MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-12833-PRE/8123

CAPÍTULO 5

ARQUITETURAS E LINGUAGENS

Karine Reis Ferreira
Marco Antonio Casanova
Gilberto Ribeiro de Queiroz
Olga Fradico de Oliveira

Bancos de dados geográficos

INPE
São José dos Campos
2005

5 Arquiteturas e linguagens

Karine Reis Ferreira
Marco Antonio Casanova
Gilberto Ribeiro de Queiroz
Olga Fradico de Oliveira

5.1 Introdução

Este capítulo inicialmente apresenta uma visão geral de sistemas de gerência de banco de dados (SGBDs) e resume os principais conceitos da linguagem SQL. Em seguida, aborda a evolução das arquiteturas de SIG. Por fim, apresenta os principais operadores espaciais e discute a definição de linguagens de consulta espacial.

Ao longo dos anos, as implementações de SIGs seguiram diferentes arquiteturas, distinguindo-se principalmente pela estratégia adotada para armazenar e recuperar dados espaciais. Mais recentemente, tais arquiteturas evoluíram para utilizar, cada vez mais, recursos de SGBDs.

Por seu lado, a pesquisa na área de Banco de Dados passou, já há algum tempo, a preocupar-se com o suporte a aplicações não convencionais (Schneider, 1997), incluindo as aplicações SIG. Uma aplicação é classificada como não convencional quando trabalha com outros tipos de dados, além dos tradicionais, como tipos de dados espaciais, temporais e espaço-temporais. Uma das vertentes de pesquisa tem sido exatamente a definição de linguagens de consulta para tratar tais tipos de dados.

5.2 Preliminares

5.2.1 Sistemas de gerência de banco de dados

Um sistema de gerência de banco de dados (SGBD) oferece serviços de armazenamento, consulta e atualização de bancos de dados. A Tabela 5.1 resume os requisitos mais importantes para SGBDs e a Tabela 5.2 lista as principais tecnologias desenvolvidas para atendê-los.

Tabela 5.1 – Principais requisitos para SGBDs

Requisito	Definição
Facilidade de uso	a modelagem do banco de dados deve refletir a realidade das aplicações, e o acesso aos dados deve ser feito de forma simples
Correção	os dados armazenados no banco de dados devem refletir um estado correto da realidade modelada
Facilidade de manutenção	alterações na forma de armazenamento dos dados devem afetar as aplicações o mínimo possível
Confiabilidade	atualizações não devem ser perdidas e não devem interferir umas com as outras
Segurança	o acesso aos dados deve ser controlado de acordo com os direitos definidos para cada aplicação ou usuário
Desempenho	o tempo de acesso aos dados deve ser compatível com a complexidade da consulta

Tabela 5.2 – Principais tecnologias para SGBDs

Requisito	*Tecnologia*
Facilidade de uso	linguagem de definição de dados e linguagem de consulta baseadas em modelo de dados de alto nível
Correção	restrições de integridade, *triggers* e assertivas
Facilidade de manutenção	especificação do banco de dados em níveis, isolando os detalhes de armazenamento das aplicações

Confiabilidade	transações atômicas implementadas através de mecanismos para controle de concorrência e mecanismos de recuperação em caso de falhas
Segurança	níveis de autorização e controle de acesso
Desempenho	otimização de consultas, baseada em métodos de acesso e de armazenamento eficientes, gerência eficaz do buffer pool e modelos de custo, entre outras tecnologias

O mercado para SGBDs concentra-se em duas tecnologias, SGBDs Relacionais (SGBD-R) e SGBDs Objeto-Relacionais (SGBD-OR), com uma pequena fatia para SGBDs Orientados-a-Objeto (SGBD-OO).

Os SGBD-R seguem o modelo relacional de dados, em que um banco de dados é organizado como uma coleção de relações, cada qual com atributos de um tipo específico. Nos sistemas comerciais atuais, os tipos incluem números inteiros, de ponto flutuante, cadeias de caracteres, datas e campos binários longos (BLOBs). Para esses tipos encontram-se disponíveis uma variedade de operações (exceto para o tipo BLOB), como operações aritméticas, de conversão, de manipulação textual e operações com data.

Os SGBD-R foram concebidos para atender as necessidades de aplicações manipulando grandes volumes de dados convencionais. De fato, tais sistemas não oferecem recursos para atender as necessidades de aplicações não convencionais. A mera simulação de tipos de dados não convencionais em um SGBD-R pode ter efeitos colaterais, como queda de desempenho, dificuldade de codificação e posterior manutenção da aplicação (Stonebraker, 1996).

Os SGBD-OR estendem o modelo relacional, entre outras características, com um sistema de tipos de dados rico e estendível, oferecendo operadores que podem ser utilizados na linguagem de consulta. Possibilitam ainda a extensão dos mecanismos de indexação sobre os novos tipos. Essas características reduzem os problemas ocorridos na simulação de tipos de dados pelos SGBD-R, tornando os SGBD-OR uma solução atrativa para aplicações não convencionais.

5.2.2 A linguagem SQL

A linguagem SQL (*Structured Query Language*) é adotada pela maioria dos SGBD-R e SGBD-OR comerciais. Desenvolvida inicialmente pela IBM na década de 70, a linguagem sofreu sucessivas extensões, culminando com a publicação do padrão conhecido por SQL:1999 (ver Tabela 5.3).

Tabela 5.3 – Breve histórico de SQL

Ano	*Versão*	*Características*
1974	SEQUEL	linguagem original, adotada no protótipo de mesmo nome, desenvolvido pela IBM
1976	SEQUEL 2	extensão de SEQUEL, adotada no *System R* da IBM
1986	SQL-86 (SQL1)	padrão publicado pela ANSI em 1986 ratificado pela ISO em 1987
1989	SQL-89	extensão do SQL-86
1992	SQL-92 (SQL2)	padrão publicado pela ANSI e pela ISO
1996	SQL-92 / PSM	extensão do SQL-92
2001	SQL:1999	padrão aprovado em 1999 pela ISO, resultado de 7 anos de trabalho, e publicado em maio de 2001

SQL é formada basicamente por duas sub-linguagens:

- Linguagem de definição de dados (SQL DDL): fornece comandos para definir e modificar esquemas de tabelas, remover tabelas, criar índices e definir restrições de integridade.
- Linguagem de manipulação de dados (SQL DML): fornece comandos para consultar, inserir, modificar e remover dados no banco de dados.

A Tabela 5.4 apresenta alguns comandos em SQL.

Tabela 5.4 – Comandos em SQL

Comando	Descrição	Tipo
select	Recupera dados de uma ou mais tabelas	DML
insert *update* *delete*	Servem para incluir, alterar e eliminar registros de uma tabela, respectivamente	DML
commit *roolback*	Responsáveis pelo controle de transações, permitem que o usuário desfaça (*rollback*) ou confirme (*commit*) alterações em tabelas	DML
create *alter* *drop*	Usados para definir, alterar e remover tabelas de um banco de dados	DDL

No contexto de SQL, usaremos os termos "tabela" em lugar de "relação", e "coluna" em lugar de "atributo", seguindo a prática mais comum. Os tipos de dados de SQL mais comumente utilizados são *char(n), int, float, date* e *time*. Estes representam, respectivamente, um conjunto de caracteres de tamanho definido, um número inteiro, um número real, uma data (composta por dia, mês e ano) e um instante de tempo (composto por horas, minutos e segundos).

