto de las anteriores, caracterizadas por permanecer inalteradas ante transformaciones topológicas como el escalado, la traslación y la rotación. La formalización de estas relaciones permite un establecimiento más sencillo de las consultas, así como su posterior procesamiento por el DBMS. Esto tiene especial significado en el área de los Sistemas de Información Geográfica, donde las operaciones a realizar se establecen sobre la base de las relaciones espaciales. Además, su introducción trata de suprimir la carencia mostrada por los lenguajes de consulta tradicionales como SQL o QUEL, en donde las relaciones de igualdad o de orden se establecen entre enteros o strings, pero no entre objetos espaciales.

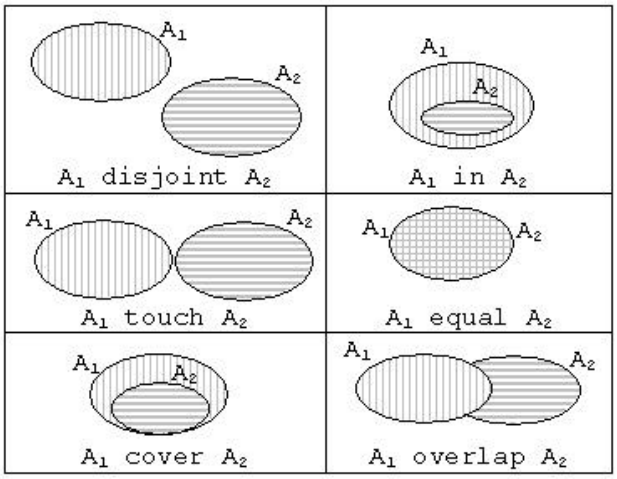
Diversos trabajos se han llevado a cabo en este campo entre los cuales cabe destacar, por el impacto que han presentado sobre todo en la definición de los lenguajes de consultas, los conocidos como *modelo de las 9-intersecciones* [14] y *modelo de las 4-intersecciones* [7], siendo el segundo una simplificación del primero. Se trata de dar un formalismo, similar a un modelo matemático,

basándose en la definición de todas las relaciones posibles entre dos objetos geométricos así como la forma en que pueden ser calculadas.

Fig. 9: Relaciones topológicas según el modelo de las 4-intersecciones

$\neg A_1 \cap \neg A_2$	$\neg A_1 \cap A_2^o$	$A_1^o \cap \neg A_2$	$A_1^o \cap A_2^o$	**Nombre Relación**
∅	∅	∅	∅	A_1 disjoint A_2
∅	∅	∅	¬∅	
∅	∅	¬∅	∅	
∅	∅	¬∅	¬∅	A_2 in A_1
∅	¬∅	∅	∅	
∅	¬∅	∅	¬∅	A_1 in A_2
∅	¬∅	¬∅	∅	
∅	¬∅	¬∅	¬∅	
¬∅	∅	∅	∅	A_1 touch A_2
¬∅	∅	∅	¬∅	A_1 equal A_2
¬∅	∅	¬∅	∅	
¬∅	∅	¬∅	¬∅	A_1 cover A_2
¬∅	¬∅	∅	∅	
¬∅	¬∅	∅	¬∅	A_2 cover A_1
¬∅	¬∅	¬∅	∅	
¬∅	¬∅	¬∅	¬∅	A_1 overlap A_2

Fig. 10: Interpretación geométrica



El modelo de las 4-intersecciones está basado en el cálculo de intersecciones, vacías o no, entre el interior y el borde de un objeto A_1, expresados como A_1^o y

∂A_1, respectivamente, con el interior y borde de otro objeto A_2 con el que se desea establecer la relación. Esto conlleva que se pueden expresar de esta manera hasta un total de 2^4=16 combinaciones posibles(Fig. 9). Teniendo en cuenta que 8 de ellas son relaciones no válidas y dos son simétricas, se obtienen un total de 6 relaciones diferentes denominadas *disjoint*, *in*, *touch*, *equal*, *cover* y *overlap* cuya interpretación geométrica se muestra en la Fig. 10.

El anterior modelo fue extendido mediante la inclusión de las características de puntos y líneas, en lo que se conoce como método de la *dimensión extendida*. Junto a la definición de las intersecciones, Clementini et al.[7] también consideran la dimensión en que se produce la intersección de forma que, en un espacio bidimensional, se considera que el conjunto intersección puede ser ∅ (vacío), 0D (punto), 1D (línea) o 2D (región). Esto provoca que el número de posibles combinaciones se eleve a 4^4=256 combinaciones, de las cuales sólo 52 son consideradas como relaciones válidas entre punto, línea y región con otras características debido a que la dimensión de la intersección no puede ser mayor que la menor dimensión de los dos operandos en la intersección. A pesar de la reducción propuesta, el número de casos es todavía demasiado elevado para su comprensión por parte del usuario.

La alternativa consiste en reducir el número de relaciones posibles a un conjunto básico, 5 en este caso (*disjoint*, *in*, *touch*, *cover* y *overlap*). De forma que al usuario se le proporcionan, a nivel de lenguaje de consulta, estas relaciones junto con tres operadores que nos permiten obtener el borde de una región y una línea (puntos extremos de la misma), simplificando con ello su utilización. Un ejemplo de este tipo sería la definición de la relación *touch* como:

Aplicada sobre dos características o bordes de característica, λ_1 y λ_2, que pueden ser región/región, línea/región, punto/región, línea/línea, punto/línea pero no punto/punto quedaría definida como:

$<\lambda_1,$ touch, $\lambda_2> : \Leftrightarrow (\lambda_1{}^o \cap \lambda_2{}^o = \emptyset) \wedge (\lambda_1 \cap \lambda_2 = \emptyset)$

El modelo de las 9-intersecciones [14] aparece como un superconjunto de anterior, compartiendo muchas de sus características. La diferencia estriba en que las intersecciones se aplican, además de entre interiores y bordes, también sobre el exterior de los objetos, es decir, sobre el espacio que los rodea. Esta diferencia provoca que el número de combinaciones con las que se trata se eleva a 512, sólo entre regiones, con la consiguiente complejidad para el usuario. La motivación de tal trabajo es la deficiencia mostrada por el modelo anterior para distinguir entre ciertas configuraciones topológicamente distintas y que podrían llegar a ser necesarias en los predicados a emplear en los lenguajes de consulta espaciales.

3.3 Lenguajes de consulta

El lenguaje ha de permitir al usuario:

- modelar aquella parte del espacio que desea registrar en la base de datos, es decir, permitir la definición;
- gestionar grandes volúmenes de información compleja como es el caso que nos ocupa, de una forma amigable para el usuario; y,
- permitir la consulta de este tipo información donde la componente visual se muestra tan importante.

Normalmente, la mayoría de los trabajos que se han realizado en este tema, centran sus esfuerzos, en como recuperar la información de la base de datos. Obviamente, las consultas espaciales emplean en su definición conceptos de índole espacial, para seleccionar aquellos objetos que cumplen unos determinados criterios que pueden ser tanto alfanuméricos como espaciales.

El lenguaje de consulta SQL, de uso tan extendido en el entorno relacional, ha mostrado serias deficiencias para su empleo en consultas que utilicen esos conceptos espaciales, dada la deficiencia de las operaciones que en él están definidas para manipular datos espaciales, así como la carencia de operaciones espaciales propiamente dichas.

Egenhofer presenta [15] las deficiencias que plantea SQL para una extensión sencilla del mismo si pretende seguirse el formato estandarizado. Principalmente, en relación a dos temas como son:

- falta de potencia expresiva del lenguaje sobre algunos conceptos fundamentales actualmente como son los metadatos y las consultas de alto nivel, así como la imposibilidad para tratar la identidad de los objetos, es decir, la posibilidad de distinguir a unos objetos frente a otros, una propiedad crucial para poder estudiar su evolución temporal;
- la inconexión entre el lenguaje de visualización y el de recuperación que impide al usuario poder establecer consultas con respecto resultados de consultas ya visualizadas, ni establecer el entorno de visualización de manera que sea fácilmente legible para el usuario.

Egenhofer planteó una serie de requisitos[13] que considera imprescindibles en la definición de cualquier lenguaje de estas características. De ellas destacan:

- La definición de tipos de datos abstractos, junto con las correspondientes operaciones y relaciones, que permitan la manipulación directa de los datos espaciales, independientemente de cómo sea la representación interna de los mismos.
- La posibilidad de presentar los resultados de forma gráfica, junto con aquella información de cualquier otro tipo que tenga relación con la misma, de manera que se permita un análisis conjunto.

- La posibilidad de anidar consultas de cualquier tipo, empleando el resultado de una o más consultas en interacción con las siguientes.
- La facilidad de definir el contexto espacial en el que se están presentando los resultados gráficos.
- Mecanismos para el chequear la integridad de la información visualizada.
- La definición de diálogos extendidos que permitan el empleo de dispositivos de selección gráfica, facilitando con ello una selección directa de la información espacial.
- Definición de diferentes formatos de salida gráfica, en cuanto a colores, patrones, etc, que aseguran un mejor diferenciación de las entidades geométricas presentadas.
- Leyendas descriptivas que permitan reflejar que diferentes clases de objetos se han visualizado conjuntamente.
- La posibilidad de incluir etiquetas desde el lenguaje de consulta, con objeto de poder comprender la visualización.
- Inclusión de escalas que permitan asimilar la dimensión de la información visualizada.
- Definición de herramientas para la delimitación de las áreas de interés, ofreciendo la posibilidad de no tener que tratar con toda la información del DBMS.

Estas carencias, junto con los requisitos mencionados anteriormente, han servido para establecer la mayoría de las propuestas se presentan a continuación a la vez que han servido de elementos de comparación entre ellas (Tabla 10).

Una de las aproximaciones más conocidas es el lenguaje SpatialSQL [16], que representa una extensión mínima del lenguaje SQL, donde se preservan los conceptos introducidos por él y los objetos espaciales se tratan a alto nivel, incorporando operaciones y relaciones espaciales. Una de sus características más representativas es la introducción de dos componentes: un lenguaje de consulta, para la recuperación de la información y un lenguaje de representación, que permite especificar cómo se realiza la visualización de los resultados. Un ejemplo típico en este sistema podría ser mostrar todas las residencias en verde y los edificios comerciales en azul de la ciudad de Orono que están en la calle “Grove Street”. En esta consulta se entremezcla tanto información alfanumérica, como espacial y temas relativos a la visualización. Mediante SpatialSQL en primer lugar deberíamos establecer el entorno gráfico de representación, es decir, los colores y leyendas de los objetos que vamos a visualizar:

```
SET LEGEND
COLOR green, blue
FOR SELECT residence.geometry, commercial.geometry
FROM residence building, commercial building
WHERE residence.type= “Residential”
          and commercial.type=”Commercial”;
```

Posteriormente, el usuario debería identificar el área de interés y asignar el contexto de las carreteras, es decir, en relación a que otros elementos debe establecer su selección:

```
SET WINDOW
   SELECT geometry
   FROM  road
   WHERE  town.name=”Orono”;
SET CONTEXT
FOR road.geometry
         SELECT parcel.geometry, building.geometry,
                                              road.geometry
         FROM road, parcel, building;
```

Finalmente, el usuario selecciona aquellas entidades que desea visualizar sobre una nueva ventana mediante una sencilla consulta SQL:

```
SET MODE new;
SELECT road.geometry
FROM     road, town
WHERE    town.name= “Orono” and
   road.name= “Grove Street” and
   road.geometry INSIDE town.geometry;
```

Este tipo de consultas es habitual en la mayoría de los lenguajes de consulta. En cambio, la definición del entorno de visualización suele descansar sobre las interfaces visuales, como en el resto de lenguajes que se presentan.

Otra propuesta, como extensión a SQL es QL/G [5], definiéndolo desde un punto de vista funcional y modular sobre una extensión del modelo relacional anidado, en la que los tipos y operaciones geométricas son introducidos de forma independiente. Su característica principal es ser fuertemente tipado, de forma que satisface la propiedad de cierre al generar, los operadores, solo tipos de datos existentes en el modelo. Además, estos operadores se definen como funciones, permitiendo su utilización como argumentos dentro de otra función, y siendo definidos ortogonalmente, tanto en su sintaxis como en su semántica, con respecto a los operadores no espaciales. Este lenguaje permite consultas del tipo “que ríos están totalmente en la ciudad de Toronto”:

```
SELECT TUPLE(river_name:river.name)
FROM city IN cities
     River IN rivers
WHERE city.name =’Toronto’ and river.route inside
   city.surface
```

También como extensión a SQL podemos encontrar el lenguaje PSQL[50], donde se incluye tanto una interfaz visual como en modo comando sobre la que realizar las consultas alfanuméricas. Una de sus características es la definición de los elementos espaciales de forma independiente del modelo relacional, de manera que su representación interna puede ser distinta según el dominio y/o la base de datos sobre el que se implemente. Además, en la definición de una relación se han de incluir especificaciones, como la localización, la granularidad y

la visualización, que son necesarios para la posterior visualización. En el lenguaje de consulta introduce una nueva cláusula, ON, en la sentencia SELECT, que permitirá indicar la imagen sobre la que se sitúa la consulta, así como el resto de los operadores definidos sobre la cláusula WHERE, con la capacidad de poder emplear el resultado de uno como operando de otro. Un ejemplo típico podría ser obtener todas la ciudades que se encuentran en un área de {4±4, 11±9} con una población mayor de 450.000, empleando PSQL:

```
SELECT city, state, population, location
FROM cities
ON  us-map
WHERE location WITHIN WINDOW (4±4, 11±9)
   AND population>450000
```

Esta información se presentaría en dos ventanas, una con la información alfanumérica y otra con la visualización de la información gráfica.

El lenguaje OOGQL [66] se define como una extensión al lenguaje OQL, añadiendo tipos espaciales, como una sencilla abstracción geométrica y una jerarquía de clases, y una serie de funciones, métodos y predicados definibles por el usuario. El lenguaje permite la recuperación de objetos y conjunto de objetos junto a tuplas y conjuntos de tuplas. Su sintaxis es del tipo *select-from-where*, donde las operaciones tienen una notación funcional y por tanto similar a los anteriores. En cambio, no incluye características para la visualización, como las comentadas anteriormente, dejando éste desarrollo para otra extensión, definida dentro de la interfaz de usuario, en la que se integraría este lenguaje de consulta. La consulta OOGQL, para obtener aquellos ferrocarriles que atraviesa Alemania sería:

```
SELECT r
FROM  r IN  railroad, s IN state
WHERE s.name=”Germany” AND (r IN s OR r CROSS s)
```

El lenguaje de consulta de DEDALE[25], intenta ocultar al usuario la complejidad del modelo de datos con restricciones subyacente. En particular, trata de ocultar las reglas relativas a las restricciones de tipado, y que las estructuras anidadas sean transparentes para el usuario final. En cambio, si implementa de una forma directa el álgebra definida por DEDALE, permitiendo el diseño de optimizadores de consultas. Su sintaxis al igual que en el caso anterior es del tipo *select-from-where*, donde se introducen un tipo de operaciones denominadas macro-operaciones que simplifican el proceso de manipulación de datos. No incluye dentro del lenguaje características relativas a la visualización, sino que implementa en la arquitectura una interfaz de usuario que permite realizar consultas que emplean rectángulos o puntos, así como para la visualización de los resultados. Para obtener que parcelas contienen el punto (a,b) con este lenguaje:

```
SELECT o.name, o.geo
FROM o IN GO
WHERE RESTRICT? o.geo WITH (x1=a ∧x2=b)
```

Donde GO mantiene toda la información referente a la ocupación del terreno. Realiza un test sobre los objetos en GO, tras realizar su proyección espacial, y comprobar aquellos que son coincidentes. Como podemos observar el lenguaje es menos intuitivo que los precedentes, conforme cabía esperar por la propia complejidad del modelo en el que está basado.

