CURSO DE AUXILIAR EM GEOPROCESSAMENTO

BANCO DE DADOS GEOGRÁFICOS

Sergio Sousa Costa

NOVOS CAMINHOS

INSTITUTO FEDERAL
Maranhão

Este Caderno foi elaborado pelo Instituto Federal de Educação, Ciência e Tecnologia do Maranhão – Campus São Luís Monte Castelo

Cláudio Leão Torres
Diretoria Geral

Chefe do Departamento de Educação a Distância
Eveline de Jesus Viana Sá

Coordenação de Curso
Ana Maria dos Santos

Professor Autor
Sergio Souza Costa

Suporte Pedagógico
Elisiane Araújo dos Santos Frazão

Projeto Gráfico
Heloisa Ferreira de Sousa Bastos

Design Instrucional
Eveline de Jesus Viana Sá

APRESENTAÇÃO DA DISCIPLINA

Bem-vindos (as),

Vivemos um período de grande disponibilidade de informações sobre os mais distintos tópicos, mas a nossa finitude exige que sejamos seletivos quanto ao conteúdo que devemos nos aprofundar. Então, por que deveríamos estudar banco de dados geográficos?

Nesse texto, assumo que vocês já reconhecem a importância do uso dos sistemas de informações geográficas (SIGs) como suporte a decisões e à compreensão de fenômenos que ocorrem no espaço e no tempo. Logo, uma resposta curta seria que devemos estudar banco de dados geográficos por ele ser um componente elementar desses sistemas.

Essa é uma resposta válida e direta, mas alguém poderia argumentar que banco de dados geográficos não é obrigatório em alguns sistemas. Por exemplo, no QGis ou no ArcGIS é possível manipular dados diretamente através de arquivos, como o *Shapefile* ou KML.

Essa pessoa estaria parcialmente certa, mas talvez ela não conheça as limitações dessa abordagem, ou não saiba que ao migrar do formato *Shapefile* para o *Geopackage*, ela já estaria usando um banco de dados. Com isso, ela também não conhece as vantagens do *geopackage* com relação ao *shapefile* e muito menos suas limitações com relação ao PostGIS.

Conhecendo os conceitos de banco de dados e dos sistemas gerenciadores de banco de dados, um usuário de SIG poderá decidir sobre como e onde irá manter seus dados. A solução escolhida para a produção de um simples mapa para ilustrar um artigo será muito diferente daquela usada em um grande projeto com múltiplos usuários. Além disso, o conhecimento da linguagem de consulta de banco de dados lhe dará poderes para executar análises e extrair informações mais rápido e eficientemente. Em alguns casos, sem a necessidade de um sistema de informação geográfica. Isso porque os bancos de dados geográficos já possuem diversas operações de consulta e análise espacial.

Esse texto tem o objetivo de preparar você para usar melhor os seus dados geográficos. Ele tem como foco os usuários iniciantes de sistemas de informação geográfica, mas poderá ser de grande utilidade para todos aqueles que ainda não se sentem confiantes na utilização de banco de dados geográficos.

Bom aprendizado a todos(as) e que esse fascículo contribua para o seu desenvolvimento!

Prof. Sérgio Souza Costa

Então, que tal começarmos? :)

SUMÁRIO

LISTA DE ÍCONES

1. **Atenção** (Chama a atenção para aspectos que merecem maior atenção para a compreensão do conteúdo).

2. **Você sabia?** (Contextualiza o conteúdo estudado a prática profissional);

3. **Saiba mais** (Disponibiliza referências complementares para aprofundamento de estudos (livros, sites, artigos, links, etc.);

4. **Sistematizando o conteúdo** (Apresenta uma atividade de aprendizagem para que o aluno realize e avalie o seu nível de compreensão do conteúdo).

Introdução a Banco de Dados

1.1 Conceitos básicos

Antes de tratar sobre os bancos de dados geográficos é necessário começar por algumas definições importantes. Começando com a distinção entre dados, banco de dados e sistema gerenciadores de banco de dados. Elmasri e Ramez (2005) definem um banco de dados como uma coleção de dados relacionados, em que os dados são fatos conhecidos que podem ser registrados e possuem significado implícito. Poder ser registrado significa que pode ser representado por estruturas computacionais, como números, datas e textos. Na definição, o autor também destaca que os dados precisam ser relacionados em um banco de dados. E é essa relação que permite extrair informações através de consultas que combinam dados, como aqueles sobre turmas, docentes e discentes em um sistema acadêmico.