Uma restrição especifica um critério de consistência para o banco de dados, podendo ser definida a nível de coluna ou a nível de tabela. A restrição de nulidade sobre uma coluna A de uma tabela T indica que nenhuma linha t de T pode ter o valor de A nulo.

Uma chave primária K (*primary key*) de uma tabela T é formada pelo nome de uma única coluna ou por uma lista de nomes de colunas de T. Indica que quaisquer duas linhas t e t' de T não podem ter valores idênticos de K, ou seja, $t[K] \neq t'[K]$.

Uma chave estrangeira F (*foreign key*) de uma tabela T para uma tabela U, com chave K, também é formada pelo nome de uma única

coluna ou por uma lista de nome de colunas de T. Indica que, para cada linha t de T, deve haver uma linha u de U tal que t[F]=u[K].

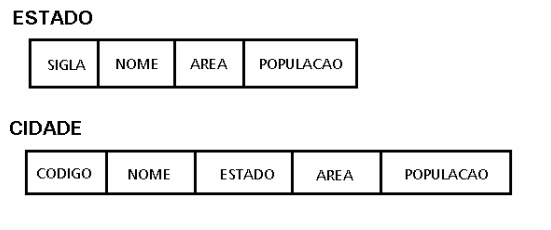

Figura 5.1 – Relações cidade e estado.

A seguir, apresentaremos alguns exemplos de SQL, baseados nas tabelas `cidade` e `estado`, ilustradas na Figura 5.1 e definidas como:

```
CREATE TABLE estado
(  sigla   CHAR(2)    NOT NULL,
   nome    CHAR(30),
   area    FLOAT,
   PRIMARY KEY(sigla) )

CREATE TABLE cidade
(  codigo  INT        NOT NULL,
   nome    CHAR(30)   NOT NULL,
   estado  CHAR(2),
   area    FLOAT,
   PRIMARY KEY(codigo),
   FOREIGN KEY(estado) REFERENCES estado(sigla) )
```

No exemplo anterior, `codigo` é a chave primária da tabela `cidade`, e `estado` de `cidade` atua como uma chave estrangeira referenciando a chave `sigla` da tabela `estado`. Assim, garantimos que somente os valores que ocorrem na coluna `sigla` de `estado` podem ser usadas como valores na coluna `estado` em `cidade`. Ainda, nessa tabela, as colunas `codigo` e `nome` não podem conter valores nulos.

Para alterar a definição de uma tabela, usamos o comando `ALTER TABLE`:

```
ALTER TABLE estado ADD região CHAR(20)
```

Para remover tanto a definição quanto os dados de uma tabela, usamos o comando `DROP TABLE`:

```
DROP TABLE cidade
```

Para remover apenas os dados, usamos o comando `DELETE`:

```
DELETE * FROM estado
```

Para inserir uma nova linha, usamos o comando `INSERT`:

```
INSERT INTO estado
VALUES ('MG', 'Minas Gerais', 10000000)
```

Para alterar dados já existentes, usamos o comando `UPDATE`:

```
UPDATE estado   SET area = 20000000
WHERE sigla = 'MG'
```

Para consultar os dados, usamos o comando `SELECT`, cuja sintaxe é resumida a seguir (ver a Tabela 5.5):

```
SELECT [distinct] {*, coluna [alias],
        expressões [alias], funções [alias], ...}
from {tabelas [alias],}
[where condição]
[group by colunas]
[having condição]
[order by colunas [asc | desc]]
```

onde os colchetes representam cláusulas opcionais, as chaves indicam que os elementos podem aparecer repetidas vezes e a barra vertical indica que as opções são mutuamente exclusivas (ou uma ou outra).

Tabela 5.5 – Sintaxe dos comandos

Opções	Descrição
`distinct`	Indica que as linhas duplicadas devem ser eliminadas do resultado da consulta.
*	Indica que todas as colunas de todas as tabelas da cláusula *from* devem ser incluídas nas linhas da resposta da consulta.
`coluna`	Coluna de uma tabela listada na cláusula *from* que deve ser

	incluída em cada linha da resposta da consulta.
`expressões`	Expressões aritméticas envolvendo uma ou mais colunas das tabelas listadas na cláusula *from*.
`funções`	Funções definas em SQL como, por exemplo, funções de agregação, que calculam estatísticas sobre colunas numéricas (*avg, min*, *max*, *count*, dentre outras).
`alias (select)`	Nome alternativo para uma coluna ou expressão, usado para melhorar a legibilidade do comando, ou para nomear diretamente uma coluna da resposta da consulta.
`tabelas`	Uma ou mais tabelas envolvidas na consulta.
`alias (from)`	Nome alternativo para uma tabela, usado para melhorar a legibilidade, ou para permitir o uso da mesma tabela mais de uma vez na cláusula *from*. Pode ser precedido por "*as*".
`condição (where)`	Especifica uma condição à qual as linhas das tabelas listadas na cláusula *from* devem satisfazer para gerar uma linha da resposta da consulta.
`group by colunas`	Indica que o resultado deve ser agrupado pelas colunas especificadas.
`condição (having)`	Limita os grupos a serem mostrados àqueles satisfazendo a condição.
`order by`	Ordenação que será aplicada ao resultado da consulta, podendo ser crescente (*asc*) ou decrescente (*desc*).

As consultas a seguir ilustram o uso do comando *select*:

Q1. Selecione todos os nomes dos estados.

```
SELECT nome
FROM estado
```

Q2. Selecione todos os nomes das cidades, sem repetição.

```
SELECT DISTINCT nome
FROM cidade
```

Q3. Recupere os nomes e as áreas das cidades que possuam a sigla de estado igual a "MG" e população maior que 50.000 habitantes.

```
SELECT nome, area
FROM cidade
WHERE estado = "MG" AND populacao > 50000
```

Q4. Retorne todas as cidades e o estado ao qual pertence.

```
SELECT C.nome, E.nome
FROM cidade AS C, estado AS E
WHERE C.estado = E.sigla
```

Q5. Calcule a média das áreas de todas as cidades.

```
SELECT AVG(area)
FROM cidade
```

Q6. Retorne o nome de cada estado e a soma das áreas de todas as cidades desse estado.

```
SELECT E.nome, SUM(C.area)
FROM estado AS E, cidade AS C
WHERE E.sigla = C.estado
GROUP BY E.nome
```

Além do comando *select*, os comandos *delete* e *update* podem usar a cláusula *where* como, por exemplo:

Q7. Remova as cidades cuja população é menor que 1000 habitantes.

```
DELETE FROM cidade
WHERE populacao < 1000
```

Q8. Altere a população do estado de Minas Gerais para 5000000.

```
UPDATE estado SET populacao = 500000
WHERE codigo = "MG"
```

5.3 Arquiteturas de SIGs

A partir desta seção, usaremos os conceitos introduzidos na Tabela 5.6 e definidos em detalhe ao longo das seções seguintes ou em outros capítulos deste texto.