GraphDB [30] presenta algunas diferencias con respecto a los anteriores por el tipo de información, topológica en la mayoría de las ocasiones, que trata de recuperar. Para ello se han definido una serie de herramientas entre las que podemos encontrar:

- *DERIVE*, con una función similar a un select-from-where pero con una semántica extendida para grafos. Incluye funcionalidades de selección, join, proyección y aplicación de función;
- La operación *REWRITE* que permite la manipulación de secuencias heterogéneas, reemplazando objetos o secuencias de ellos; en especial, su uso es similar a una sentencia case que permite especificar comportamientos diferentes para diferentes tipos de objetos.
- La operación *UNION* que permite la generalización dinámica, es decir, transformar colecciones de objetos heterogéneas en un único objeto, visto como un supertipo de los otros;
- Una serie de operaciones sobre grafos que permiten por ejemplo obtener el camino más corto. Para ello, tiene en cuenta la diferencia entre los nodos y los bordes implicados en los diferentes caminos posibles. Por este motivo, tiene en cuenta el tipo de camino que identifica el grafo que será empleado en la consulta, la clase de bordes implicados asignándoles un peso que permita la ponderación de los caminos alternativos, así como una función de estimación de la distancia de cada nodo de origen al nodo destino.

Una consulta para obtener el camino más corto de la salida 16 a la 252, evitando un área cubierta por la niebla y representada mediante un colección de polígonos recogidas en una relación denominada *niebla*, mediante este lenguaje sería:

```
subgraph [seccion where not seccion.ruta intersects fog)];
salida(nr=16) salida (nr=252) shortest path [seccion+,
fun(s:seccion)
length(s.ruta)/s.max_velocidad]
```

En el primer paso de la consulta se recuperan solamente aquellos bordes que están libres de la niebla y en el segundo paso se calcularía el camino más corto. La sintaxis del lenguaje se aleja notablemente del resto de los lenguajes presentados anteriormente, en parte motivado por tratar de recuperar en este caso un tipo de información que los otros no tratan.

De este análisis se desprende que todos los lenguajes utilizan una notación del tipo selec-from-where, a pesar de que algunos de ellos están basados en un modelo objetual. Especialmente se aprecia, la carencia en casi todas las propuestas de tratamiento de la representación gráfica mediante el lenguaje(Tabla

10), pese a la fuerte componente visual que tiene este tipo de información. En la mayoría de los casos se delega en la interfaz de usuario esa funcionalidad.

Tabla 10 Evalución según los criterios de Egenhofer [13]

Requisito	**SpatialSQL**	**QL/G**	**PSQL**	**OOGQL**	**DEDALE**	**GraphDB**
Tipos de datos espaciales	*	*	*	*	*	*
Presentación gráfica	*	*	*			
Combinación resultados	*	*	*	*	*	*
Contexto	*	*	*			
Examen de contenido	*					
Selección por puntero	*					
Manipulación visualización	*		*			
Leyenda	*					
Etiqueta	*					
Escala de visualización	*		*			
Área de interés	*		*			

3.4 Métodos de indexación

La implementación de una base de datos espacial debe ser tal que permita el almacenamiento y consulta, de forma eficiente, tanto de información alfanumérica como espacial. El problema deriva de la complejidad propia del dominio:

- complejidad de las estructuras que han de registrar la información, donde el modelo relacional clásico se muestra deficiente, implicando un mayor coste de procesamiento;
- el dinamismo que suele tener este tipo de información, que conlleva la necesidad de estructuras de datos para realizar de forma eficiente tanto actualizaciones, como borrados e inserciones;
- la necesidad de almacenamiento secundario, por el volumen de información que suele gestionar las bases de datos espaciales;
- la carencia una álgebra estándar que permita definir tanto las operaciones, con independencia de la aplicación concreta con la que esté tratando y sin los problemas que tienen la mayoría de las álgebras actuales, que presentan inconsistencias en el cierre de las operaciones como la intersección;
- además, la inexistencia un lenguaje de consulta estándar.

Los métodos tradicionales de indexación, no son capaces de tratar información de esta complejidad, en la que ni siquiera existen operaciones como

la igualdad entre dos objetos queda definida de forma estándar, o donde criterios topológicos, como la vecindad al intentar obtener las regiones colindantes a una dada. Es indudable, además, que el coste de procesamiento aumenta de forma insostenible, si cada una de las operaciones geométricas que se realizan sobre la información se aplicará sobre todos los objetos geométricos que estén almacenados, por la propia complejidad de los algoritmos de computación gráfica. Este hecho pone de manifiesto la necesidad de definir *métodos de indexación espacial*, también conocidos como *multidimensionales*, basados en propiedades espaciales, normalmente no almacenadas en la base de datos, que permitan ordenar esos datos de forma eficiente para facilitar su consulta, limitándola a aquellos objetos que podrían llegar a cumplir los criterios de búsqueda.

Generalmente el criterio que se suele aplicar, dado que la información espacial suele almacenarse junto con información alfanumérica, es desarrollar motores de búsqueda especializados para información espacial, empleando los motores tradicionales para el resto de tipos dentro de la base de datos, uniendo posteriormente sus resultados. Una consulta típica podría ser obtener todas las carreteras que atraviesan la provincia de Albacete. Para ello, sería necesario en primer lugar, obtener el polígono que delimita la provincia de Albacete, empleando el enlace entre la información alfanumérica y espacial, y posteriormente realizar la intersección espacial entre las carreteras y el polígono obtenido previamente. Gaede[21]definió un conjunto de propiedades que deben cumplir los métodos de acceso multidimensionales:

- el dinamismo frente a inserciones y borrados, de forma que puedan atender los cambios con independencia del orden en que estos se realicen;
- independencia de los datos de entrada y de la secuencia de inserción, de especial importancia ante datos heterogéneamente distribuidos;
- gestión de almacenamiento en memoria secundaria y terciaria, que permita asimilar el volumen de información;
- variedad de las operaciones soportadas, sin dejar de dar soporte a una operación, por ejemplo borrado, frente a otra como podría ser la inserción;
- escalabilidad ante el crecimiento de la base de datos;
- simplicidad del algoritmo, que se traduce en una sencilla, y generalmente más eficiente, implementación;
- la eficiencia temporal y de almacenamiento, que permita la búsqueda de una forma rápida, manteniendo el espacio de índices dentro de unos límites aceptables.

Podemos distinguir dos tipos de métodos de accesos multidimensionales [21]: métodos de acceso a punto (*PAM, point access method*) y métodos de acceso espaciales (*SAM, spatial access method*). Los primeros están orientados a bases de datos que contienen únicamente, o mayoritariamente, puntos como objetos espaciales. Dentro de este grupo se emplean técnicas hashing unidimensionales, de forma que, aunque no se pueda aplicar una relación de

orden total sobre objetos d-dimensionales al introducir esa reducción de dimensión, permita asegurar que objetos próximos en el espacio estén en la misma región o en regiones continuas en la indexación. Ejemplos de este tipo, son *grid file* o *EXCELL*([21]). Además de los anteriores, se encuentran dentro de esta categoría métodos de acceso jerárquicos, tales como quadtree o k-d-b-tree, que aplican un principio de descomposición recursiva de espacio. Este tipo de estructuras de datos presentan una especial habilidad para centrarse sobre determinados subconjuntos de los datos, que son de interés para la consulta a realizar, dado que en cada nivel de la jerarquía nos permite reducir el espacio que se ve afectado por la consulta, mejorando el rendimiento de las operaciones de conjunto. Incluso hay ocasiones en las que su uso se prefiere, frente a otras estructuras de datos de igual rendimiento, por su claridad y sencillez de implementación.

Frente a los anteriores, los métodos de acceso espaciales permiten gestionar objetos con extensión espacial, tales como líneas o polígonos. Dentro de este grupo existen diferentes posibilidades según como se trate la información a indexar. Existen alternativas que emplean los métodos PAM, después de transformar esos objetos con extensión en puntos en un espacio dimensional superior, o su descomposición en objetos más sencillos como rectángulos. Otra alternativa, trata de encuadrar los objetos, descomponiendo el espacio en una estructura jerárquica, permitiendo o no el solapamiento en nodos intermedios, como es el caso de R-tree, R^+-tree o R*-tree.

Además las diferentes las técnicas que mencionadas, tienen en cuenta la paginación en memoria secundaria, uno de los principales problemas con que contaban los primeros algoritmos introducidos, como k-d-tree o BSP-tree [21], que empleaban únicamente la memoria principal, haciéndolos poco apropiados para el volumen de información que suele contener una base de datos espacial.

Actualmente, los dos métodos más difundidos, son R-trees y quadtress, en sus diferentes variantes. Algunas de estas mejoras, tratan de combinar ambos métodos, empleando cada uno de ellos para aquel tipo de datos que ofrece mejor rendimiento.

3.4.1 R-trees

Los R-trees son estructuras jerárquicas de descomposición del espacio en las que se almacena el *mínimo paralelepípedo marco*(*MBB, minimum bounding box*) en lugar del propio objeto que se desea indexar. El MBB de un objeto n-dimensional se define como el mínimo paralelepípedo n-dimensional que puede contener ese objeto. El uso del MBB permite filtrar fácilmente aquella información que no se ve afectada por una consulta.

Una de las características que este tipo algoritmos posee, y que ha potenciado su uso frente a otros métodos de indexación, es su capacidad para poder implementar join espaciales, una de las operaciones que suele tener cualquiera de los modelos de datos presentados anteriormente. Este tipo de operación permitiría, por ejemplo, obtener todas las ciudades que tengan a menos de 50km de distancia un río:

```
Ciudades rios join [dist[center, route)<50]
```

donde se mezclan puntos y líneas en la misma operación.

Un R-tree [27] es un árbol balanceado en altura, similar a un B-tree, con nodos hoja, como punteros a objetos de datos y nodos intermedios. Cada uno de esos nodos intermedios está asociado con un rectángulo que encierra a cada uno de lo rectángulos de los nodos de los niveles inferiores. En la Fig. 11 y Fig. 12, los nodos hojas (R_i) se encargan de mantener un rectángulo que enmarcará a las entidades geométricas, junto con un puntero a dónde están almacenadas, y los nodos de los niveles superiores, A y B en este caso, encerrarán a su vez a los anteriores. Empleando este método se subdivide el espacio en sub-regiones que pueden solaparse entre sí. Así, un objeto, espacialmente está incluido en varias sub-regiones sólo está asociado con un nodo. Esto implica que una consulta podría requerir visitar varios nodos antes de poder asegurar la presencia o ausencia de un determinado nodo.

Fig. 11: Rectángulos organizados para un R-tree

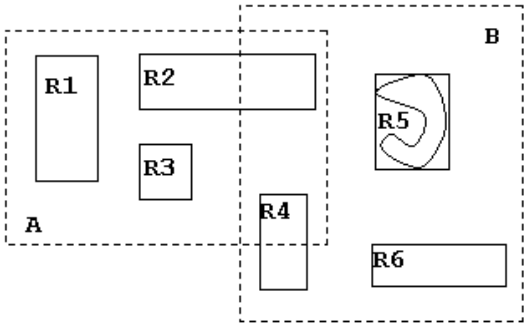
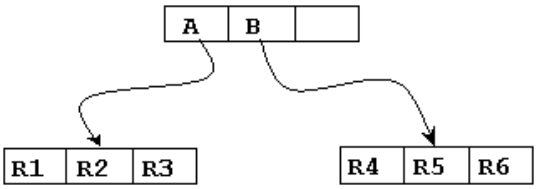


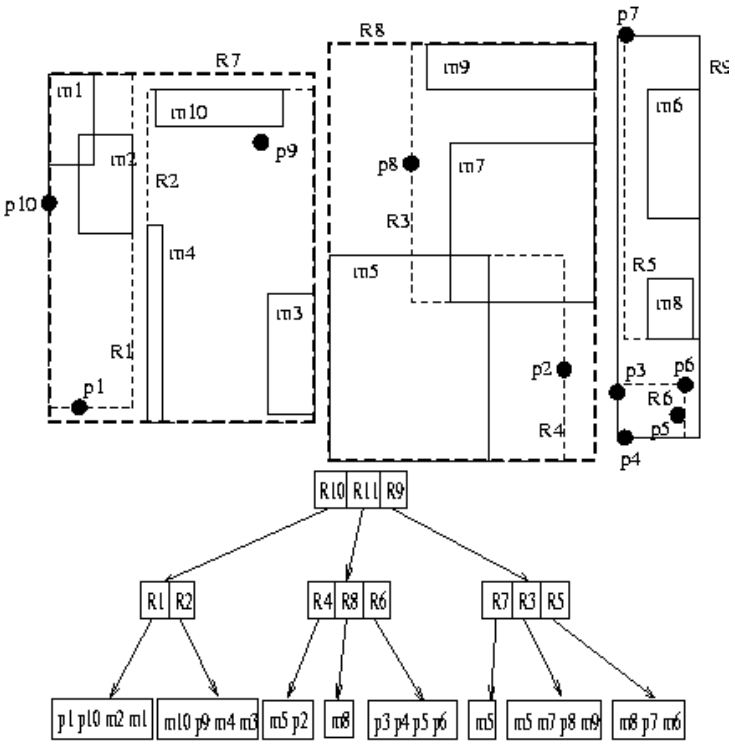
Fig. 12: R-tree de los rectángulos anteriores



Las dos características principales de esta estructura son, por una parte, tener un comportamiento completamente dinámico ante inserciones y borrados, aunque

estos se encuentren entremezclados con búsquedas, y por otro, no requerir una reorganización periódica. Sin embargo, unos de los principales problemas que plantea ante determinadas distribuciones de la información espacial son el solapamiento excesivo de regiones y la gran cantidad de espacios no utilizados en el árbol, aunque puede en parte reducirse su influencia en el rendimiento con el empleo de técnicas de empaquetamiento, poco efectivas sobre inserciones simples.

Fig. 13: Ejemplo de R*- tree: puntos y MBB



Las diferentes debilidades mostradas por el algoritmo R-tree, como la alta sensibilidad en el rendimiento del proceso de búsqueda a la fase de inserción de la información, motivaron la aparición de *R*-tree*[2], se muestra un ejemplo en la Fig. 13. Esta versión introduce nuevas políticas de inserción, denominadas *reinserción forzada*, de forma que una nueva entrada en un nodo que provoca su desbordamiento no provoca que éste sea partido, sino el borrado de *p* entradas ya existentes y su posterior reinserción en el árbol. El objetivo de esta política es una mejora en el rendimiento del árbol, mediante la minimización de la región de solape entre nodos hermanos en el árbol. Además, introduce otra diferencia, con respecto a R-tree, en el algoritmo de partición de los nodos, por como se realiza la selección del eje con respecto al que se realiza la misma, ya que se selecciona aquel que provoque un menor solapamiento entre las proyecciones de los MBB

sobre los ejes. Esta política de partición es introducida debido a que el algoritmo R*-tree, no solamente trata de minimizar el área cubierta por los nodos, sino que tiene también en consideración otros objetivos como:

- minimización del solape entre regiones en el mismo nivel del árbol, reduciendo con ello la probabilidad de seguir múltiples búsquedas en el árbol;
- la minimización del perímetro de las regiones, y
- la maximización de la utilización del almacenamiento.