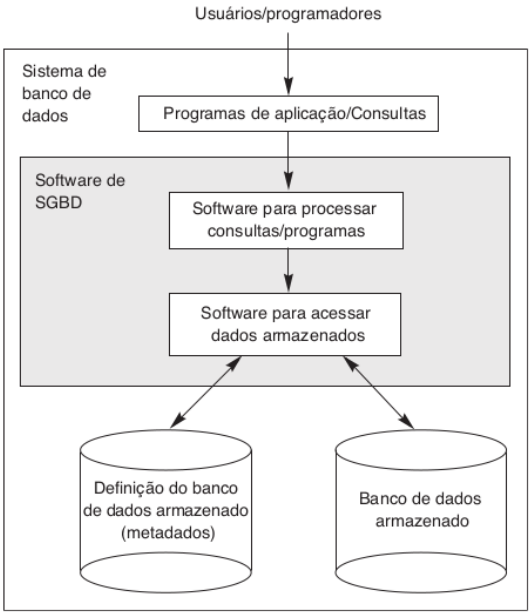
Até a década de 70, era comum que o sistema de informação fosse responsável por gerenciar sua base de dados, ou seja, possuir rotinas de armazenamento e recuperação. Porém, existem vários desafios para realizar essas rotinas de modo seguro e eficiente. Desenvolver essas rotinas exigiam muito tempo do desenvolvimento que poderia ser direcionado a um problema específico de uma dada empresa. Como armazenar e recuperar dados era uma necessidade comum, logo surgiram sistemas específicos para essas tarefas denominados de sistemas gerenciadores de banco de dados (SGBD).

Elmasri e Ramez (2005) conceituam um sistema gerenciador de banco de dados como uma coleção de programas que permite aos usuários criar e manter um banco de dados. Ele facilita o processo de definição, construção, manipulação e compartilhamento de bancos de dados entre diversos usuários e aplicações. Outras funções e vantagens de usar um sistema gerenciador de banco de dados incluem:

- Restringir acesso não autorizado, como são usados em ambiente multiusuário, o SGBD funciona geralmente como servidor e permite criar diferentes níveis de usuário e acesso.
- Tornar as consultas mais eficientes, pois por serem especialistas esses sistemas tendem a possuir todo um conhecimento acumulado através de algoritmos eficientes para armazenar e consultar dados.
- Backup e recuperação, os sistemas normalmente possuem utilitários que ajudam no processo de criar e recuperar backup.
- Controlar a redundância, pois permitem relacionar os dados de modo a evitar esse problema.
- Representar relacionamentos complexos inclui diversos tipos de relacionamentos.
- Impor restrições de integridade, após definido um esquema com os relacionamentos, pois esses sistemas têm mecanismos que garantem a integridade dos dados.

Um sistema de banco de dados é ilustrado na Figura 1, onde Elmasri e Ramez (2005) destacam os diferentes programas desses sistemas e diferenciam o banco de dados da sua definição. A definição de um banco de dados pode ser denominada como esquema ou metadados. É esse conceito que define as relações e restrições de integridade entre os dados.

Figura 1: Diagrama simplificado de um ambiente de sistema de banco de dados.

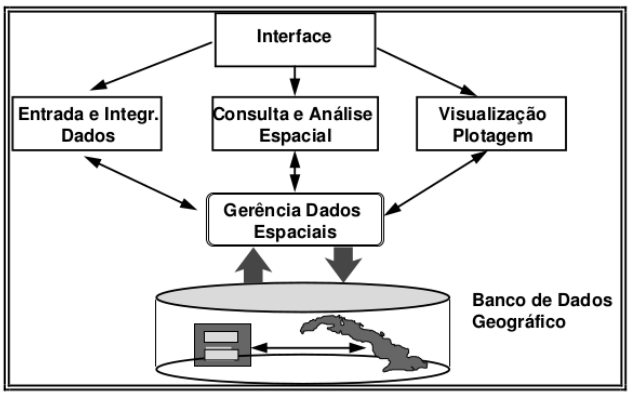


Fonte: (ELMASRI e RAMEZ, 2005)

Alguns gerenciadores de banco de dados dão suporte à manipulação de dados espaciais, sendo referidos como banco de dados espaciais (ou geográficos). Dessa forma, um banco de dados espacial é um sistema que oferece tipos de dados espaciais, uma linguagem de consulta que suporta operações espaciais e fornece métodos de indexação espacial (GUTING, 1994).

O banco de dados geográficos é um importante componente dos sistemas de informação geográfica, são os sistemas que realizam o tratamento computacional de dados geográficos, sendo um sistema imprescindível para as atividades de geoprocessamento (CÂMARA, 2005). Na Figura 2, é destacado o banco de dados geográfico como o componente base para a arquitetura de um sistema de informação geográfica.

Figura 2: Arquitetura de sistemas de informação geográfica.