Tabela 5.6 – Resumo dos principais conceitos relativos a bancos de dados espaciais

Conceito	*Definição*
geometria matricial	uma matriz de elementos, usualmente pertencentes a um sub-conjunto dos números reais $\Re$
geometria vetorial	um elemento do $\Re^2$, considerado como um espaço topológico.
	Pontos, linhas e regiões são particulares geometrias.
geometria	uma geometria vetorial ou matricial
atributo espacial	um atributo de um objeto cujo domínio seja um conjunto de geometrias.
objeto espacial	qualquer objeto com um atributo espacial.
	Os geo-objetos são uma classe particular de objetos espaciais.
componente espacial ou geometria de um objeto	valor de um atributo espacial de um objeto.
banco de dados espacial	um banco de dados armazenando, entre outros, objetos espaciais.
consulta espacial	uma consultas definida sobre um banco de dados espacial.

Existem basicamente duas principais formas de integração entre os SIGs e os SGBDs, que são a *arquitetura dual* e a *arquitetura integrada*.

A *arquitetura dual*, mostrada na Figura 5.2 armazena as componentes espaciais dos objetos separadamente. A componente convencional, ou alfanumérica, é armazenada em um SGBD relacional e a componente

espacial é armazenada em arquivos com formato proprietário. Os principais problemas dessa arquitetura são:

- Dificuldade no controle e manipulação das componentes espaciais;
- Dificuldade em manter a integridade entre a componente espacial e a componente alfanumérica;
- Separação entre o processamento da parte convencional, realizado pelo SGBD, e o processamento da parte espacial, realizado pelo aplicativo utilizando os arquivos proprietários;
- Dificuldade de interoperabilidade, já que cada sistema trabalha com arquivos com formato proprietário.

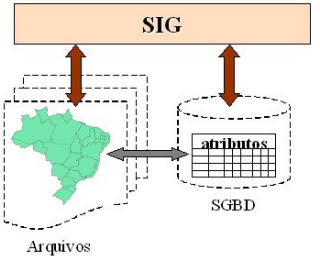

Figura 5.2 – Arquitetura Dual

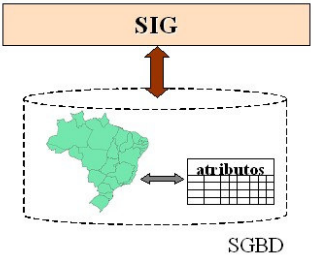

Figura 5.3 – Arquitetura Integrada

A *arquitetura integrada*, mostrada na Figura 5.3, consiste em armazenar todos os dados em um SGBD, ou seja, tanto a componente espacial quanto a alfanumérica. Sua principal vantagem é a utilização dos recursos de um SGBD para controle e manipulação de objetos espaciais, como gerência de transações, controle de integridade, concorrência e linguagens próprias de consulta. Sendo assim, a manutenção de integridade entre a componente espacial e alfanumérica é feita pelo SGBD.

Esta última arquitetura pode ainda ser subdividida em três outras: *baseada em campos longos, em extensões espaciais* e *combinada*.

A *arquitetura integrada baseada em campos longos* utiliza BLOBs para armazenar a componente espacial dos objetos. Como comentado anteriormente, um SGBD-R ou -OR trata um BLOB como uma cadeia de bits sem nenhuma semântica adicional. Portanto, esta arquitetura apresenta algumas desvantagens:

- um BLOB não possui semântica;
- um BLOB não possui métodos de acesso;
- SQL oferece apenas operadores elementares de cadeias para tratar BLOBs.

Portanto, ao codificar dados espaciais em BLOBs, esta arquitetura torna a sua semântica opaca para o SGBD. Ou seja, passa a ser responsabilidade do SIG implementar os operadores espaciais, capturando a semântica dos dados, e métodos de acesso que possam ser úteis no processamento de consultas, embora seja bastante difícil incorporá-los ao sistema de forma eficiente.

A *arquitetura integrada com extensões espaciais* consiste em utilizar extensões espaciais desenvolvidas sobre um SGBD-OR. Esta arquitetura oferece algumas vantagens:

- permite definir tipos de dados espaciais, equipados com operadores específicos (operadores topológicos e métricos);
- permite definir métodos de acesso específicos para dados espaciais;

Exemplos desta arquitetura são o Oracle Spatial (Murray, 2003) e PostGIS , descritos em maior detalhe no Capítulo 8 deste livro. Embora largamente baseados nas especificações do OpenGIS (OGC, 1996), estas implementações possuem variações relevantes entre os modelos de dados, semântica dos operadores espaciais e mecanismos de indexação.

As extensões espaciais utilizadas nas arquiteturas integradas possuem as seguintes características :

- fornecem tipos de dados espaciais (TDEs) em seu modelo de dados, e mecanismos para manipulá-los;
- estendem SQL para incluir operações sobre TDEs, transformando-a de fato em uma linguagem para consultas espaciais;
- adaptam outras funções de nível mais interno ao SGBD para manipular TDEs eficientemente, tais como métodos de armazenamento e acesso, e métodos de otimização de consultas.

Normalmente, essas extensões tratam somente objetos espaciais cuja componente espacial seja uma geometria vetorial, utilizando BLOBs para armazenar dados matriciais, com todos os problemas já citados.

De fato, no caso de aplicativos SIG que manipulam objetos com geometrias tanto matriciais quanto vetoriais, é possível a utilização de uma *arquitetura integrada combinada*, formada pela combinação das duas últimas. Ou seja, as geometrias vetoriais são armazenadas utilizando-se os recursos oferecidos pelas extensões e as geometrias matriciais são armazenadas em BLOBs. As funcionalidades para manipulação de geometrias matriciais são fornecidas por uma camada externa ao SGBD, de modo a complementar os recursos ausentes, até o momento, nas extensões. Um exemplo desta arquitetura, a TerraLib (Câmara et al., 2000), será discutida em detalhe nos Capítulos 12, 13 e 14.

5.4 Operações espaciais

As consultas espaciais baseiam-se em relacionamentos espaciais de vários tipos: métricos, direcionais e topológicos. Por serem dois conceitos de natureza distinta, as operações sobre as componentes espaciais de geo-campos e geo-objetos também são diferentes. Abordaremos nesta seção apenas operações sobre as geometrias vetoriais de geo-objetos. Portanto, no que se segue, omitiremos o adjetivo "vetorial". As operações espaciais podem ser classificadas em (Rigaux et al., 2002):

Operação unária booleana: mapeia geometrias em valores booleanos. Como exemplos, temos: *Convex,* que testa se uma geometria é convexa; e *Connected,* que testa se uma geometria está conectada.

Operação unária escalar: mapeia geometrias em valores escalares. Como exemplos, temos: *Length,* que computa o comprimento ou perímetro de uma geometria; e *Área,* que computa a área de uma geometria.

Operação unária espacial: mapeia geometrias em geometrias. Como exemplos, temos: *Buffer,* que retorna uma nova geometria a partir de uma distância em torno de uma geometria específica; *ConvexHull*, que retorna uma geometria convexa a partir da geometria; MBR, que retorna o mínimo retângulo envolvente de uma geometria; e *Centroid,* que retorna o centróide de uma geometria.