En general, suele ofrecer una mejora considerable en el rendimiento frente a su predecesor, incluso de hasta el 50%[2].

Otra de las alternativas planteadas, que pretendían evitar los problemas planteado por R-tree, es la conocida como *R^+-tree* [52], basada en una descomposición del espacio en sub-regiones disjuntas, donde cada objeto se encuentra asociado con todos los rectángulos marco con los que intersecta. No existe por tanto solapamiento entre los diferentes rectángulos marco, a excepción de los que se encuentran en las hojas y que contienen los objetos. En Fig. 14 y Fig. 15, se aprecia como se comportaría el algoritmo ante la misma distribución espacial que en el caso anterior. Como resultado se ha creado un nuevo rectángulo, C, en el primer nivel del árbol de forma que se evita el solapamiento entre A y B.

Fig. 14: Rectángulos organizados para un R^+-tree

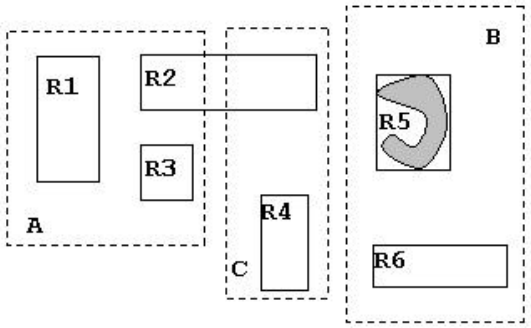
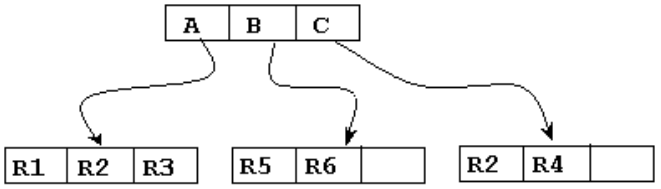


Fig. 15: R^+-tree de los rectángulos anteriores



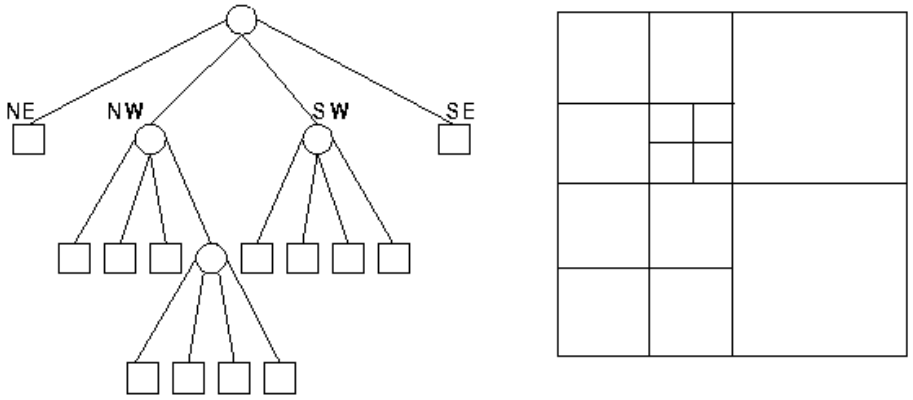
En cambio, se ha de pagar un precio por esa disyunción ya que para poder obtener el área cubierta por un determinado objeto, es necesario recuperar todas las celdas que éste ocupa. Esta situación se encuentra en el rectángulo R2. En el

caso de R-tree había una única ocurrencia en el árbol. Pero ahora se mantienen dos referencias suyas debido a que queda solapado por dos rectángulos, A y C. Esta situación implica igualmente la existencia de diferentes caminos para poder acceder al mismo objeto y, por consiguiente, a un incremento en la altura del árbol, que sin embargo no afecta a los tiempos de recuperación que suelen ser excelentes.

3.4.2 Quadtrees

Otra de las técnicas más difundidas, y a la que se han propuesto diferentes variantes, es la conocida como quadtree [53]. Al igual, que sucedía con R^+-tree, produce una descomposición disjunta del espacio, pero en este caso, el proceso de descomposición, se realiza mediante una subdivisión sucesiva del espacio en cuatro cuadrantes de igual tamaño (depende del algoritmo en concreto), y con una mayor independencia de los datos.

Fig. 16: Un quadtree y la correspondiente subdivisión del espacio



Un quadree se considera un árbol, en el que cada uno de sus nodos corresponde a una extensión espacial cuadrada. En el caso de los nodos interiores del árbol, estos tienen cuatro nodos hijos correspondientes a los cuatro cuadrantes del rectángulo que representa, de ahí deriva su nombre. La Fig. 16 muestra un ejemplo de un quadtree junto, con la correspondiente subdivisión espacial que representa. Se observa como desde el nodo raíz parten 4 nodos hijos, cada uno de ellos etiquetado como NE, NW, SW y SE, según el cuadrante que representan, es decir, NE representa al cuadrante que se encuentra hacía el norte y al este en el rectángulo que subdivide. Se trata, por tanto, de un proceso recursivo de subdivisión, en cuatro cuadrantes de acuerdo a los datos de entrada.

Los diferentes variantes de quadtrees se diferencian básicamente por tres razones:

- el tipo de datos que emplean para realizar la representación;

- el principio que guía el proceso de descomposición;
- la resolución, que puede o no ser variable.

Tal y como se menciona, una de las características de los quadtrees es el tipo de datos sobre el que se emplea, abarcando tanto a puntos, como líneas, regiones o volúmenes. Esta diversidad ha introducido variantes (como region quadtree, PM quadtree, PR quadtree, etc., tratados más adelante) que se especializan sobre el tipo de datos, a fin de conseguir un mejor rendimiento, tanto en el proceso de búsqueda como de almacenamiento.

El segundo punto, hace referencia a la posibilidad de emplear los datos de entrada en el proceso de subdivisión. Esta forma de utilizarlos se traduce en la posibilidad de generar cuadrantes de diferente o igual tamaño, adaptándose o no a la distribución espacial de la información.

La resolución de la descomposición, significa el número de veces que puede llegar a aplicarse el proceso de descomposición. Es posible que sea fijado de antemano, o bien, tener en cuenta las propiedades de los datos de entrada, depende de la aproximación.

De los diversos datos que puede representar un quadtree, uno de los más representativos son las regiones de datos binarios bidimensionales. Más concretamente, el conocido como *region quadtree* [53] está basado en una subdivisión sucesiva de la región, en cuadrantes de igual tamaño, sobre un array. El proceso que sigue es comprobar si cada una de las celdas de ese array no consiste enteramente de 1s y 0s, por contener celdas que son parcialmente cubiertas por la región, en cuyo caso cada una de ellas es nuevamente subdividida en subcuadrantes hasta que el array contengan únicamente 0s y 1s. Cada bloque es enteramente contenido en la región o enteramente disjunto. La región quadtree se caracteriza por ser una estructura de datos de resolución variable. En la Fig. 17, se aprecia como sería el array binario de una región y los correspondientes bloques máximos que permiten representarla. El quadtree resultante se recoge en la Fig. 18. En él las regiones sombreadas aparecen como cuadrados sombreados en el árbol.

Fig. 17: Región muestra, su representación binaria, sus bloques máximos

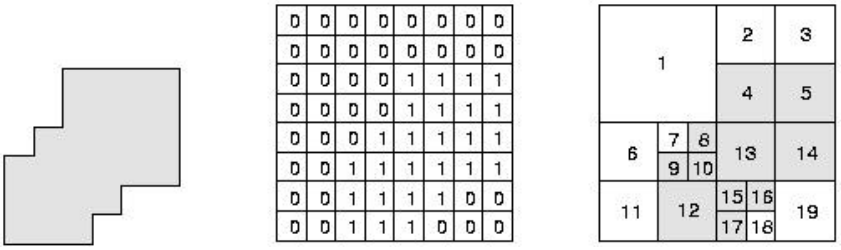
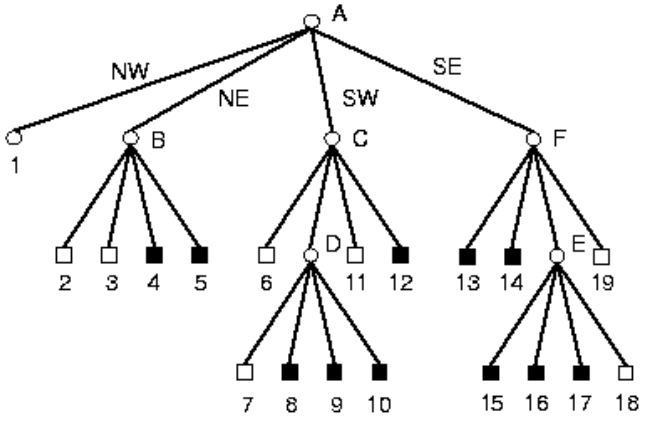


Fig. 18: Quadtree correspondiente a la región anterior



La representación de los puntos de datos multidimensionales ha presentado otra alternativa, con respecto a la anterior, conocida como *PR quadtree* [53], basada en una descomposición regular. Éste tiene una organización similar a la region quadtree, la diferencia reside en los nodos hojas, además de poder ser vacíos, pueden contener puntos de datos junto con sus coordenadas, es decir, contiene directamente la información que indexa. La Fig. 11 recoge un ejemplo de esta estructura, donde cada cuadrante contiene como máximo un punto. Esta estructura presenta diversos inconvenientes, como la cantidad de testeos que requiere, con el consiguiente coste computacional, sobre cada una de las *k* claves de un quadtree k-dimensional. Además de la elevada cantidad de nodos vacíos que puede haber en las hojas. Estos problemas motivaron la aparición de los k-d-trees [53] y sus variantes en un intento de mejorar ese rendimiento.

Esta misma estructura puede aplicarse para la representación de regiones, que consisten únicamente de una colección de polígonos y de líneas. El resultado es una familia de representaciones conocida como *PM quadtree* [53]. En ella, las regiones quedan especificadas empleando sus bordes, en contraste con la region quadtree que emplea una descripción basada en el interior de una región. En esta aproximación, el mapa de polígonos es sucesivamente subdividividido hasta que no contenga más que una línea, salvo que abarque el punto final de varias líneas, en cuyo caso permite la presencia de aquellas líneas que intersectan con el mismo, pero no permite la presencia de más de un punto final.

Fig. 19: Point quadtree

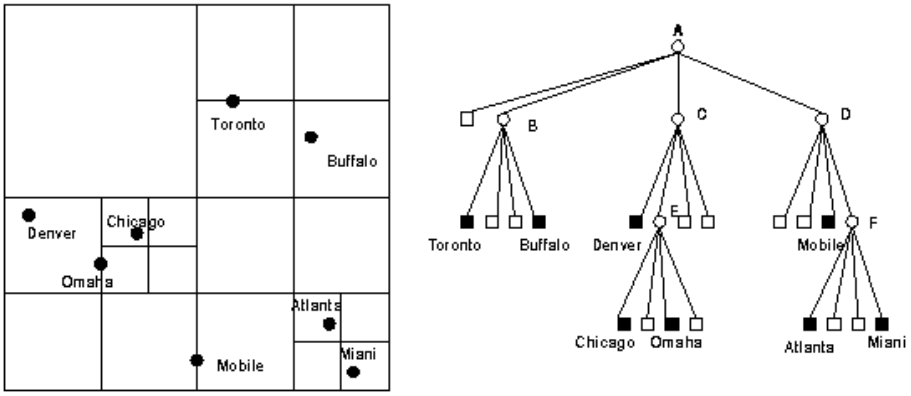
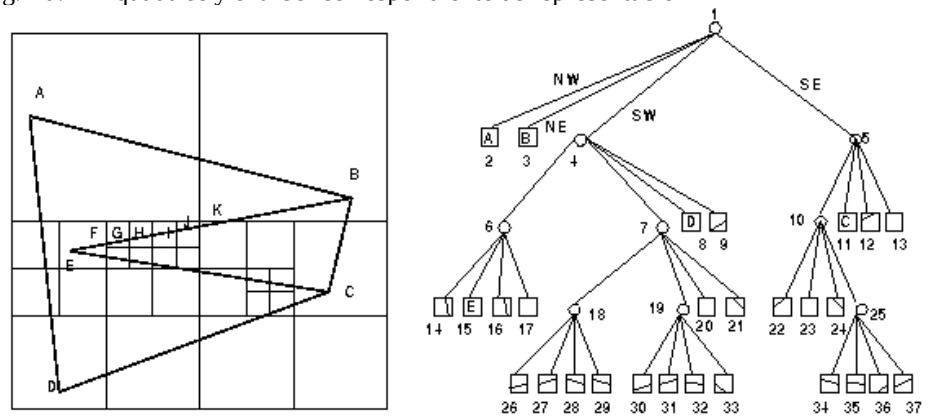


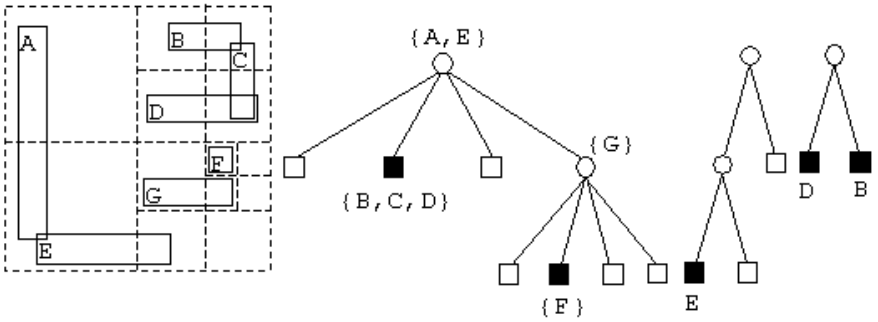
Fig. 20: PM quadtree y el árbol correspondiente de representación



Otro tipo de objetos que podemos encontrar al realizar la indexación con quadtrees son los rectángulos. Éstos pueden emplearse para delimitar otros objetos, al igual que los R-trees, donde la extensión del rectángulos se tiene en cuenta para realizar la indexación, no su centroide. En la aproximación basada en quadtrees, conocida como MX-CIF quadtree [53], cada nodo del árbol se encuentra asociado al bloque más pequeño que puede contener, cesando la subdivisión en el momento en que un nodo no contiene ningún bloque.

En la Fig. 21 se muestra el quadtree que generaría este algoritmo para esa colección de rectángulos. El rectángulo F ocupa un bloque completo por lo que está asociado con el bloque padre. Asimismo, los rectángulos pueden estar asociados tanto con nodos no terminales, como con nodos hojas. El motivo básico es que este algoritmo trata de diferenciar los rectángulos más que encerrarlos en un bloque.

Fig. 21: Quadtree generado por MF-CIF



4 Bases de Datos Espacio-Temporales

Durante las últimas décadas, se ha desarrollado la investigación en el campo de las bases de datos temporales y espaciales desde perspectivas separadas. Es evidente que cuando se trata de integrar tiempo y espacio, se trata con geometrías que cambian a lo largo del tiempo.

Así, una base de datos espacial registra información de puntos, líneas y regiones, sobre su localización y su geometría, e incluso en algunos casos, como en el modelo topológico (ver apartado 3.1.3) podía reflejar las relaciones topológicas sobre los diferentes objetos, permitiendo realizar consultas sobre caminos más cortos, alternativas, etc. Sin embargo, ninguna de las anteriores aproximaciones, en el área de las bases de datos espaciales, se contempla la posibilidad de que esos objetos cambien su posición a lo largo del tiempo. Dado un sistema de control del tráfico, los vehículos podrían ser considerados como puntos ya que solo nos interesa su posición, se mueven continuamente en el tiempo. No interesa registrar únicamente su posición en un momento determinado.