Operação binária booleana: também chamada de *predicado espacial* ou *relacionamento espacial*, mapeia pares de geometrias em valores booleanos. Esta classe pode ser dividida em:

Relacionamento topológico: um relacionamento que não é alterado por transformações topológicas, como translação, rotação e mudança de escala. Como exemplos, temos: contém (*contains*), disjunto (*disjoint*), intercepta (*intersects*), cruza (*crosses*), como apresentado na Seção 2.9.

Relacionamento direcional: um relacionamento que expressa uma noção de direção. Como exemplos, temos: acima de (*above*), ao norte de (*northOf*), dentre outras;

Relacionamento métrico: um relacionamento que expressa uma noção métrica. Por exemplo, o relacionamento que retorna Verdadeiro se duas geometrias estão a menos de uma determinada distância uma da outra.

Operação binária escalar: mapeia pares de geometrias em valores escalares. Por exemplo, *distance* computa a distância entre duas geometrias.

Operação binária espacial: mapeia pares de geometrias em geometrias. Como exemplos, temos as operações de conjunto, como interseção (*Intersection*), união (*Union*) e diferença (*Difference*).

Operação n-ária espacial: mapeia n-tuplas de geometrias em geometrias. Por exemplo, a operação *ConvexHull* pode pertencer a essa classe quando receber mais de uma geometria como parâmetro de entrada.

5.5 Linguagens de consulta espacial

Como SQL-89 não acomodava consultas espaciais, várias propostas surgiram na década de 90 para estender a linguagem, notadamente (Egenhofer, 1994) (OGIS, 1995). Segundo Frank e Mark (1991), as extensões devem considerar dois pontos básicos:

- embora seja possível estender SQL com operadores espaciais, a semântica destes operadores deve ser formalmente definida;
- embora seja possível estender SQL para incluir controle de apresentação de geometrias, aconselha-se projetar uma linguagem separada para lidar com esta questão.

Esta seção apresenta inicialmente uma breve classificação para consultas espaciais. Em seguida, discute algumas extensões para incluir suporte a consultas espaciais na linguagem SQL, incluindo o padrão publicado pela ISSO, chamado SQL/MM Spatial.

5.5.1 Tipos de consultas espaciais

A eficácia das estratégias de otimização de consultas espaciais depende fundamentalmente da complexidade e freqüência das consultas. Embora estes fator seja de difícil caracterização, esta seção sugere uma classificação das consultas espaciais bastante útil para o estudo do problema, seguindo (Brinkhoff et al., 1993).

Dentre os tipos de consultas, destacamos os seguintes:

seleção espacial: dado um conjunto de objetos espaciais D e um predicado de seleção espacial ρ sobre atributos espaciais dos objetos em D, determine todos os objetos em D cujas geometrias satisfazem ρ.

junção espacial: dados dois conjuntos de dados espaciais, D e D', e um predicado de seleção espacial θ, determine todos os pares $(d,d') \in D \times D'$ cujas geometrias satisfazem θ.

Identificamos ainda os seguintes casos particulares importantes de seleção espacial:

seleção por ponto: dado um ponto P e um conjunto de objetos espaciais D, determine todos os objetos em D cujas geometrias contêm P.

seleção por região: dada uma região R e um conjunto de objetos espaciais D, determine todos os objetos em D cujas geometrias estão contidos em R.

seleção por janela: dado um retângulo R com os lados paralelos aos eixos e um conjunto de objetos espaciais D, determine todos os objetos em D cujas geometrias estão contidos em R.

Um *predicado de seleção espacial* é uma expressão booleana B(x) com uma variável livre, x, varrendo geometrias, tal que B(x) envolve apenas operações espaciais. Semelhantemente, um *predicado de junção espacial J(x,y)* é uma expressão booleana com duas variáveis livres, x e y, varrendo geometrias, tal que *J(x,y)* envolve apenas operações espaciais. Um exemplo de seleção espacial seria:

S1. *Selecione as regiões da França adjacentes à região de Midi-Pirenées.*

A Figura 5.4 ilustra o resultado desta consulta, onde a região de Midi-Pirenées aparece em cinza escuro, e as regiões adjacente, em cinza claro.

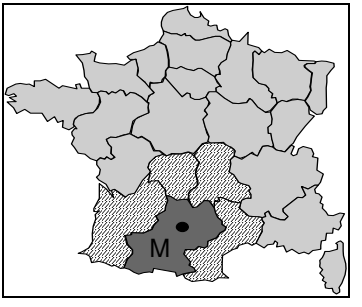

Figura 5.4 – Operação de seleção espacial.

Exemplos de junção espacial seriam:

J1. *Para cada estrada da Amazônia, selecione as reservas indígenas a menos de 5 km de uma estrada.*

J2. *Para as cidades do sertão cearense, selecione quais estão a menos de 10 km de algum açude com capacidade de mais de 50.000 m^3 de água*

5.5.2 SF-SQL

A SF-SQL, proposta pelo *OGC - Open Geoscience Consortium* , especifica um conjunto de tipos de geometrias vetoriais, operações topológicas e operações métricas. A proposta especifica ainda um esquema de tabelas para metadados das informações espaciais. Ela introduz o conceito de "tabela com feições" para representação dos dados geográficos. Nesta tabela, os atributos não espaciais são mapeados para colunas de tipos disponíveis na SQL-92, e os atributos espaciais para colunas cujo tipo de dados é baseado no conceito de "tipos de dados geométricos adicionais para SQL".

A representação dos atributos espaciais pode seguir dois modelos, chamados SQL-92 e SQL-92 com Tipos Geométricos. O primeiro modelo utiliza uma tabela para representar os atributos espaciais. O segundo utiliza tipos abstratos de dados específicos para geometrias, estendendo os tipos da SQL.

Hierarquia de tipos de geometrias da SF-SQL

A Figura 5.5 ilustra a hierarquia de tipos da SF-SQL. Este diagrama é o mesmo tanto para o modelo da SQL-92 quanto para o da SQL-92 com Tipos Geométricos.

Alguns tipos são abstratos como: *Curve*, *Surface*, *MultiSurface* e *MultiCurve*. Um tipo especial é a *GeometryCollection*, que pode ser composta por mais de um tipo de geometria (tipo heterogêneo). Os outros são tipos básicos, como *Point*, *LineString* e *Polygon*, que podem formar tipos de coleções homogêneas como *MultiPoint*, *MultiLineString* e *MultiPolygon*, respectivamente. Cada um destes tipos possui uma série de atributos, métodos e definições que são apresentadas na especificação.

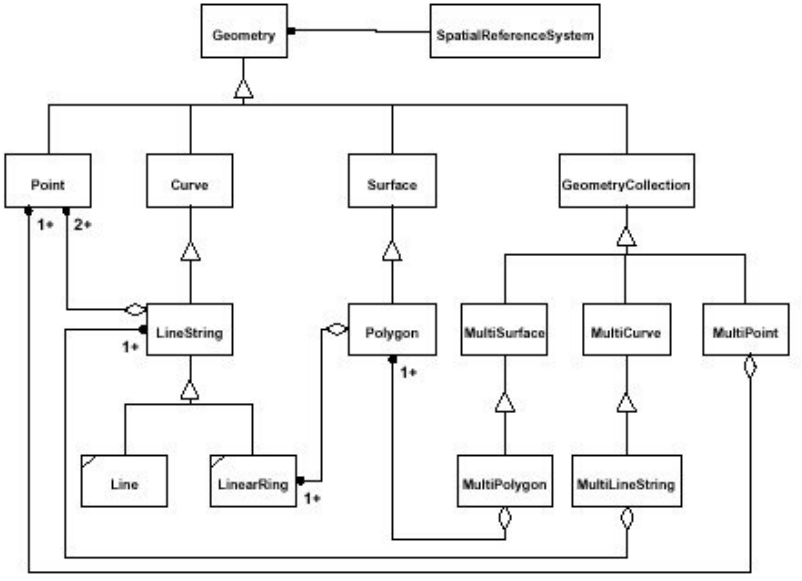

Figura 5.5 – Hierarquia de tipos de geometrias da SF-SQL .

Relacionamentos Topológicos

A SL-SQL basicamente adota os relacionamentos topológicos introduzidos na Seção 2.9, definidos como métodos do tipo *Geometry*. Há ainda um outro relacionamento, chamado *relate*, que recebe a geometria a ser comparada e um segundo parâmetro que representa um padrão da matriz de interseção, na forma de uma cadeia de nove caracteres, representando um teste para interseção entre fronteira, interior e exterior. Ele retorna o inteiro 1 (TRUE) se as geometrias forem relacionadas espacialmente de acordo com os valores especificados na matriz. Considere, por exemplo, a seguinte consulta espacial:

Q. *Selecione os municípios que fazem fronteira com o município de Belo Horizonte.*

Esta consulta pode ser expressa em SF-SQL da seguinte forma:

```
SELECT M1.name
FROM Municipio M1,Municipio M2
```

```
WHERE Touch(M1.location,M2.location)=1
AND M2.Name ='Belo Horizonte'
```

Outros Operadores Espaciais

Além dos relacionamentos topológicos, SL-SQL inclui outros métodos para o tipo *Geometry*. Entre eles, estão:

distance(outraGeometria:Geometry):Double
retorna a distância entre as geometrias

buffer(distância:Double):Geometry
retorna uma geometria definida por um mapa de distância

convexHull():Geometry
retorna um polígono convexo com todos os pontos da geometria

intersection(outraGeometria:Geometry):Geometry
retorna a geometria resultante da interseção das geometrias

union(outraGeometria:Geometry):Geometry
retorna a geometria resultante da união de duas geometrias

difference(outraGeometria:Geometry):Geometry
retorna a geometria resultante da diferença entre as geometrias

Os tipos *Surface* e *MultiSurface* possuem ainda os seguintes métodos:

area ():double
área de uma região

centroid():point
um ponto representando o centróide da geometria

pointOnSurface():point
um ponto que esteja na superfície

Tabelas com Feições

A especificação da SF-SQL propõe o esquema de metadados mostrado na Figura 5.6 para representar conjuntos de geo-objetos em tabelas com atributos dos tipos de geometrias anteriormente descritos:

SPATIAL_REF_SYS, tabela armazenando dados sobre cada sistema de referenciamento espacial (SRS) utilizado no banco de dados.

GEOMETRY_COLUMNS, tabela de metadados para as colunas geométricas das tabelas com feições *(feature tables)*.

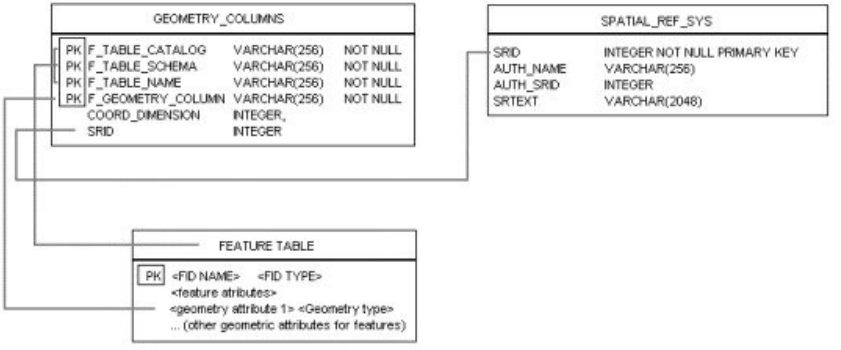

Figura 5.6 – Esquema de Metadados da SF-SQL .

Cada atributo do tipo *Geometry* (ou de seus sub-tipos), de cada tabela com feições, deve estar associado a um SRS, de tal forma que seja possível determinar o sistema de referenciamento espacial utilizado para representar cada geometria armazenada na tabela. Isso é essencial para que os métodos possam verificar a compatibilidade dos sistemas de referenciamento espacial utilizados pelas geometrias.

5.5.3 Spatial SQL

A linguagem Spatial SQL, desenvolvida por Egenhofer (Egenhofer, 1994), representa um outro exemplo de linguagem de consulta espacial baseada em SQL. Spatial SQL divide-se em duas sub-linguagens, uma para consulta e outra para apresentação de objetos espaciais, buscando aproximar-se da forma como os seres humanos conceitualizam o espaço geográfico. A sub-linguagem de consulta estende SQL com operadores e relacionamentos espaciais. Distingue-se de outras extensões por preservar os conceitos de SQL e por conseguir um tratamento de alto nível dos objetos espaciais.

Exemplos de operações disponíveis na Spatial SQL são:

- Operadores unários: fornecem, por exemplo, dados de limite, interior, comprimento, área e volume de objetos.
- Operadores binários: fornecem, por exemplo, distância e direção.
- Funções de agregação: operam sobre conjuntos de objetos, como as funções de mínimo e média.

Os relacionamentos topológicos são baseados na matriz de 9-interseções. (ver Seção 2.9) Podemos citar como exemplo: *overlap* (sobreposição), *inside* (dentro) e *cover* (cobertura).

A sub-linguagem de apresentação, chamada *Graphical Presentation Language* (GPL), baseia-se em um trabalho anterior (Egenhofer e Frank, 1988) e especifica como o resultado de uma consulta pode ser visualizado e manipulado em um ambiente gráfico. GPL permite ao usuário definir as características do ambiente gráfico, fornecendo operações e relacionamentos espaciais. Possui ainda métodos para referenciar objetos, apresentados na tela, através de apontamento.

Uma seqüência de comandos escritos em Spatial SQL, adaptada de (Egenhofer, 1994), é mostrada abaixo. Os comandos exibem um mapa de *Grove Street* em *Orono,* com suas construções, limites de terrenos e rodovias. Para a visualização, utiliza-se verde para as residências, azul para os prédios comerciais e preto para os limites de terrenos. Utiliza-se também os nomes das ruas para identificá-las.