Tampoco se presentan soluciones ante situaciones en las que la geometría de los objetos cambie ante determinados eventos. En un sistema de gestión catastral, se puede dar el caso de una parcela que ha cambiado su forma porque el propietario ha comprado la parcela colindante. ¿Cómo se puede contemplar este cambio?, se crea un nuevo objeto cuya geometría viene determinada por la unión de las dos anteriores, pero ¿cómo se puede establecer una relación entre ese nuevo objeto y los dos anteriores?, o bien, ¿consideramos que un objeto se ha destruido y el otro se ha modificado para contemplar ese cambio haciendo necesario registrar de alguna forma esa evolución?

Se desprende de los ejemplos anteriores, la existencia de necesidades reales en las que es preciso contemplar la gestión del espacio y del tiempo con un mismo enfoque integrando las características de ambos como soporte para la gestión de este tipo de información. Es necesario un conjunto de herramientas que permitan un uso y gestión eficiente de la llamada información espacio-temporal.

La introducción de esa información espacio-temporal, no se limita únicamente a incluir unos atributos temporales que permitan registrar ese cambio, sino que necesita la definición de nuevos modelos de datos que permitan recoger la semántica de ese cambio, junto con lenguajes y técnicas de indexación que permitan la recuperación y manipulación de una forma eficiente, de este tipo de información. Estas necesidades ponen de manifiesto las carencias no solo de los DBMS tradicionales, sino también de las bases de datos espaciales para tratar con este tipo de información.

4.1 Modelos espacio-temporales

Los modelos espacio-temporales se caracterizan por tratar objetos cuya geometría cambia a lo largo del tiempo, es decir, tienen capacidad de gestionar geometrías en cambio continuo. Existen diferentes conceptos que deben tenerse en cuenta a la hora de establecer una posible clasificación entre los diferentes modelos.

En primer lugar, en el dominio espacial, la elección de la representación para el almacenamiento de datos espaciales. En la actualidad existen dos aproximaciones. La primera de ellas representación ráster en la que el espacio es fraccionado en una cuadrícula de forma que cada celda puede ser direccionada por su posición. La ventaja que ofrece este tipo de representación es la sencillez de cualquiera de las operaciones geométricas, al tratar de determinar la intersección de dos objetos en el espacio, su unión, etc. Sin embargo, su utilización plantea problemas de eficiencia tanto temporal como de almacenamiento, puesto que se necesita una gran cantidad de tiempo para procesar grandes volúmenes de información, sobre todo al trabajar con objetos de cierta resolución. La segunda, emplea la representación vectorial, según la cual el objeto espacial se representa a partir de una serie de puntos. Esta representación permite paliar los problemas antes comentados tanto respecto a la eficiencia temporal como de almacenamiento. Sin embargo, la complejidad de implementación de cualquiera de las operaciones que se defina es mucho mayor.

Otro aspecto a considerar se centra en el dominio temporal. Se trata en este caso de considerar como se introduce esa información relativa al tiempo dentro del modelo. Tal y como se introdujo en las bases de datos temporales (apartado 2.1) existen dos conceptos básicos que son tiempo de validez y tiempo de transacción. La forma en que estos conceptos son recogidos, o no, dentro del modelo permitirá dar diferentes semánticas a los tipos definidos, y que, a su vez, permitirá establecer diferencias entre los distintos modelos que se comentaran seguidamente.

Otro aspecto a tener en cuenta es si el modelo bajo estudio permite capturar las nociones de *movimiento* y de *cambio*, cuyos significados se muestra en los ejemplos del sistema de control de tráfico y catastral. Se trata de reflejar, mediante el movimiento, alteraciones en la posición en el espacio a lo largo del tiempo. Por el contrario, el concepto de cambio se ocupa de recoger como el objeto como sufre una transformación en su extensión. Ambas nociones se ven afectadas por cómo se produce esa evolución, es decir, si ésta se contempla de forma discreta, con lo que podría emplearse el espacio de los enteros para realizar su representación, o bien, es continua con lo que deberían emplearse los reales o racionales para recogerla.

Relacionado con los conceptos de cambio y movimiento, otro aspecto que influye en la clasificación se refiere a la capacidad del modelo para realizar

consultas sobre esa evolución, es decir, la posibilidad de contemplar la semántica que conlleva la alteración de la posición y de la extensión.

Otro aspecto que se contempla en la siguiente clasificación es el modelo de base de datos sobre el que se realiza la extensión espacio-temporal, es decir, si se emplea en su definición el modelo relacional u objetual. Ambos modelos han sufrido diferentes críticas que han de ser valoradas en el momento de decidir cual de los dos emplear.

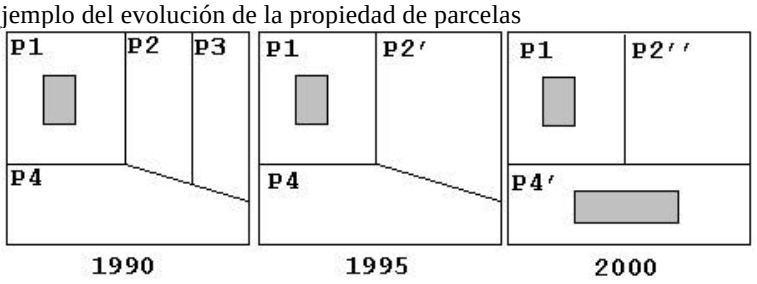
- El modelo objetual se considera todavía carente de la madurez necesaria, debido a que no existe todavía una base formal lo suficientemente fuerte. Esto se acentúa al ser aplicado en un dominio como el espaciotemporal en el que no existe todavía ningún acuerdo ni en su semántica ni en su sintaxis. En cambio, ofrecen una mayor riqueza expresiva.
- En el caso del modelo relacional, se han encontrado diversas deficiencias, que dificultan su empleo en este campo, para incorporar conceptos espaciales como la visualización gráfica y su especificación, así como la carencia de potencia para soportar consultas cualitativas, de conocimiento o sobre metadatos.

Finalmente, la programación con restricciones (ver apartado 3.1.2) se extiende tanto a la representación espacial, como temporal, dada la posibilidad que ofrecen de representar información infinita, en el dominio continuo.

4.1.1 Ejemplos de estudio

Para la comparación de los distintos modelos se ha introducido un ejemplo que nos permita evaluar las características de cada uno frente a un mismo problema. Una de las áreas en la actualidad, donde se emplea de forma más prolífica esta información espacial y temporal, es dentro del campo de los Sistemas de Información Geográfica, donde es posible encontrar múltiples casos en los que la introducción de modelos espacio-temporales facilitan la gestión.

Fig. 22: Ejemplo del evolución de la propiedad de parcelas



Recurriendo al ejemplo anterior del catastro, se plantea: cómo contemplar el hecho de que una parcela pueda cambiar de propietario o cómo puede sufrir una ampliación por la adquisición de la parcela contigua es un claro ejemplo de

cambio en la información espacial a lo largo del tiempo. En la Fig. 22 se representa la evolución de la propiedad de las parcelas. Una posible cronología de esa evolución sería:

- Año 1990: se registra información acerca de la parcela P1 propiedad de Juan, la parcela P2 propiedad de José, la parcela P3 propiedad de Pedro y P4 propiedad del ayuntamiento.
- Año 1995: José compra la parcela P3 a Pedro, pasando su propiedad a ser P2´.
- Año 1997: el ayuntamiento anuncia la construcción de un nuevo edificio y la ampliación del terreno de su propiedad.
- Año 2000: el ayuntamiento ejecuta sus planes.

Otro ejemplo, relacionado con el concepto de movimiento, son los sistemas de control de flotas, fuertemente relacionado con el rango de la logística. En ellos, una de las necesidades básicas es conocer la posición exacta de un transporte en cada instante. Esta información, junto con otra información espacial, como la red de carreteras, podría emplearse para establecer la ruta óptima ante la demanda de un servicio. Se enfrenta en este caso con un objeto espacial que se encuentra en movimiento, altera su posición en cada instante, al que se conoce generalmente en la bibliografía como *moving object*.

4.1.2 Modelo snapshot

Langran [38] presenta una de las aproximaciones más simples en las que la dimensión temporal está basada en un modelo discreto y lineal del tiempo, empleando una estructura ráster para registrar la información espacial. El dominio temporal se incorpora en el modelo mediante el tiempo de validez únicamente, pero con diferentes con diferentes granularidades.

La información espacial se incorpora mediante un conjunto de capas, en las que cada una de ellas representa una colección de información cuya validez es homogénea, es decir, toda la información que se encuentra recogida en una capa es valida durante el mismo intervalo temporal. La información temporal se incorpora como una marca temporal sobre cada una de esas capas, es decir, se considera un atributo de la información espacial.

La Fig. 22 recoge un claro ejemplo de este modelo. Cada una de las imágenes para los instantes 1990, 1995 y 2000 representarían una capa de información espacial, en el que las parcelas P1, ..., P4 forman unidades homogéneas. El intervalo de tiempo, como podemos observar es arbitrario, es decir, se recoge la información en el momento en que se produce un cambio, no de acuerdo a unos intervalos determinados.

Se presenta como el modelo más sencillo para representar la información espacio-temporal, pero con serias limitaciones en cuanto a las consultas

relacionadas con la evolución temporal. No olvidemos que no existe ninguna relación entre las diferentes capas; en su lugar, cada capa recoge la información valida en un momento determinado. Por lo que, para detectar si las dos capas han evolucionado exige una comparación exhaustiva de las mismas, impidiendo que el modelo refleje claramente esa evolución.

Además, ofrece serias deficiencias en cuanto al rendimiento en el almacenamiento. Para cada uno de los cambios que sufre la información se ha de recoger toda la información, por la propia definición del modelo, tanto aquella que ha sufrido una evolución como la que no, con la consiguiente duplicación de la información.

Otra de las desventajas que ofrece este modelo es que solo ofrece la posibilidad de registrar el tiempo de validez, no el tiempo de transacción, con lo que no es posible recuperar la información a un estado previo teniendo en cuenta los instantes de modificación del usuario.

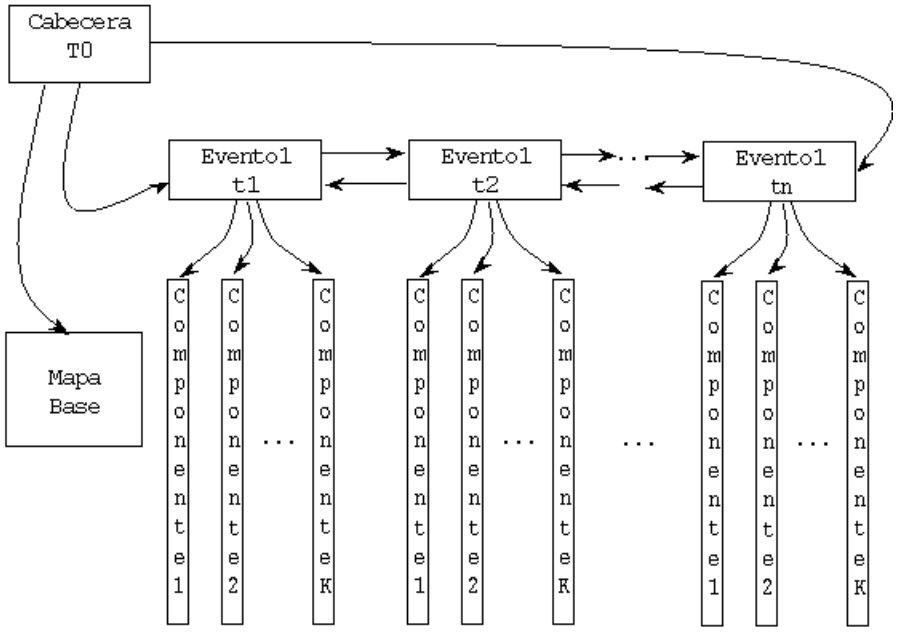
En cuanto a la posibilidad de registrar el movimiento de los objetos, hace prácticamente inviable su utilización en este tipo de situaciones debido a la gran cantidad de snapshot que tendría que almacenar.

4.1.3 Modelo de datos orientado a eventos

Un modelo de datos orientado se caracteriza por representar los cambios más que las entidades, de manera que los diferentes estados por los que ha pasado una entidad pueden ser recuperados rastreando los cambios hasta alcanzar el instante deseado. Concretamente se aproxima a las bases de datos rollback (apartado 2.1) dado que se utiliza el tiempo de transacción para marcar la entrada de la entidad espacial en la base de datos, de forma que cuando ésta sufre una modificación una nueva versión es introducida. En este caso la información actualmente accesible en la base de datos, representaría el estado actual, la información que actualmente es válida, por lo que para recuperar el estado anterior se realizaría un proceso de rollback, que nos permita recuperar el estado de la información que fue actual en un momento anterior.

Al contrario que en el caso anterior, no se registra toda la información nuevamente sino únicamente aquella entidad que ha sufrido los cambios. Esto permite solucionar el problema de la duplicación de la información. Además permite de esta forma reflejar expresamente la evolución puesto que los cambios son almacenados como diferencias con respecto a la versión anterior.

Fig. 23: ESTDM



Event Oriented Spatio-Temporal Data Model (ESTDM) [48] es un ejemplo de modelo de datos orientado a eventos que emplea una estructura ráster para registrar la información espacial. En este modelo, se presentan dos alternativas para registrar los cambios de la información espacial, una de ellas es almacenar únicamente los cambios, con relación a un estado previo, o bien, si los cambios son muy elevados se registraría una instantánea del mapa completo. Un mapa inicial muestra el estado de partida de la información(ver ejemplo de la Fig. 23). Se incorpora, por cada cambio un evento, al que se asocia tanto una marca temporal como diferentes punteros a cada una de las componentes espaciales que reflejan la nueva versión. Estos eventos se asocian mediante una lista de eventos, de manera que quedan ordenados según la secuencia temporal en la que sucedieron, permitiendo así consultar la información de acuerdo a esa secuencia de cambios.

Para el ejemplo de la Fig. 22, se incorporaría una imagen inicial referida al año 1990, y para cada uno de los cambios que van ocurriendo en los diferentes años, se incorporaría un nuevo evento en el que se registraría el año (1995, 2000) y los punteros a los nuevos componentes que han aparecido. Los estados de estas parcelas sólo pueden ser obtenidos mediante el seguimiento de estos cambios.

Mediante este modelo se pueden realizar tanto consultas temporales como espaciales, pero sufre carencias para seguir la historia de las diferentes versiones de las entidades y de sus localizaciones. La ventaja que ofrece la utilización de este modelo es que es apropiado para consultas sobre la evolución temporal de un área, dado que permite las consultas espaciales, podemos seguir los eventos para

estudiar los cambios producidos a lo largo del tiempo, soportando al igual que antes diferentes granularidades. Además, soluciona el problema que mostraba el modelo anterior sobre la redundancia de la información, ya que se reduce al mínimo la duplicación de información, con las implicaciones que ello conlleva en cuanto a la consistencia. Dos objetos que comparten una historia común son representados mediante un único conjunto de objetos espaciales, evitando por tanto que se registre información errónea. En cambio, ofrece como desventaja el no permitir contemplar la noción de movimiento.

Roddick [49] presenta un modelo más complejo. En este caso, una versión de esquema proporciona el medio para registrar la evolución de la información espacial a través de las diferentes modificaciones del esquema de la base de datos. Los esquemas están etiquetados y referenciados con una región espacio-temporal, por defecto asignada a todo tiempo y espacio, junto con una etiqueta suministrada por el usuario. Si las etiquetas referidas a tiempo y espacio no son introducidas, el modelo se degrada a un modelo de esquema estático.

Para su definición introduce el concepto de *esquema de relación completo* como el mínimo esquema capaz de mantener todos los datos sin que se produzca pérdida. Definiendo C como un esquema de relación completo de un esquema de relación R, formado a partir de la unión mínima de todos los atributos que son definidos explícitamente mediante la expansión espacio-temporal de la relación R. Cada uno de los atributos definidos en C es lo suficientemente general como para mantener todos los datos almacenados bajo la versión de R. La definición de la clave primaria implícita de C se obtiene como el máximo conjunto de atributos clave obtenido en esa expansión espacio-temporal. De esta manera, las diferentes versiones de esquema pueden ser contempladas como vistas de C.

Para contemplar el ejemplo de la Fig. 22 se introducen diferentes esquemas de la base de datos, cada uno de ellos como una versión diferente, con los correspondientes tiempos de validez. Permite esta aproximación, la realización de consultas en un rango espacio-tiempo, generando como salida un nuevo esquema en el que los datos pueden consultarse sin pérdida de información. Es decir, permite estudiar la evolución de esa información.

Por el contrario, este modelo no ofrece soluciones a problemas similares a la gestión de flotas de transporte, dado que el número de versiones del esquema necesarias es demasiado elevado.

4.1.4 Modelo basado en time-stamping

Lorentos et al. [43] presentan un modelo en el que el tiempo es considerado como absoluto discreto y lineal, en el que se permiten múltiples granularidades. En él, cada objeto espacial considerado se le asigna un tiempo de vida marcado por los dos extremos de un intervalo cerrado. En este modelo los tipos espaciales, *point*,

line y *region* (ver apartado 3.1.1), tienen asociada una información temporal que permite registrar el tiempo de validez de esa información.

Así, en la Fig. 24, vemos como evoluciona con respecto al tiempo el objeto espacial Morpheas, desde que es un "nacimiento" hasta que se convierte en un "lago" y como ello se recoge en la relación S. En este caso que se trata con el cambio del mismo objeto mediante la identificación que proporciona el nombre.

Para el caso de de la Fig. 22, quedaría registrado en la relación como cada una de las parcelas evoluciona a un nuevo estado. Podríamos encontrar una relación similar a la Tabla 11, en la que se recogen los atributos de identificación de la parcela, el intervalo de validez y la información sobre la geometría de los diferentes objetos.

Tabla 11 Evolución de las parcelas

Id	**T**	**G**
P1	[1990, 2000]	g1
P2	[1990, 1994]	g2
P3	[1990, 1994]	g3
P4	[1990, 1999]	g4
P2	[1995, 1999]	g5
P2	[2000, 2001]	g6
P4	[2000, 2001]	g7

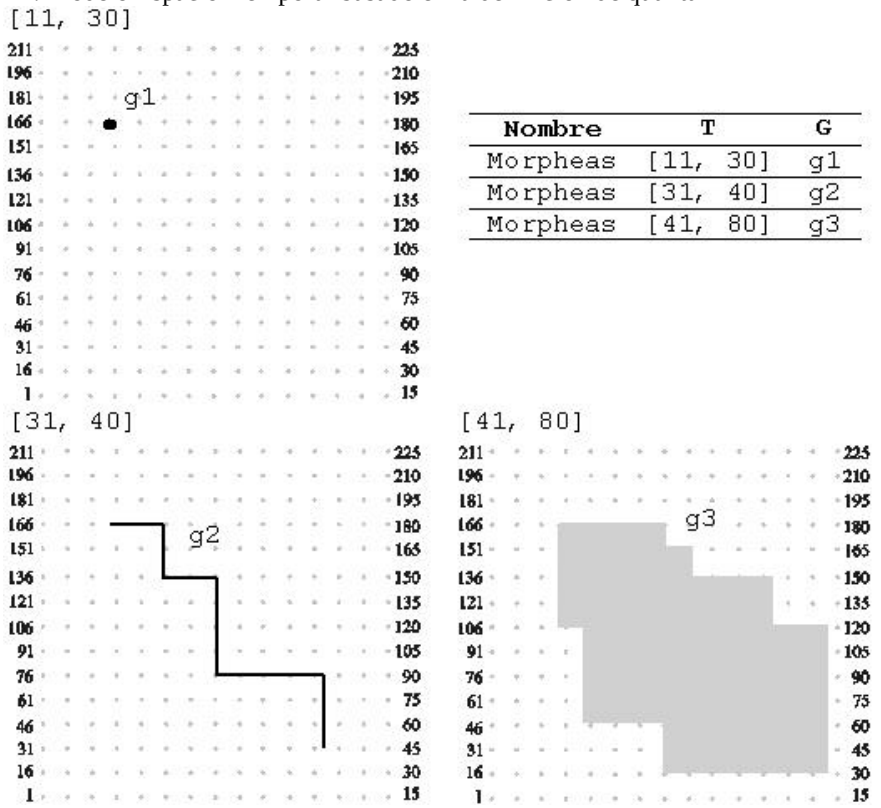
Tal y como podemos apreciar los cambios en los objetos son recogidos mediante la incorporación de aquella información que sufre el cambio, no de todos aquellos objetos entre los que puede existir una relación de vecindad, dado que cada uno de ellos es considerado de forma independiente del resto.

No se hace una especificación explicita en el modelo de si el intervalo de tiempo que se registra es el tiempo de validez o de transacción, por lo que su aplicación podría depender de las necesidades de la aplicación para la que fuera emplearse. Asimismo, tampoco se hace ninguna restricción en cuanto a la granularidad temporal que podría aplicarse. Para ello, hace uso de las definiciones que fueron introducidas a nivel de intervalo temporal en IXSQL[39], tal y como se comentó anteriormente (apartado 2.2).

Las funciones y predicados que se han definido permiten la manipulación conjunta de información espacial y temporal, junto con un conjunto de operaciones del álgebra relacional, tales como *Union*, *Except*, *Project*, etc. Además de ellas, dos operaciones especiales conocidas como *Fold* y *Unfold*, anteriormente comentadas, tanto para las bases de datos temporales (apartado 2.2) como espaciales(apartado 3.1.1), son las que proporcionan la potencia a este

modelo, ya que permiten definir de una forma sencilla el resto de las operaciones del álgebra.

Fig. 24: Modelo Espacio-Temporal basado en la definición de quanta



Nombre	T	G
Morpheas	[11, 30]	g1
Morpheas	[31, 40]	g2
Morpheas	[41, 80]	g3

La ventaja que ofrece este modelo es que las operaciones pueden ser aplicadas uniformemente tanto a datos espaciales, como temporales o espacio-temporales. En cambio, no recoge de una forma explícita la evolución respecto al movimiento de los objetos. Cada alteración sufrida por un objeto, generaría una nueva tupla en la que la relación con la versión anterior ha de ser reflejada por el usuario mediante la introducción de algún otro atributo, lo que la hace poco adecuada para este tipo de aplicaciones.

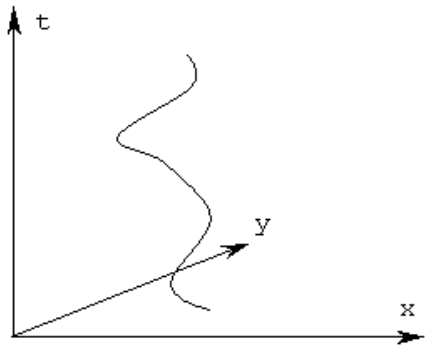
4.1.5 Modelos de datos espacio-temporal con moving objects

Una aproximación distinta al problema consiste en la incorporación al modelo de tipos de datos espacio-temporales. Se trata en este caso de contemplar un concepto conocido como *moving object* (objeto en movimiento. Diferentes trabajos se han realizado en esta línea como los desarrollados por Erwig et al [18] y Sistla et al. [56]. Ambos permiten la consulta de datos espaciotemporales,

capturando esa evolución con respecto al tiempo tanto de la posición como de la extensión.

Se trata de por tanto de aproximaciones que permiten gestionar tanto el cambio como el movimiento de los objetos, con un tratamiento totalmente general, dado que esta evolución puede contemplarse tanto si es continua como discreta. Si únicamente nos interesa registrar el movimiento del objeto, se puede contemplar el objeto como una abstracción en un punto y emplear un moving point para seguir su evolución es decir, podemos considerarlo como una función desde el tiempo en el plano bidimensional o como una polilínea en el espacio tridimensional (2D más tiempo), ver Fig. 25. En cambio, si son cambios en la extensión del objeto, el empleo de las moving region nos permite registrar sus variaciones geométricas, tanto el crecimiento como la disminución del mismo. Se puede considerar esta última como un subconjunto del plano con un interior no vacío o como un polígono con agujeros, según la representación sea o no finita, respectivamente.

Fig. 25: Moving point



Junto con la introducción de esos dos tipos de datos abstractos, en el trabajo de Erwig[18], se incorporan otros tipos de datos auxiliares como los puramente espaciales que tienen su base en trabajos anteriores realizados en la Rose Algebra (ver apartado 3.1.1) y temporales, además de un conjunto de predicados y funciones para su manipulación que pueden ser fácilmente incorporados en cualquier tipo de DBMS, independientemente de si es relacional u objetual.

Así, en el proceso de modelado se definen en primer lugar un conjunto básico de operaciones sobre todos los tipos no temporales, es decir, excluyendo los moving object, tal y como veíamos anteriormente. Una vez se han definido éstas, se extienden consistente y uniformemente a los tipos temporales(moving). Más concretamente define un constructor τ de tipo que transforma cualquier tipo de datos atómico α en un tipo $\tau(\alpha)$ con la semántica:

$\tau(\alpha) = \text{time} \rightarrow \alpha$

de forma que puede denotar los tipos mpoint y mregion como τ(point) y τ(region), respectivamente. Permitiendo así definir un conjunto de operaciones genéricas para los moving object como:

```
τ(α)x time      → α          at
τ(α)            → time       start, stop
```

Finalmente, un conjunto de operaciones sobre los tipos moving object se definen para recuperar información referente a tipos básicos, como por ejemplo:

```
mpoint → line  trajectory
```

Esta aproximación es lo bastante genérica como para incluir los casos donde la geometría cambia de forma discreta y no sólo como una función continua de tiempo. Este hecho proporciona una solución al ejemplo presentado en la Fig. 22, mediante el empleo del tipo moving region.

En cambio, El modelo de datos MOST[56], se considera un modelo en el que existen atributos *estáticos*, en el sentido tradicional, y *dinámicos* , que son aquellos que permiten reflejar una evolución a lo largo del tiempo. Estos atributos dinámicos describen solamente la posición actual y la esperada en el futuro cercano como un vector de movimiento. Se trata de una función de tiempo(*Function(t)*) donde se introducen atributos que cambian continuamente en función del movimiento. Su principal problema se centra en determinar que frecuencia de actualización es necesaria para este vector, de forma que el coste de las actualizaciones frente al de la imprecisión se encuentre balanceado. Esta actualización es realizada automáticamente por el sistema de gestión de base de datos, por lo que el tiempo de validez y de transacción son iguales en este modelo. Al realizarse las actualizaciones periódicamente por el DBMS implica diferentes respuestas en función del instante en que realicemos la consulta, permitiendo incluso realizar consultas sobre el estado futuro de la base de datos.

De esta forma un atributo dinámico, *A*, es representado por tres sub-atributos, *A.value*, *A.UpdateTime* y *A.Function*, que pueden ser consultados independientemente. Así, el valor del atributo *A* en el instante *A.UpdateTime* es *A.value* y hasta el siguiente instante de actualización de *A*, en el instante t_0 será de $A.value+A.Function(t_0)$.

Este modelo, a diferencia del anterior, no es apropiado para el ejemplo de la Fig. 22, dado que no es capaz de registrar un cambio discreto a lo largo del tiempo, solamente, aquellas alteraciones en la posición similares al caso de control de tráfico de flotas, donde interese, por ejemplo, recuperar la posición de un objeto en un determinado instante de tiempo, e incluso predecir su posición futura. No le afectan por tanto los problemas de redundancia de la información que padecían los modelos iniciales.

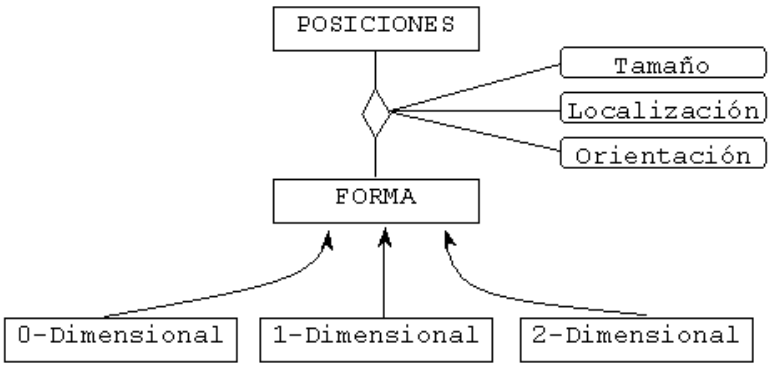
4.1.6 Modelo de espacio-temporal Entidad-Relación

Tryfona et al. [63] han definido un modelo espacio-temporal que contempla el diseño de aplicaciones espacio-temporales, mediante una extensión al modelo entidad-relación y que es conocida como ST-ER. Definen un modelo para ser usado en la fase de diseño conceptual de cualquier aplicación espaciotemporal, previa a la fase de implementación. STRM puede ser empleado para formalmente:

- definir los objetos, atributos y relaciones del dominio de aplicación,
- expresar las vistas de usuario y las consultas al entorno de la base de datos.

Tras definir los objetos, como entidades del mundo real, caracterizados por un conjunto de atributos, de un determinado tipo de datos, e instancias de una determinada clase, cuyo comportamiento queda definido por un conjunto de métodos, introduce el concepto de los objetos espaciales. Para ello tienen en cuenta que para todo objeto en el mundo real su posición queda definida en función de su localización, forma, tamaño y orientación; siendo el modelo de espacio, empleado para definir la localización, euclídeo y definido como homomórfico a los reales. Con el fin de representar la forma se introducen tipos geométricos 0-dimensionales (puntos), 1-Dimensionales (líneas) y 2-Dimensionales (regiones). De esta manera las posiciones son definidas por una generalización (Fig. 26).