- Definição do ambiente gráfico:

```
SET  LEGEND
       COLOR      black
       PATTERN  dashed
FOR  SELECT  boundary (geometry)
       FROM     parcel;
SET  LEGEND
       COLOR    green, blue
FOR SELECT residence.geometry, commercial.geometry
       FROM     residence.type="Residential" AND
              commercial.type="Commercial";
```

- Identificação da janela de interesse e definição da visualização:

```
SET  WINDOW
       SELECT   geometry
       FROM     road
       WHERE    town.name = "Orono";

SET CONTEXT
FOR road.geometry,
              SELECT parcel.geometry, road.name,
                     building.geometry
              FROM   road, parcel, building;
```

- Exibição de um novo mapa e recuperação dos dados:

```
SET MODE new;
SELECT road.geometry
FROM   road, town
  WHERE   town.name = "Orono"  AND
        road.name = "Gove Street" AND
        road.geometry INSIDE town.geometry
```

Este exemplo ilustra vários pontos interessantes de Spatial SQL. Além de oferecer operações espaciais, Spatial SQL separa as etapas de consulta e visualização em etapas menores, permitindo que o usuário reuse um ambiente em diversas consultas. Porém, a linguagem possui uma sintaxe complexa, principalmente para os usuários finais de SIGs, além de supor a existência de uma interface gráfica capaz de apresentar e manipular dados em GPL.

5.5.4 SQL/MM Spatial

A especificação da última versão de SQL, designada pelo nome de SQL:1999 (Melton, 2002), divide-se em várias partes. A SQL/MM (MM para *MultiMedia*) compreende extensões para tratar de texto, dados espaciais e imagem estática ou em movimento.

A especificação da SQL/MM (ISO, 2000; Melton e Eisenberg, 2001; Stolze, 2003) também se desdobra em várias partes, bastante independentes entre si. A SQL/MM Spatial define tipos e métodos para tratar geometrias no $\Re^2$. SQL/MM também modela sistemas de

referenciamento espacial (SRS). Revisões futuras tratarão de tipos de geometrias no $\Re^3$, ou mesmo em dimensões superiores.

A especificação da SQL/MM Spatial está alinhada com outros esforços de padronização para SIGs, notadamente o ISO Technical Committee, TC 211 (Geomatics) e o OGC - Open Geoscience Consortium. Em particular, a especificação segue a hierarquia de tipos geométricos definidos pelo OGC, discutida na Seção 5.6.2.

A especificação do SQL/MM consistentemente usa o prefixo "ST" para os nomes de todas as tabelas, tipos, métodos e funções, para indicar *"Spatial and Temporal"*. De fato, a intenção original da especificação era adotar um modelo de dados espaço-temporal. Porém, durante o desenvolvimento do SQL/MM, decidiu-se que a componente temporal transcendia o escopo das aplicações espaciais e que, portanto, deveria ser incorporado ao SQL:1999 em separado, formando a sub-linguagem SQL/Temporal (ISO, 2001). Porém, esta parte do SQL:1999 foi abandonada, e a proposta de padrão, retirada.

A Figura 5.7 mostra a hierarquia de tipos geométricos de SQL/MM Spatial, adaptada da hierarquia definida pela OGC, onde os tipos sombreados não são instanciáveis. A hierarquia de tipos geométricos da SQL/MM Spatial difere, porém, da hierarquia da OGC em alguns pontos:

- adota *ST_LineString* em substituição a *Line* e *LinearRing*;
- oferece arcos circulares e regiões cujas fronteiras são arcos circulares;
- omite a indicação de que tipos são agregações de outros tipos. Voltando à Figura 5.7, não é óbvio uma geometria do tipo *ST_MultiPoint* seja uma agregação de geometrias do tipo *ST_Point*.

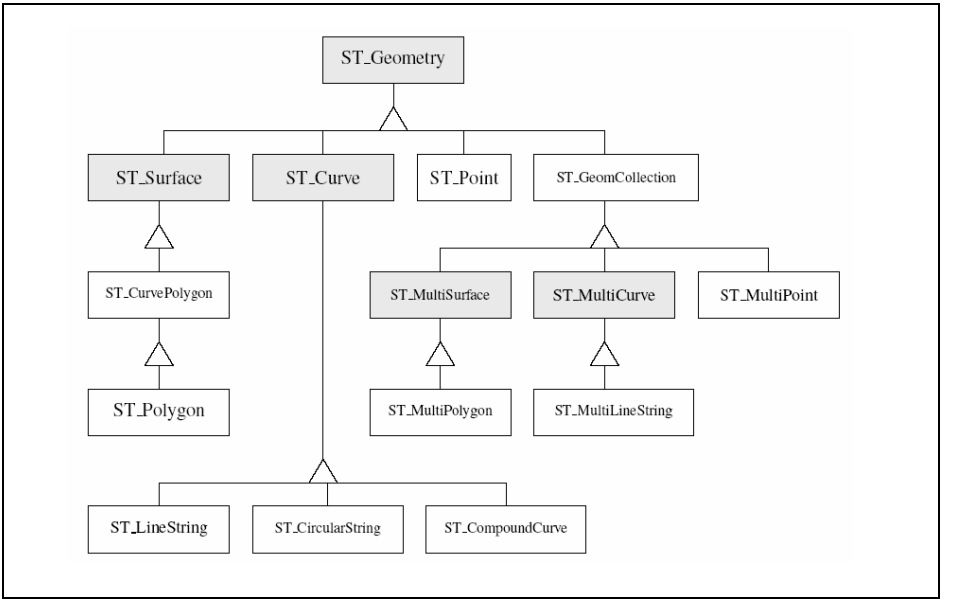

Figura 5.7 – Hierarquia de tipos do SQL/MM Spatial (Stolze, 2003).

A especificação do SQL/MM Spatial agrupa os métodos implementando as operações espaciais em quatro categorias (compare com a classificação introduzida naSeção 5.4):

- métodos convertendo geometrias para formatos de exportação e vice-versa;
- métodos computando propriedades métricas de geometrias;
- métodos implementando relacionamentos topológicos;
- métodos gerando geometrias a partir de outras.

Os tipos possuem ainda métodos que permitem extrair informações básicas sobre as suas instâncias, como as coordenadas de um ponto. Por exemplo, considere a seguinte tabela:

```
CREATE TABLE cidade (   nome VARCHAR(30),
                        populacao INTEGER,
                        localizacao ST_GEOMETRY )
```

A consulta abaixo retorna a área da Cidade de Belo Horizonte:

```
SELECT localizacao.area
FROM city
WHERE nome = 'Belo Horizonte'
```

A expressão `localizacao.area` retorna o valor do atributo `area` da instância do tipo estruturado *ST_Geometry* armazenada na coluna `localizacao` da linha da tabela `cidade` tal que a coluna `nome` possui valor `'Belo Horizonte'`.

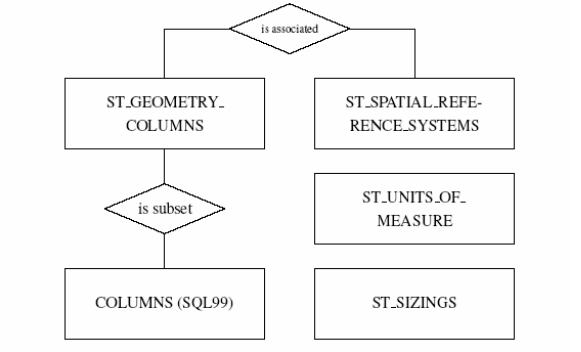

Figura 5.8 – Esquema de metadados da SQL/MM Spatial (Stolze, 2003).