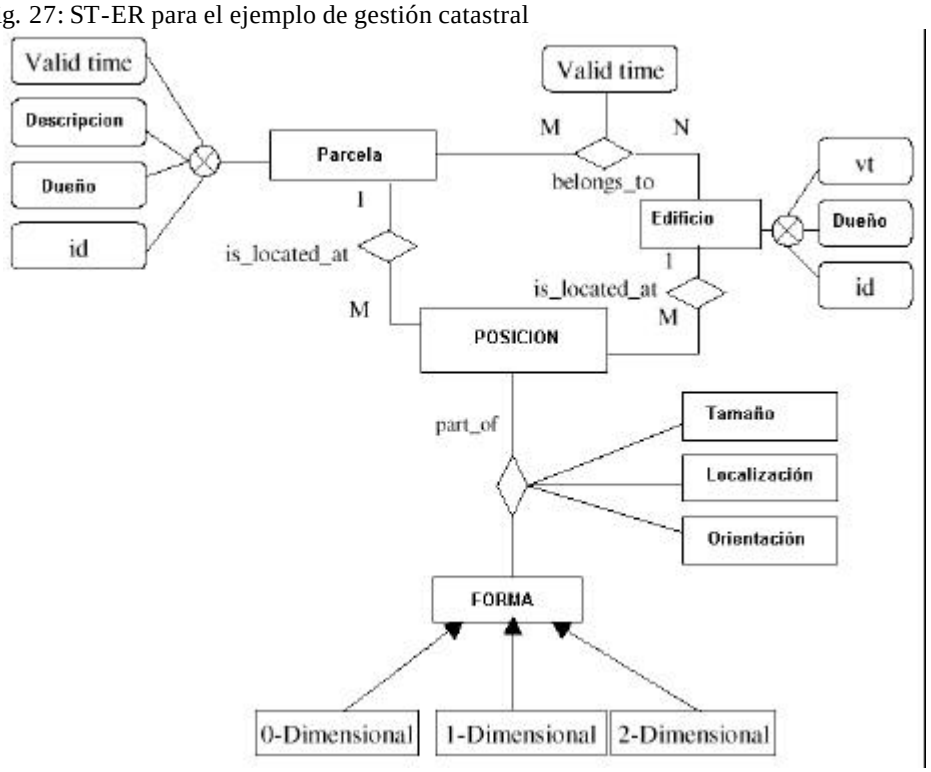
Fig. 26: Modelo de entidades



La dimensión temporal en la que se basa el modelo es lineal, absoluta y discreta, en la que refleja tanto el tiempo de validez como de transacción, de manera que cualquier objeto, atributo y relación le puede ser asignada esa dimensión temporal, y puede por tanto ser indexado sobre ella. Sin embargo, no se considera que los objetos espaciales tengan una extensión temporal realizándose operaciones sobre ellos sin tener en cuenta posibles cambios futuros, es decir, el objeto geométrico es únicamente definido por su posición, por lo que si ésta sufre una modificación, el objeto en sí mismo cambia, se crea un nuevo objeto. Los atributos que varían en el espacio son propiedades del espacio, que

indirectamente llegan a ser propiedades de los objetos situados en esa misma posición del espacio.

Fig. 27: ST-ER para el ejemplo de gestión catastral



En este caso se considera que los atributos poseen una variación, tanto en tiempo como en geometría, como propiedades del espacio, es decir, los atributos que sufren un cambio espacial a lo largo del tiempo, dependen únicamente de la posición no del objeto en sí mismo, cada nueva posición genera un nuevo objeto espacial, una nueva forma. Se podría considerar que el comportamiento es similar a la quantum álgebra en este sentido (apartado 4.1.4). Este tipo de variación se puede contemplar estableciendo una relación uno a muchos entre la entidad y las diferentes posiciones de ellas. En cambio, una relación puede tener diferentes versiones a lo largo del tiempo, por lo que aunque siga siendo la misma, las características que la describen podrían cambiar a lo largo del tiempo.

Se trata en este caso de proporcionar un medio de modelar aplicaciones espaciotemporales de manera que posteriormente puedan ser integradas dentro de una base de datos relacional. En la Fig. 27, se modela el ejemplo de gestión catastral empleando ST-ER. La parcela puede tener asociadas diferentes posiciones a lo largo del tiempo, pero no son las posiciones las que tienen asociadas el tiempo de validez, sino que éste es aplicado sobre el objeto parcela como un atributo de ésta. No aparece en este caso el tiempo de transacción, por no

considerarse necesario en el sistema pero podría ser incluido igual que el tiempo de validez.

En cambio, para el caso de la aplicación de gestión de flotas su aplicación requeriría llegar a un compromiso entre el volumen de información y la precisión, dado que cada movimiento implicaría la generación de un nuevo objeto forma. Este modelo se desarrolló enfocado a Sistemas de Información Geográfica. En particular, a la gestión catastral sin un número de alteraciones tan elevado.

En el modelo no se incluyen referencias acerca de la granularidad del dominio espacial, ni ninguna aproximación de cómo realizar consultas sobre el modelo generado, ni operaciones más allá de las que proporciona el modelo relacional.

4.1.7 Modelo de datos espacio-temporal orientado a objetos

Los modelos de datos orientados a objetos están basados en el paradigma de la orientación a objetos que incluye objetos, clases, encapsulación, herencia y polimorfismo como herramientas. Diferentes aproximaciones pueden acometerse mediante el empleo de este concepto, para introducir la dimensión temporal en el diseño de aplicaciones espaciotemporales. Así, se embebe dentro de una misma entidad las diferentes versiones que ésta ha tendido a lo largo del tiempo, registrando su evolución. Otra aproximación sería construir un espacio 3-dimensional o 4-dimensional en la que el tiempo formará la última dimensión, que se presenta como la alternativa más intuitiva.

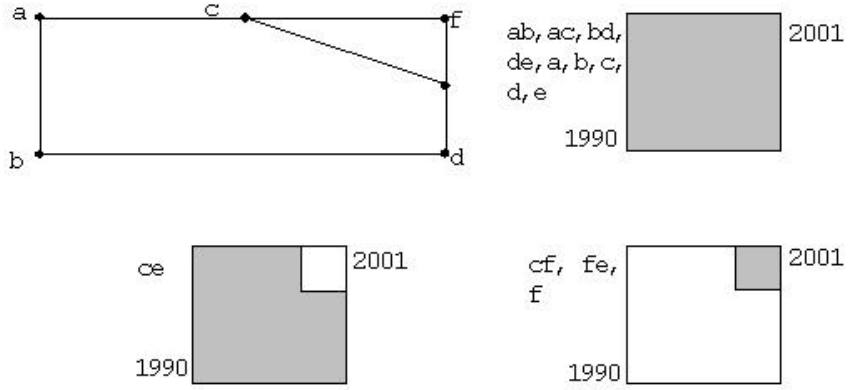
Worboys [67] ha propuesto uno de los primeros de modelos en los que se unifica la información espacial y temporal. En él se propone la introducción del objeto espa**c**iotemporal que tiene un elemento espacial y otro *bitemporal*(BTE). Su representación consiste en etiquetar cada componentes de objetos espaciales simples con BTEs. Estos objetos espaciales simples, son las unidades de representación espacial mínimas no solapadas, denominadas *simplex* (puntos, líneas rectas y triángulos), que representa el modelo. El elemento bitemporal, es definido como la unión de un conjunto finito de productos cartesianos de intervalos representando, en una dimensión, el tiempo de transacción y en otra el tiempo de validez. Con ambos elementos se forma un par ordenado, denominado *ST-simplex*.

La estructura que representa la configuración espacial referenciada bitemporalmente es un *ST-complex*, que está formado por un conjunto de ST-simplex con diferentes restricciones, como: la no repetición de los elementos ST-simplex; que estos tengan una proyección espacial disjunta, así como que cada objeto espacial simplex ha de tener tantas referencias temporales como su nodo padre(por ejemplo, para una nodo final de línea ha de tener tantas referencias como tenga ésta). Sobre estos conjuntos ST-complex, se realizan consultas sobre

la evolución. Además de la definición de estos objetos, se introducen una serie de operaciones, como *equals, union, product, union,...*, así como operaciones de proyección para recuperar la información tanto espacial como bitemporal.

Otra de las cuestiones que se ha de tener en cuenta es en qué nivel incluir la referencia temporal. Una primera opción sería incluirla marca temporal sobre el objeto geométrico completo, limitando con ello la expresividad de las propiedades temporales del objeto a lo largo de su vida. Una segunda opción sería incluirla sobre los objetos espaciales primitivos (puntos, líneas y polígonos), aunque se traduzca en una mayor carga de almacenamiento. O, por último, fundir tiempo y espacio al nivel de punto y propagar después esta unión. En este caso, Worboys introduce esa marca temporal al nivel de los objetos primitivos con objeto de simplificar el modelo.

Fig. 28: Evolución de la parcela P4 como ST-complex



Se puede apreciar en la Fig. 28, como se representaría la parcela P4 en este modelo, donde no se representan las propiedades de área de la parcela sino únicamente sus bordes, para ello sería necesario emplear triángulos en su representación. Cada objeto simplex en la figura (a, b, ab, c, cf, ...) es representado asociándole una referencia bitemporal. Para simplificar su representación se considera que el instante actual es 2001, así como la igualdad del tiempo de validez y de transacción. Cada simplex en la figura es asociado con un elemento bitemporal, que indica su intervalo existencia mediante el tiempo de validez(eje y) y el tiempo de transacción(eje x). Las consultas acerca de la información espaciotemporal se realizarían empleando las proyecciones espaciales, de aquellos objetos simplex que existieran durante el intervalo de tiempo consultado, es decir, mediante una proyección espacial y una proyección temporal.

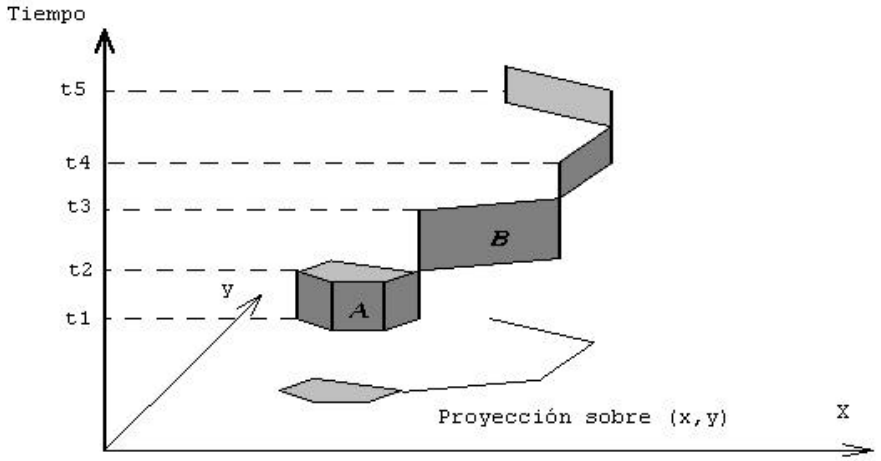
4.1.8 Modelos de datos espacio-temporales polinomiales

Grumbach et al. [25] presentan un modelo de datos basado en el paradigma de las restricciones cuya idea básica es representar de forma finita colecciones de puntos en espacios d-dimensionales. La representación finita usa restricciones expresadas en un lenguaje de primer-orden interpretado en algún dominio aritmético, como por ejemplo el de los racionales y las restricciones lineales $\leq$ y + sobre éste, tal y como se veía anteriormente (apartado 3.1.2). En esta aproximación introduce el espacio y el tiempo como componentes naturales de conjuntos de puntos tridimensionales. Este diseño trata de reflejar el mismo equilibrio que al reflejar la información espacial, es decir, una limitada pérdida de corrección a favor de una mayor eficiencia computacional.

Ahora introduce un nuevo dominio, D, para tratar con esa nueva dimensión temporal de manera que las fórmulas son expresadas en términos de $L \cup D$, con dos tipos de restricciones:

- restricciones de igualdad sobre los objetos de D, y
- restricciones lineales sobre los objetos de Q.

Fig. 29: Un objeto espacio-temporal



En la Fig. 29 se puede apreciar un objeto en movimiento representado como una relación de restricciones lineales en el espacio Q^2 de dimensión ortográfica 2. Para permitir la asociación del espacio y del tiempo se introducen cuatro tuplas. La tupla A, por ejemplo, tiene una geometría que da las posiciones validas del objeto durante un intervalo de tiempo *[t1, t2]*. Esto puede ser representado mediante una restricción sobre *t* y cinco restricciones sobre *x* e *y*, con una partición ortográfica *{{t}, {x, y}}*. Añadiendo la disyunción se refleja la evolución de la geometría, tanto su cambio de forma como alteraciones de posición, con respecto al tiempo. En el ejemplo de la figura, significaría introducir una nueva tupla B, que representaría un instancia de un segmento en el plano asociado con el

intervalo $[t_2, t_3]$. Esto significa que cada tupla esta formada por una proyección del espacio y otra del tiempo.

Para el ejemplo de la Fig. 22, la representación sería mediante un conjunto de tuplas, una para cada una de las parcelas. De manera que uno de los atributos, con la información espacial y temporal, consistiría en una tabla anidada. Cada tupla almacenaría la fórmula de primer orden que define la geometría del objeto, junto con el tiempo de validez que determina el intervalo durante el cual era valida la información, todas ellas unidas mediante la disyunción. Para facilitar su manipulación se restringe el nivel de anidamiento a un nivel.

El ejemplo de la flota de camiones, podría ser modelado fraccionando la trayectoria en intervalos de tiempo, de forma que podamos definir que el vehículo se encontraba entre esos determinados puntos entre esos intervalos de tiempo. Como resultado, se fraccionaría el espacio en rectas que permitiendo determinar su ubicación en cada instante.

Se captura mediante este modelo la evolución temporal de la información espacial, pero a cambio presenta una criticada dificultad para poder establecer consultas de forma sencilla. Ofrece limitaciones para expresar cálculos de distancia o predicados de conectividad. Evita los problemas presentados en los primeros modelos en cuanto a redundancia en el almacenamiento. Aunque no introduce ninguna referencia en cuanto a granularidad de la información temporal o la aplicabilidad del tiempo de validez o de transacción, dejando libertad al usuario.

4.2 Lenguajes de consulta espaciotemporales

Las diferentes aproximaciones presentadas dependen en gran medida del modelo en el que se basan para realizar esa extensión. La riqueza lingüística depende de las aportaciones que pueda ofrecer el modelo en cuanto a las operaciones y predicados que implemente, más concretamente, que semántica sea capaz de aportar. Depende del modelo que el lenguaje sea capaz o no de responder a preguntas sobre el cambio de forma de un objeto geométrico o de alteraciones en su posición. En el apartado anterior se expone como determinados modelos no son capaces de capturar el movimiento de los objetos, o como no pueden establecer una relación entre las diferentes formas que adoptaba porque sencillamente consideran cada nueva forma como un nuevo objeto sin relación con el precedente. Estas carencias en el modelo hacen por tanto que se refleje en el lenguaje. Una de las formas de caracterizar a los lenguajes podría ser su capacidad para dar soporte a consultas relativas al cambio y al movimiento de los diferentes objetos geométricos implicados.

La mayoría de las aproximaciones desarrolladas hasta la fecha, de manera similar a lo que ocurría en las bases de datos espaciales, son desarrolladas a partir

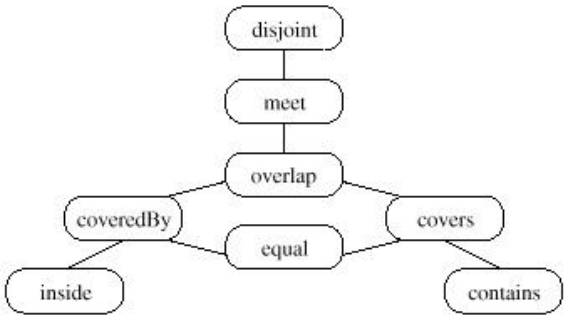
del lenguaje SQL, extendiéndolo para dotarle de la sintaxis necesaria para recoger el tipo de consultas expuestas anteriormente.

Generalmente, en el caso de aquellas alternativas que son orientadas al tipo abstracto de datos, necesitan únicamente realizar una extensión que sea consistente con las definiciones previas. Para lo cual añaden al lenguaje:

- un conjunto básico de predicados espaciotemporales,
- un mecanismo de extensión para nuevos y más complejos predicados espaciotemporales.