A Parte 3 da especificação do SQL/MM Spatial define um esquema de metadados, semelhante ao esquema definido para o SF-SQL, ilustrado no diagrama entidade-relacionamento da Figura 5.8. O esquema de metadados compreende as seguintes tabelas:

- *ST_GEOMETRY_COLUMNS,* armazena metadados descrevendo as colunas geométricas das tabelas do banco de dados (idêntica à tabela *GEOMETRY_COLUMNS* da SF-SQL, definida pela OGC).
- *ST_SPATIAL_REFERENCE_SYSTEMS,* armazenando dados sobre cada sistema de referenciamento espacial (SRS) utilizado no banco de dados (idêntica à tabela *SPATIAL_REF_SYS* da SF-SQL).
- *ST_UNITS_OF_MEASURE,* armazena as unidades de medida utilizadas no banco de dados.

- *ST_SIZINGS,* semelhante à tabela *SIZINGS* do SQL99, define as meta-variáveis, e seus valores, específicas para a componente especial do banco de dados.

Por fim, a segunda versão do padrão da SQL/MM deverá incluir suporte à *Geography Markup Language (GML)*, definida pela OGC, e suporte a ângulos e direções.

5.6 Direções de pesquisa

5.6.1 Consultas espaço-temporais

Como o nome indica, *consultas espaço-temporais* tratam de dados convencionais, espaciais e temporais. Tais consultas recuperam dados a respeito do estado, da história e da evolução dos objetos ao longo do tempo. Para exemplificar os dados temporais, utilizaremos o monitoramento do desmatamento da Amazônia realizado periodicamente pelo PRODES (Sistema de Detecção de Desmatamentos) e mostrado em Correia et al. (2005).

Como exemplos dessa evolução temporal, temos áreas que em um determinado momento eram parte da floresta e algum tempo depois foram desmatadas. Essas áreas podem alterar seu tamanho ou sua forma em outras observações. Existe também a possibilidade de uma área desmatada ser replantada ou sofrer outros tipos de regeneração.

Um banco de dados espaço-temporal que modela os dados de desmatamento poderia conter várias imagens semelhantes, e os dados relativos a essas imagens. Aliando esses dados a informações, como reservas indígenas ou hidrografia, pode-se pensar em consultas que incluem a dimensão espacial como:

a) *Quais são as áreas de reservas indígenas próximas a áreas desmatadas?*

b) *Quais são as áreas de preservação de hidrografia que possuem desmatamentos?*

Quando incluímos a dimensão temporal e buscamos combiná-la com a dimensão espacial, as consultas se tornam mais complexas. Podemos

pensar em consultas que tratem a evolução ao longo do tempo dos objetos envolvidos, como:

a) *Quais são as áreas desmatadas no período de 01-01-1990 a 31-12-1999 e posteriormente recuperadas?*

b) *Quais são as áreas desmatadas no último bimestre que ficam a menos de 50 Km de distância de estradas ou rios?*

Na primeira consulta nos deparamos com o desafio de definir as áreas que eram florestas e foram desmatadas na década de 90. Dentro dessas áreas, procuramos quais evoluíram e deixaram de ser áreas desmatadas. Na segunda consulta também nos deparamos com outro desafio, o de definir quais áreas foram desmatadas e como essas áreas se relacionam com estradas e rios próximos.

Essas duas consultas ilustram questões complexas de serem modeladas pelas linguagens disponíveis, e mostram que as linguagens de consulta espaço-temporais ainda têm muitos desafios a superar.

5.6.2 Álgebra para objetos móveis

Uma álgebra para *objetos móveis*, ou seja, objetos que se movem continuamente ao longo do tempo, é apresentada em Güting et al (2003). Essa álgebra foi implementada no ambiente SECONDO, um SGBD extensível e modular, que permite a utilização de diversas álgebras (Guting et al., 2004).

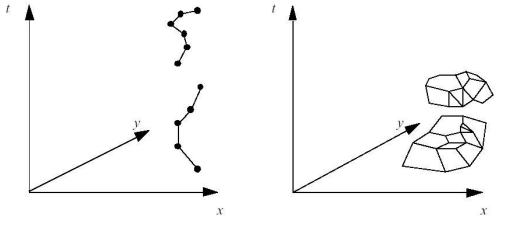

Figura 5.4 – Exemplos de representações de *mpoint* e *mregion* (Fonte: adaptada de Guting et al., 2004).

Dois tipos básicos de objetos móveis são (ver Figura 5.4):

- *moving point (mpoint)*: descreve um objeto cuja relevância está nas posições ocupadas no espaço ao longo do tempo. Um exemplo de um *mpoint* é o deslocamento de um veículo.
- *moving region (mregion):* descreve uma área que possui extensão e se modifica, por exemplo, crescendo ou se movimentando. Um exemplo pode ser o movimento de uma tempestade.

Operações podem ser definidas envolvendo tipos convencionais e os tipos básicos de objetos móveis. Por exemplo, temos:

- *intersection: mpoint* × *mregion* → *mpoint*

  retorna um *mpoint* formado pela parte de um *mpoint* que estiver dentro de uma *mregion*.

- *trajectory: mpoint* → *line*

  projeta um *mpoint* no plano, retornando uma *line,* definida como uma curva em espaço bidimensional.

- *deftime: mpoint* → *periods*

  retorna um intervalo de tempo, do tipo periods, definido por determinado mpoint.

Por fim, considere as seguintes tabelas:

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

```
weather (id:string,kind:string,area:mregion)
```

Exemplos de consultas sobre estas tabelas são:

Q1. *Selecione todos os vôos partindo de Düsseldorf que percorrem mais de 5000 Km*

```
SELECT id
FROM flights
WHERE from = "DUS" AND
   length(trajectory(route)) > 5000
```

Q2. *Que horas o vôo BA488 atravessou a tempestade de neve de identificador S16?*

```
SELECT deftime(intersection(f.route,w.area))
```

```
FROM flights as f, weather as w
WHERE f.id = "BA488" AND w.id = "S16"
```

5.7 Leituras suplementares

Há inúmeros livros-texto sobre bancos de dados e a linguagem SQL. Recomenda-se, em especial, os textos sobre SQL:1999 (Melton e Simon, 2001) (Melton, 2002). O primeiro trata dos conceitos relacionais básicos da linguagem, enquanto que o segundo aborda as extensões objeto-relacionais, incluindo suporte a consultas espaciais.

Um pouco da história do desenvolvimento de bancos de dados espaciais pode ser avaliada através de referências básicas sobre a área, como Güting (1994) e Medeiros (1994). Recomenda-se também a leitura dos livros-texto de Rigaux e Voisard (2001) e Shekhar (2002).

Conforme discutido na Seção 5.3, a arquitetura integrada com extensões espaciais mostra-se promissora. O Oracle Spatial (Murray, 2003), o DB2 Spatial Extender (Adler, 2001) e o PostGIS são exemplos desta arquitetura, que merecem um estudo detalhado. O documento descrevendo a arquitetura de referência do *Open Geoscience Consortium* (Percivall, 2003) apresenta uma abordagem para o problema de interoperabilidade em SIGs, assunto do Capítulo 10.

A definição de relacionamentos topológicos foi extensamente estudada, incluindo a matriz de 4-interseções e suas variantes (Egenhofer e Franzosa, 1995), a matriz de 9-Interseções (Egenhofer et al., 1994), a matriz de 9-Interseções estendida dimensionalmente (Paiva, 1998) e o trabalho de Clementini et al. (1993) adotado pela SF-SQL e SQL/MM Spatial.