No existe, en el caso de los lenguajes espaciotemporales, un acuerdo sobre el conjunto de requisitos que debe cumplir, tal y como ocurría en el caso de los lenguajes de consulta espacial(apartado 3.3, [13]). Los requisitos relativos a la visualización permanecen en este caso, pero el problema estriba en cómo realizar esa visualización. Se trataría entonces de un espacio como mínimo tridimensional, considerando únicamente el espacio euclideo para representar el dominio espacial y el dominio temporal. En las diferentes aproximaciones que se recogen a continuación, no se estable referencia alguna a este tipo de situaciones, sino que las considera totalmente ajenas a la definición del lenguaje, siendo los trabajos relacionados con la interfaz de usuario los que se preocupan de dar respuesta a este tipo de situaciones.

Fig. 30: Grafo de relaciones topológicas



Al igual que para los lenguajes de consulta espacial(apartado 3.2), son necesarias la definición de relaciones topológicas entre cualesquiera dos objetos espaciotemporales. En este caso, al considerar otra dimensión más, es indudable que el número de posibles relaciones se eleve considerablemente. Egenhofer et al. [17] han propuesto el conjunto de relaciones topológicas más próximas, es decir, ante cualquier deformación que puede sufrir un objeto determinar como afecta a las relaciones topológicas que mantenía con otro objeto. Para ello toman el conjunto básico de relaciones que se definieron para el caso espacial (*disjoint, in, touch, equal, cover* y *overlap)* y define para cada una de ellas cual es la relación más próxima, mediante un grafo(Fig. 30), definiendo una relación de orden parcial. Fundamentalmente, presenta una propuesta que permite el *razonamiento*

espacial, es decir, poder establecer cual será la relación topológica siguiente en función del cambio que puede sufrir un objeto.

Una de las aproximaciones es el denominado STSQL (Spatio-Temporal Query Language) [19]. En este lenguaje toda la funcionalidad es capturada mediante los tipos abstractos de datos y las operaciones. Esto implica diversas ventajas en cuanto al mantenimiento de los conceptos de SQL, el tratamiento de los objetos espaciotemporales en un nivel más elevado y la sencillez de incorporación de los predicados espaciales. Sigue por tanto una notación del tipo select-from-where, manteniendo las operaciones y predicados definidas para los tipos espaciales, de acuerdo a la propuesta hecha en SpatialSQL (apartado 3.3). Teniendo en cuenta el conjunto de relaciones topológicas, introducido por Egehofer, definen un conjunto de predicados de acuerdo a los tipos de datos introducidos en el modelo, es decir, moving point y moving region. Si, por ejemplo, se integran estos tipos de datos en un modelo relacional y se define una relación del tipo:

```
Flights(id:string, from:string, to:string, route: mpoint)
```

Se puede responder a consultas donde se quisiera recuperar la ruta entre dos instantes de tiempo:

```
select trajectory (Route(7:00..9:00))
from flights
where id=”UA207”
```

Se trata en este caso de una consulta en la que se entremezcla información espacial, temporal y espaciotemporal dado que la ruta del avión es una información que evoluciona a lo largo del espacio y del tiempo, mediante la notación (..) indica el intervalo de tiempo para el que tiene que hacer la proyección espacial y así recuperar la línea que forma esa trayectoria recogida por el avión.

El lenguaje permite anidamiento entre diferentes consultas, al igual que sucedía en el caso de SpatialSQL. Además admite consultas tanto sobre el movimiento como el cambio de forma de los objetos geométricos. No ofrece en cambio, ninguna base para la especificación de la representación visual.

Future Temporal Logic (FTL) [56] es un lenguaje que permite introducir consultas de una forma sencilla e intuitiva sobre el modelo MOST(apartado 4.1.5). Permite tres tipos de consultas que son instantáneas, continuas y persistentes, a diferencias de las bases de datos tradicionales que solo permiten las dos primeras. La misma consulta puede introducirse empleando alguno de los tipos anteriores, produciendo resultados diferentes ya que dependen de la historia sobre la que se evalúa y del instante de evaluación. Una consulta instantánea es un predicado sobre el estado actual de la base de datos, sin embargo, aunque puede referir a los estados futuros de la base de datos, una consulta continua es un predicado sobre cada uno de sus estados futuros. Una consulta instantánea en un

instante t, es una consulta evaluada sobre la historia infinita comenzando en el instante t.

A diferencia del caso anterior, este lenguaje proporciona herramientas para consultas sobre situaciones futuras. Para el ejemplo, podría responder a preguntas del tipo, dónde estará el avión a las 14:00, es decir, permite hacer predicciones acerca de una posición futura. Estas respuestas son tentativas, dado que dependen de la situación actual, que puede cambiar en cualquier momento. Para ello, introduce los operadores temporales básicos *UNTIL* y *NEXTTIME*, junto con otros operadores que pueden ser definidos en función de los anteriores.

La sintaxis que emplea es similar a OQL, donde que los símbolos de función son empleados para denotar consultas atómicas, es decir, aquellas que devuelven solo un valor. Además, permite su empleo dentro de consultas anidadas, igual que el lenguaje anterior. Un ejemplo de una consulta, en la que se predice la situación de los objetos, obteniendo todos los objetos que entran dentro del polígono P al cabo de tres unidades de tiempo:

```
RETRIEVE o
WHERE Eventually_within 3(INSIDE (o, P))
```

FTL no está pensada para ser introducido sobre cualquier tipo de base de datos, sino específicamente sobre bases de datos MOST. Al igual que en el caso anterior, tampoco presenta características relativas a la visualización de los resultados.

Sobre el modelo de datos espaciotemporal con restricciones (apartado 4.1.8), se ha definido también un lenguaje de consulta [25]que toma como base el lenguaje que se definió para las bases de datos espaciales (apartado 3.3). Emplea las operaciones definidas en el álgebra relacional (producto cartesiano, interesección, etc) y añade un operador más denominado MAP. El motivo es poder trabajar sobre conjuntos de puntos anidados dado que, el modelo emplea anidamiento para almacenar las diferentes representaciones espaciales. Esta operación permite proyectar sobre un determinado instante de tiempo, un conjunto de información a la que aplicar una determinada operación algebraica.

Para el ejemplo anterior de los línea de vuelos, si quisiéramos consultar donde está un avión entre las 12 y las 14 horas, y según la definición de la relación siguiente:

```
Relation flights (id:string
                          path:(from:string, to:string)
                          traj(space: float(2), time:float(1))
```

quedaría como:

```
select (restric traj with '12 < time < 14').space
from flights
where id= 'UA207'
```

Utiliza una sintaxis similar al anterior en cuanto a símbolos de función. Sin embargo, permite su empleo sobre cualquier tipo de base de datos mediante su

apropiada extensión para el tratamiento de ese conjunto de operaciones. Trata de ocultar, de manera similar a la extensión espacial, la complejidad del modelo al usuario, proporcionando un conjunto suficiente de operaciones. Pero, al igual que en los ejemplos anteriores no se introduce ninguna referencia en cuanto a los problemas de visualización de la información.

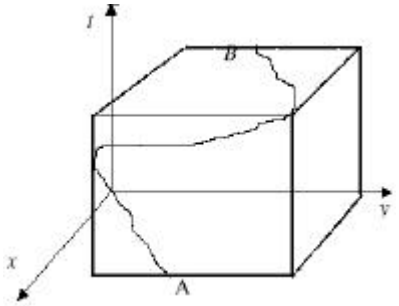
4.3 Métodos de indexación espaciotemporales

Al igual que se veía con las bases de datos espaciales, en las que se precisaban estructuras que facilitarán una recuperación eficiente de la información, las mismas necesidades pueden aplicarse a las bases de datos espaciotemporales. Se trata de presentar estructuras que sean capaces de mantener la historia de cada objeto dinámico, de forma que sean capaces de agilizar el rendimiento de la consulta tanto en espacio como en tiempo. Los requisitos que se exigen a este tipo de estructuras son los mismos para el caso de la indexación espacial (apartado 3.4), en cuanto a independencia de los datos de entrada, dinamismo frente a inserciones, etc. El problema que se plantea es como indexar dos dominios distintos, el espacio y el tiempo, de forma que permita la consulta eficiente sobre ambos. La mayoría de las alternativas que se han propuesto, son extensiones de algunas de las presentadas con anterioridad, como los R-trees(apartado 3.4.1) y quadtrees (3.4.2).

4.3.1 R-trees

Sobre este tipo de índices se han presentado varias modificaciones para poder gestionar tanto la información espacial como temporal. Las más de uso más extendido son 3D R-tree[61], RT-tree[69] y HR-tree[46].

Fig. 31: MBR para un objeto en movimiento



La estructura 3D R-tree trata el tiempo como otra dimensión extra sobre el espacio bidimensional, transformando los rectángulos de búsqueda en paralelepípedos. Por tanto, ello implica que los MBBs no cambian su localización

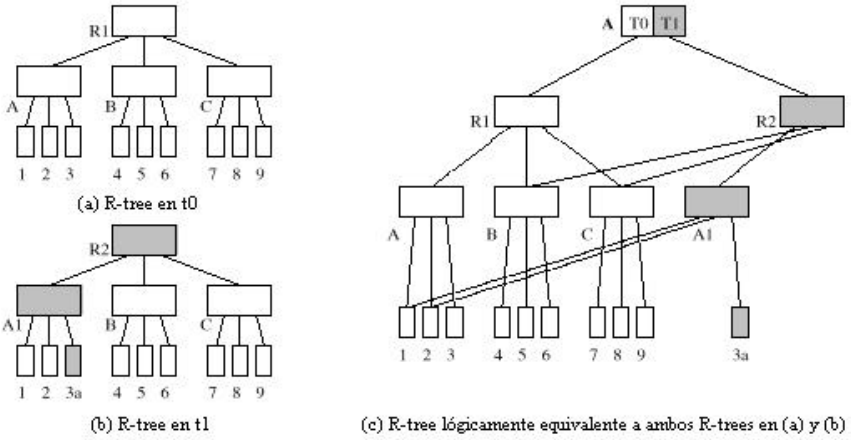
a lo largo del tiempo, ni se introducen espacios no utilizados por su representación tridimensional. Sin embargo, en su utilización para objetos en movimiento genera una gran cantidad de espacio inútil (Fig. 31).

En la estructura RT-tree se soluciona parcialmente este problema anterior. La información temporal queda recogida, mediante intervalos de tiempo, en la estructura bidimensional del árbol R-tree. Cada entrada en un nodo, sea hoja o no, contiene tres entradas *(S, T, P)*, donde *S* es la información espacial (MBB), *T* es la información temporal (intervalo) y *P* es un puntero a un subárbol o a la descripción detallada del objeto que representa(según se trate de un nodo intermedio y hoja, respectivamente). Si $T=(t_i, t_j)$, siendo $i \pounds j$, t_j será el timestamp actual y t_{j+1} será el siguiente timestamp consecutivo. Si un objeto no cambia su localización espacial de t_j a t_{j+1}, entonces su información espacial *S* continua siendo la misma, mientras que *T* es actualizado a *T'*, incrementando su intervalo superior, es decir, $T'=(t_i, t_{j+1})$. Sin embargo, tan pronto como un objeto cambia su situación espacial, una nueva entrada con la información temporal $T'=(t_{j+1}, t_{j+1})$ es creada e insertada en el árbol. Esta estrategia de inserción consigue una estructura eficiente para bases de datos con baja movilidad, pero si el número de objetos que cambian es elevado, entonces demasiadas entradas tienen que ser insertadas en el árbol con un incremento excesivo en el tamaño del árbol. Además, se le critica el hecho de que la construcción de los nodos R-tree depende de la información espacial *S*, mientras *T* desempeña un papel complementario, lo que la hace poco adecuada para implementar consultas temporales del tipo *"encontrar todos los objetos que existen en la base de datos dentro de un intervalo de tiempo dado"*.

La idea básica de HR-tree es emplear dos árboles para representar la evolución de la información espacial. Obviamente, no sería práctico mantener los R-trees de todos los estados previos del actual R-tree, por lo que algunos nodos, por no decir la mayoría, se mantendrán solamente una única copia del mismo, compartiéndose entre diferentes estructuras. Para ello, HR-tree mantiene todos los estados previos R-tree bidimensionalemente solo lógicamente, es decir, emplea dos árboles siendo el más joven una evolución del más viejo. Este último se representa incrementalmente. Podría ser visto como un grafo acíclico más que como una colección de estructuras de árbol independientes. Es una estructura sencilla de comprender e implementar, pero ofrece problemas cuando el número de objetos en movimiento de un instante al siguiente es demasiado elevado, está estructura degenera en estructuras de árboles, debido a la inexistencia de caminos comunes.

En la Fig. 32, se muestran dos R-trees (a y b), temporalmente consecutivos junto con uno de los posibles HR-tree(c) lógicamente equivalente. La inclusión de un array A, en la estructura de indexación, facilita el acceso al timestamp deseado únicamente a través del nodo raíz. De esta manera, se obtiene el mismo coste de procesamiento que si todos los R-trees estuvieran almacenados físicamente.

Fig. 32: R-trees en t0 y t1 y HR equivalente



Es deseable mantener tan bajo como sea posible el número de ramas. Por esa razón algunas variantes de R-tree no son apropiadas para ser empleadas en el desarrollo de HR-trees, como es el caso de R^+-tree(ver apartado 3.4.1). Para este último el MBB es recortado, podría aparecer en diversos nodos internos del árbol. Esto podría ser evitado forzando la reinserción, aunque ello afecte a varias ramas del árbol e incremente su tamaño.

4.3.2 Quadtrees espaciotemporales

La técnicas de indexación basada en quadtrees más conocida es la denominada *Multiversion Linear Quadtree*(MVLQ) [64]. Es una técnica de indexación para almacenamiento y acceso a imágenes ráster en evolución (datos regionales), basada en los *linear quadtree*, empleando técnicas de solapamiento.

Comparte con HR-tree el uso del solapamiento para la generación del árbol, es decir, evita el almacenamiento de subcuadrantes idénticos de instancias sucesivas de datos de imagen, que evolucionan sobre el tiempo de transacción, es decir, pierde rendimiento temporal a cambio de rendimiento en espacio.

Un overlapping linear quadtree consiste en una lista de valores, uno para cada bloque con valor binario 1 que hay en la imagen. Un nodo es una dirección que describe la posición y el tamaño del bloque correspondiente en la imagen. Para cada nodo se almacenan dos partes, la primera de ella es una palabra clave (codeword) que consiste en n digitos, base 4, indicando la dirección (NW, NE, SW y SE). La segunda parte contiene $log_2(n+1)$ dígitos que indican el camino que se ha seguido para llegar a este nodo. Mediante la parte direccional de esos códigos se reconstruye un B^+tree en todos sus niveles y con solapamiento. Es decir, almacena, por cada imagen consecutiva, un grupo de palabras clave que

comparten el mismo tiempo de transacción. Cada una de ellas representa una subregión espacial. De esta manera, la nueva estructura puede ser empleada como un mecanismo de indexación para almacenar y acceder a imágenes máster en evolución. En la Fig. 34 se muestra se muestra el árbol que se generaría para las imágenes de la Fig. 33.