Quanto a linguagens de consulta espacial, Stolze (2003) apresenta uma análise sucinta, porém criteriosa, do padrão SQL/MM Spatial.

Além dos tópicos abordados acima, há linguagens de consulta endereçando outros tipos de dados geográficos, ou ambientes com necessidades específicas. Como exemplo, podemos citar linguagens de consulta para redes georeferenciadas, como a rede de distribuição de água de uma cidade, onde a noção de espaço Euclidiano não é adequada (Papadias et al., 2003). Há também toda uma linha de pesquisa sobre

consultas dependentes de localização em ambientes para computação móvel (Ayse et al., 2001) (Zhang et al., 2003) (Manical et al., 2004).

Referências

ADLER, D.W. DB2 Spatial Extender - Spatial data within the RDBMS. 27th International Conference on Very Large Data Bases, 2001. Morgan Kaufmann Publishers Inc. p. 687-690

AYSE, Y.; DUNHAM, H. M.; KUMAR, V. M. Location dependent query processing. In: 2nd ACM international workshop on Data engineering for wireless and mobile access. Santa Barbara, CA, USA, 2001. p. 47-53.

BRINKHOFF, T.; HORN, H.; KRIEGEL, H.-P.; SCHNEIDER, H. A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems. In: Third International Symposium on Advances in Spatial Databases. **Lecture Notes In Computer Science 692**. Springer-Verlag London, UK, 1993. p. 357-376.

CÂMARA, G.; SOUZA, R.; PEDROSA, B.; VINHAS, L.; MONTEIRO, A. M.; PAIVA, J.; CARVALHO, M. T.; GATTASS, M. TerraLib: Technology in Support of GIS Innovation. In: II Brazilian Symposium on Geoinformatics, GeoInfo2000. São Paulo, 2000. p.

CLEMENTINI, E.; DI FELICE, P.; VAN OOSTEROM, P., 1993. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In: ABEL, D.; OOI, B. C., eds., **SSD '93: Lecture Notes in Computer Science, v. 692**: New York, NY, Springer-Verlag, p. 277-295.

CORREIA, A. H.; PIROMAL, R. A. S.; QUEIROZ, G. R.; SOUZA, R. C. M., 2005, Modelagem de um banco de dados espaço-temporal para desmatamento e queimadas, XII Simpósio Brasileiro de Sensoriamento Remoto, Goiânia.

EGENHOFER, M. Spatial SQL: A Query and Presentation Language. **IEEE Transactions on Knowledge and Data Engineering**, v. 6, n.1, p. 86-95, 1994.

EGENHOFER, M.; FRANK, A. Towards a Spatial Query Language: User Interface Considerations. In: 14th International Conference on Very Large Data Bases. Los Angeles, CA, 1988. p. 124-133.

EGENHOFER, M.; P. DI FELICE; CLEMENTINI, E. Topological Relations between Regions with Holes. **International Journal of Geographical Information Systems**, v. 8, n.2, p. 129-144, 1994.

EGENHOFER, M.; FRANZOSA, R. On the Equivalence of Topological Relations. **International Journal of Geographical Information Systems**, v. 9, n.2, p. 133-152, 1995.

FRANK, A.; MARK, D., 1991. Language Issues for GIS. In: MAGUIRE, D.; GOODCHILD, M.; RHIND, D., eds., **Geographical Information Systems, Volume 1: Principles**: London, Longman, p. 147-163.

GÜTING, R. An Introduction to Spatial Database Systems. **VLDB Journal**, v. 3, n.4, p. 357-399, 1994.

GÜTING, R. H.; BEHR, T.; ALMEIDA, V. T. D.; DING, Z.; HOFFMANN, F.; SPIEKERMANN, M., 2004, SECONDO: An Extensible DBMS Architecture and Prototype, Fernuniversität Hagen.

GÜTING, R. H.; BOHLEN, M. H.; ERWIG, M.; JENSEN, C. S.; LORENTZOS, N.; NARDELLI, E.; SCHNEIDER, M.; VIQUEIRA, J. R. R., 2003. Spatio-temporal Models and Languages: An Approach Based on Data Types. In: KOUBARAKIS et al. ed., **Spatio-Temporal Databases**: Berlin, Springer.

ISO, 2001, Information Technology – Database Languages – SQL – Part 7: Temporal (SQL/Foundation), International Organization for Standardization.

MANICAL, H.; CAMARGO, S. M.; CIFERRI, A. D. C.; CIFERRI, R. R. Processamento de Consultas Espaciais Baseado em Cache Semântico Dependente de Localização. In: VI Simpósio Brasileiro de Geoinformática – GeoInfo 2004. Campos de Jordão, SP, 2004.

MEDEIROS, C. B.; PIRES, F. Databases for GIS. **ACM SIGMOD Record**, v. 23, n.1, p. 107-115, 1994.

MELTON, J.; SIMON, A. **SQL: 1999 Understanding Relational Language Components**. Morgan Kaufmann, 2001.

MELTON, J. **Advanced SQL 1999: Understanding Object-Relational, and Other Advanced Features**. New York, NY, USA: Elsevier Science Inc., 2002.

MELTON, J.; EISENBERG, A. SQL Multimedia and Application Packages (SQL/MM). **SIGMOD Record**, 2001.

MURRAY, C., 2003, **Oracle® Spatial User's Guide and Reference 10g Release 1 (10.1)**, Redwood City, Oracle Corporation, p. 602.

OGC, ed., 1996, **The OpenGIS® Guide - Introduction to Interoperable Geoprocessing**. Boston, Open GIS Consortium, Inc.

OGIS, 1995, **OpenGIS® simple features specification for SQL revision 1.1.**

PAIVA, J. A. C. Topological Equivalence and Similarity in Multi-Representation Geographic Database. University of Maine,1998.

PAPADIAS, D.; ZHANG, J.; MAMOULIS, N.; Y., T. Query Processing in Spatial Network Databases. In: 29th Very Large Data Base Conference. Berlin, Germany, 2003.

PERCIVALL, G., 2003, **OpenGIS® Reference Model, Open Geoscience Consortium**.

RIGAUX, P.; SCHOLL, M.; VOISARD, A. **Spatial Databases with Application to GIS**. San Francisco: Morgan Kaufmann, 2002.

SCHNEIDER, M. **Spatial data types for database systems.** Berlin Hidelberg: Springer-Verlag, 1997.

SHEKHAR, S.; CHAWLA, S. **Spatial Databases: A Tour.** New York: Prentice-Hall, 2002.

STOLZE, K. SQL/MM Spatial: The Standard to Manage Spatial Data in Relational Database Systems. In: BTW 2003, Datenbanksysteme für Business, Technologie und Web. Leipzig, Alemanha, 2003. p. 247-264.

STONEBRAKER, M. **Object-relational DBMSs: the next great wave.** San Francisco: Morgan Kaufmann, 1996.

ZHANG, J.; ZHU, M.; PAPADIAS, D.; TAO, Y.; LEE, D. L. Location-based spatial queries. In: 2003 ACM SIGMOD International Conference on Management of Data. San Diego, CA, USA, 2003. p. 443-454.