Fig. 33: Dos imágenes binarias

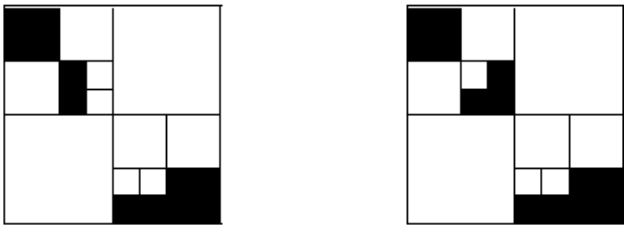
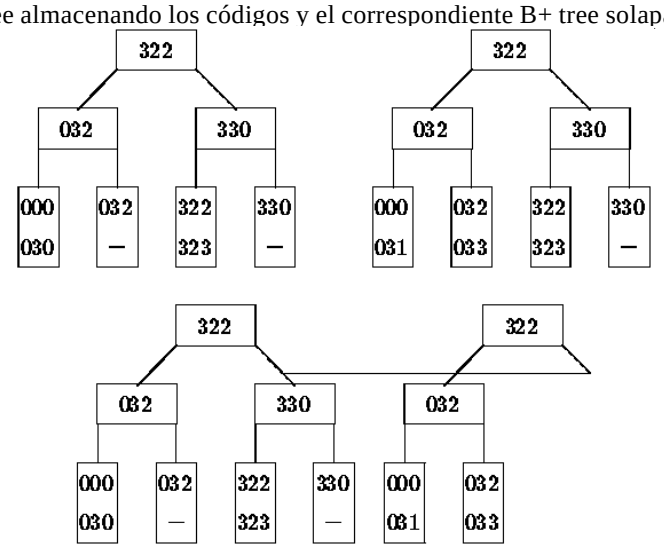


Fig. 34: B+tree almacenando los códigos y el correspondiente B+ tree solapado



Sobre el Overlapping Linear Quadtree se emplea otra estructura para indexar los valores de tiempo de transacción y referencias las raíces de los B^+tree respectivos. Todos los nodos tienen un campo extra, denominado *StartTime* que puede ser empleado para detectar si un nodo está siendo compartido por otros árboles. Este valor se asigna en el momento de creación del nodo correspondiente sin que sean necesarias modificaciones futuras. En cambio, los nodos hoja tiene un campo extra denominado *EndTime*, empleado para registrar el momento de la transacción cuando una hoja específica cambia, y pasa a ser histórica.

5 Conclusiones

El presente trabajo da una visión global de las diferentes tendencias que se están realizando, en el área de las bases de datos temporales y espaciales, y de cómo han posibilitado, mediante su integración la definición de las bases de datos espaciotemporales.

Para ello se han presentado los modelos de datos más relevantes, junto con una clasificación de los mismos, a fin de poder comprender la semántica que se pretende capturar en ambos dominios, espacial y espaciotemporal. Además, se presentan algunas de las características más representativas de los lenguajes de consulta que permiten recuperar este tipo de información, así como algunos ejemplos que permiten apreciar que sintaxis de algunos de ellos. Finalmente, se han introducido las técnicas de indexación más representativas en ambos dominios.

Se han podido observar diferentes lagunas en el área de las bases de datos espaciales, que ven su reflejo en las espaciotemporales, algunas de ellas de gran importancia. La inexistencia de un álgebra estándar ha motivado la falta en numerosas ocasiones de una semántica común sobre las operaciones, y por tanto, también en el dominio espaciotemporal. Carencia que se ha podido observar también en los diferentes lenguajes presentados.

Se ha hecho patente, en las estructuras de indexación espaciotemporal, el desequilibrio entre el acceso a la información temporal frente a la espacial, que restringe la eficiencia en las consultas que implican movimiento o cambio.

Este trabajo supone el punto de partida para el de las diferentes arquitecturas de implementación, extensiones de los sistemas gestores, que se están desarrollando en la actualidad, soportadas por los conceptos aquí introducidos.

6 Bibliografía

1. J.F.Allen, Maintaining knowledge about Temporal Intervals, Communications of ACM 16, No. 11, November, 1983.
2. Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles, Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'90), Atlantic City, NJ, pp. 322-331, 1990.
3. E. Bertino, E. Ferrari, G. Guerrini, A Formal Temporal Object-Oriented Data Model, Proceedings of the 5th International Conference on Extending Database Technology,1996.
4. R. Bljujute, C. S. Jensen, S. Saltenis and G. Slivinskas, R-tree based indexing of now-relative bitemporal data, TimeCenter Technical Report TR-25 (1998).
5. E. P.F. Chan and R. Zhu, QL/G – A query language for geometric data bases, Proceeding of the 1st International Conference on GIS in Urban Regional and Environmental Planning, Samos, Greece, pp. 271-286, April 1996
6. J. Chomicki, Temporal Query Languages: a Survey. Proc. International Conference on Temporal Logic, *July 1994*, Bonn, Germany, Springer-Verlag (LNAI 827), pp. 506-534.
7. E. Clementini, P. Di Felice, and P. van Oosterom., A Small Set of Formal Topological Relationships for End-User Interaction. , D. Abel and B. C. Ooi (Ed.), Advances in Spatial Databases - Third International Symposium, SSD'93.
8. C. J. Date, An Introduction to Database Systems. Chapter 22/ Temporal Databases, Addison-Wesley Pub Co, 7th Ed, October 1999.
9. J.R. Davis , IBM's DB2 Spatial Extender: Managing Geo-Spatial Information Within the DBMS. , Technical report, IBM Corp., May 1998.
10. D.J. DeWitt, N. Kabra, J. Luo, J.M. Patel and J. Yu, Client-Server Paradise, Proceedings of the 20th International Conference on Very Large Data Bases, (VLDB'94), J. B. Bocca, M. Jarke, C. Zaniolo (eds.), Morgan Kaufmann, Santiago de Chile, Chile, (1994) 558-569.
11. C. Dyreson, W. Evans, H. Lin, and R. T. Snodgrass, Efficiently Supporting Temporal Granularities, To appear in IEEE Transactions on Knowledge and Data Engineering, 38 pages, 2000
12. ESRI White Paper, ArcSDE—The Universal Spatial Server for ARC/INFO, April 1999
13. M. Egenhofer and A. Frank, Towards a Spatial Query Language: User Interface Considerations. 14th International Conference on Very Large Data Bases, Long Beach, CA, D. DeWitt and F. Bancilhon (eds.), pp. 124-133, August 1988.
14. M. J. Egenhofer and Herring, J. R., Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases, 4th International Symposium on Spatial Data Handling, Zurich, 1990.

15 M. Egenhofer, Why not SQL!, International Journal of Geographical Information Systems 6 (2): 71-85, 1992.

16 M. J. Egenhofer, Spatial SQL, IEEE Transactions on Knowledge and Data Engineering 6(1), 86-95, 1994.

17 M. Egenhofer and K. Al-Taha, Reasoning about Gradual Changes of Topological Relationships *Theory and Methods of Spatio-Temporal Reasoning in Geographic Space*, Pisa, Italy A. Frank, I. Campari, and U. Formentini (eds.), *Lecture Notes in Computer Science*, Vol. 639, Springer-Verlag, pp. 196-219, September 1992.

18 M. Erwig, R. H. Güting, M. Schneider and M. Vazirgiannis, Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases, GeoInformatica,Vol. 3,No. 3, 269-296, 1999.

19 M. Erwig, and M.Schneider, Developments in Spatio-Temporal Query Languages, IEEE International Workshop on Spatio-Temporal Data Models and Languages, 441-449, 1999.

20. R. Fegaras, A. Elmasri, Temporal Object Query Language TIME 1998: 51-59

21. V. Gaede and O. Günther, Survey on Multidimensional Access Methods , Technical Report ISS-16, August 1995

22. M. Gargano, E. Nardelli and M. Talamo, Abstract Data Types for the Logical Modelling of Complex Data, Inofrmation Systems, Vol. 16, No. 6, pp. 565-583, 1991.

23. I. A. Goralwalla, A. U. Tansel and M. T. Ozsu Experimenting with Temporal Relational Databases, Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM), 1995

24. F. Grandi, F. Mandreoli, M. R. Scalas A Formal Model for Temporal Schema Versioning in Object-Oriented Databases. CSITE-014-98, November 1998.

25. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints, In Proc. Intl. Symp. on Geographic Information Systems, 1998.

26. S. Grumbach, P. Rigaux, L. Segoufin, The DEDALE System for Complex Spatial Queries, Laura M. Haas, Ashutosh Tiwary (Eds.): SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA. ACM Press (1998), 213-224.

27. A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. ACM SIGMOD, 1984.

28. R.H. Güting, Second-Order Signature: A Tool for Specifying Data Models, Query Processing, and Optimization. Proccedings ACM SIGMOD Conference, Washington, pp. 277-286, May 1993.

29. R.H. Güting, An Introduction to Spatial Database Systems, VLDB Journal 3, pp. 357-399, 1994.

30. R.H. Güting, GraphDB: A Data Model and Query Language for Graphs in Databases. FernUniversität Hagen, Informatik-Report 155, February 1994

31. R.H. Güting and M. Schneider, Realm-Based Spatial Data Types: The ROSE Algebra. VLDB Journal 4, 100-143, 1995.

32. R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis, A Foundation for Representing and Querying Moving Objects, FernUniversität Hagen, Informatik-Report 238. Versión revisada por aparecer en ACM Transactions on Database Systems, Septiembre 1998

33. T. Hadzilacos, N. Tryfona, An Extended Entity-Relationship Model for Geographic Applications, ACM SIGMOD Record 26(3).

34. Informix white paper, Managing Spatial Data: The ESRI Spatial Database Engine for Informix, 1999.

35. C. S. Jensen J. Clifford S. K. Gadia A. Segev R. T. Snodgrass, A Glossary of Temporal Database Concepts, SIGMOD Record, Vol.21, No.3, September 1992.

37. Kanellakis, P.C., Kuper, G.M., Revesz, P.Z., Constraint Query Languages, Journal of Computer and System Sciences, vol. 51, no. 1, pp. 26-52, 1995

38. G. Langran, A Framework for Temporal Geographic Information Systems, Cartographica, 25 (3), 1988

39. N. A. Lorentzos and Y. Manolopoulos, Functional requirements for historical and interval extensions to the relational model, IEEE Data & Knowledge Engineering 17, 59 - 86, 1995

40. N. A. Lorentzos and Y. G. Mitsopoulos, SQL Extension for Interval Data, IEEE Transactions on knowledge and Data Engineering 9, No. 3, May/June 1997

41. N. A. Lorentzos, Nectaria Tryfona, Jose Ramon Rios Viqueira: Regional Algebra for Spatial Data Management. Integrated Spatial Databases, pp. 192-208, 1999

43. N. A. Lorentzos, J. R. Rios Viqueira, N. Tryfona, On a Spatiotemporal Relational Model Based on Quanta, Chapter 4.4.3 of the CHOROCHRONOS book, in lit.

45. P. McBrien, A. H. Seltveit, and B. Wangler. An Entity-Relationship Model Extended to Describe Historical Information. In International Conference on Information Systems and Management of Data, pages 244--260, Bangalore, India, July 1992.

46. M Nascimento and J.R.O. Silva. Towards historical Rtrees. In Proc. of the 1998 ACM Symposium on Applied Computing, pages 235 { 240, February 1998.

47 Oracle White Paper, Oracle ® Spatial, Data Sheet, March 1999.

48 D.J Peuquet y N. Duan, An Event-Based Spatio-temporal Data Model Review of SpatioTemporal Data Models 50 (ESTDM) for Temporal Analysis of Geographical Data, Int. Journal of Geographical Information Systems, vol. 9, no. 1, pp. 7-24, 1995.

49. J. F. Roddick, F. Grandi, F. Mandreoli y M. R. Scalas Towards a Model for Spatio-Temporal Schema Selection. W06 - Tenth International Workshop on Database and Expert Systems Applications), Florence, Italy, IEEE Computer Society Press, 1999.

50. N. Roussopoulos, C. Faloutsos, T. K. Sellis: An Efficient Pictorial Database System for PSQL. IEEE Transactions on Software Engineering 14(5): 639-65, 1988.

51. B. Salzberg and V. J. Tsotras. Comparison of Access Methods for Time-Evolving Data. ACM Computing Surveys 31(2), pp. 158-221, 1999.

52. H. Samet. Spatial Data Structures. Modern Database Systems. The Object Model, Interoperability, and Beyond, pp. 361-385, 1995.

53. H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, Reading, Ma. 1990

54. T. K. Sellis, N. Roussopoulos, C. Faloutsos: The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. VLDB, pp. 507-518, 1987.

55. M. Scholl and A. Voisard. Thematic Maps Modeling, in Design and Implementation of Large Spatial Databases, Lecture Notes in Computer Science No. 409, A. Buchmann et al. (Eds.), Springer-Verlag, Berlin, 1989.

56. P. Sistla, O. Wolfson, S. Chamberlain, S. Dao, Querying the Uncertain Position of Moving Objects, capítulo en Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada eds., Springer Verlag Lecture Notes in Computer Science nº 1399, pp. 310-337, 1998.

57. R. T. Snodgrass, I. Ahn, G. Ariav, D. Batory, J. Clifford, C. E. Dyreson, C. S. Jensen, R. Elmasri, F. Grandi, W. Käfer, N. Kline, K. Kulkarni, T. Y. Cliff Leung, N. Lorentzos, J. F. Roddick, A. Segev, M. D. Soo, and S. M. Sripada, TSQL2 Language specification, ACM SIGMOD Record 23 (1), 65-86, September 1994

58. R. T. Snodgrass Temporal databases, In Frank, A. U. (ed.) Theories and methods of Spatio-Temporal Reasoning in Geographic Space. Pp 22-64, Springer, Berlin.

59. M. Stonebraker, L. A. Rowe and M. Hirohama, The Implementation of Postgres, IEEE Transactions on Knowledge and Data Engineering, 1990

60. B Theodoulidis, et al. Review of Temporal Object-Oriented Approaches, Timelab Technical Report TR-96-1 (1996)

61. Y. Theodoridis, M. Vazirgiannis, and T. Sellis. Spatio-temporal indexing for large multimedia applications. In Proc. of the 3rd IEEE Conf. on Multimedia Computing and Systems, pages 441 { 448, June 1996.

62. K. Torp, C. S. Jensen, and M. Boehlen, Layered Temporal DBMSs - Concepts and Techniques, Proc. Fifth International Conference on Database Systems for Advanced Applications, Melbourne, Australia, April 1-4, 10 pages, 1997.

63. N. Tryfona and Th. Hadzilacos, Logical Data Modeling for Spatio-Temporal Applications: Definitions and a Model, International Database Engineering and Applications Symposium, IEEE Press Proceedings in lit, 1998

64. Tzouramanis T., Vassilakopoulos M., and Manolopoulos Y.: Overlapping Linear Quadtrees - a Spatio-Temporal Access Method", Proceedings 6th ACM Workshop on Advances in Geographical Information Systems (ACM-GIS 98), Bethesda MD, November 1998.

65. T. Vijlbrief and P. Van Oosteron, The GEO++ System: An Extensible GIS, Proceedings of the 5th International Symposium on Spatial Data Handling, Vol. 1, (1992), 40-50.

66. A. Voigtmann, L. Becker, K. Hinrichs An Object-Oriented Data Model and a Query Language for Geographic Information Systems, Bericht Nr. 15/95-I, Institut für Informatik, Westf. Wilhelms-Universität, Münster, 1995

67. M. F. Worboys, A Unified Model for Spatial and Temporal Information, The Computer Journal, Vol. 37, No. 1, pp. 27-34, 1994.

68. M. F. Worboys, Imprecision in Finite Resolution Spatial Data, GeoInformatica, 2(3):257-279, 1998.

69. X. Xu, J. Han, and W. Lu. RT-tree: An improved R-tree index structure for spatiotemporal databases. In Proc. of the 4th Intl. Symposium on Spatial Data Handling, pages 1040 - 1049, 1990.

7 Índice

8 Lista de figuras