Fonte: (CÂMARA, 2005)

Atenção!

O Capítulo 4 irá apresentar o sistema de informação geográfica denominado de QGIS, que será usado neste fascículo. Então, logo mais estaremos praticando esse conceito.

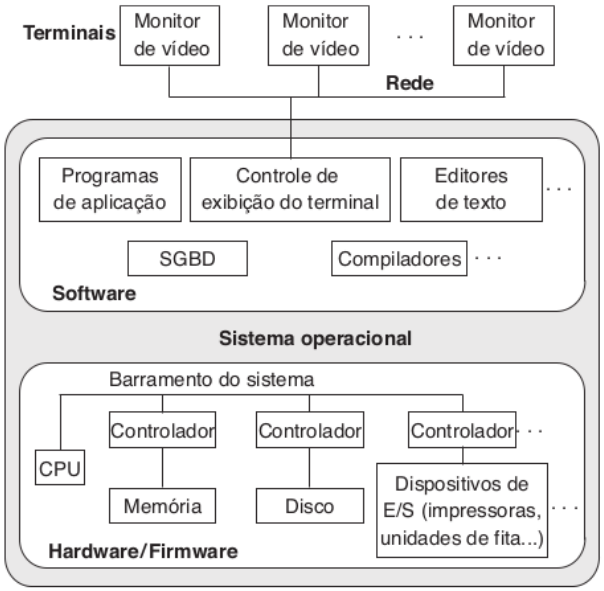
Você sabia??

Para quem já trabalha com geoprocessamento, vale destacar que uma pasta organizada com seus arquivos *shapefiles* não define um banco de dados geográficos.

1.1.1 Arquitetura dos gerenciadores de bancos de dados

Os gerenciadores de banco de dados, como outros sistemas, foram implementados inicialmente através de uma arquitetura centralizada, onde todo o processamento ocorria em grandes computadores denominados de *Mainframes*. Nela, os usuários tinham acesso a esses sistemas através de terminais burros, que eram apenas dispositivos de entrada e saída de dados equipados como monitores e teclados conectados ao computador central, como ilustrado na Figura 3.

Figura 3: Uma arquitetura física centralizada.

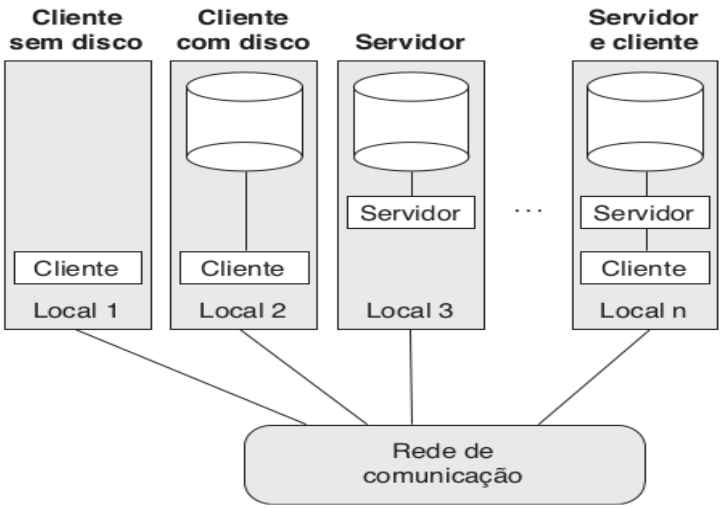


Fonte: (ELMASRI e RAMEZ, 2005)

Com a redução dos preços, os terminais foram substituídos por computadores que agora eram capazes de realizar parte do processamento. Desse modo, foi possível migrar para a arquitetura cliente/servidor. Essa é a arquitetura utilizada pelos atuais sistemas gerenciadores de banco de dados como o PostgreSQL, que iremos usar neste fascículo. Essa arquitetura não é exclusiva para gerenciadores de banco de dados, mas é usada em diversos outros sistemas que usamos atualmente, como a web, correio eletrônico, impressoras, sincronização de arquivos entre outros. Quando digitamos um endereço em um navegador, que é uma aplicação cliente, é enviado uma requisição a um recurso em um computador servidor que está localizado naquele endereço. O mesmo ocorre com os gerenciadores de banco de dados. Nos Capítulos 4 e 5 serão usadas aplicações clientes como o PGAdmin ou o QGIS para acessar os dados armazenados em um servidor de banco de dados, no caso, no PostgreSQL. Note que nessa arquitetura não é obrigatório que o cliente e o servidor estejam em computadores distintos. Nesse fascículo, serão apresentados alguns tutoriais onde o cliente e servidor são instalados no mesmo computador. Porém, em

ambientes institucionais e corporativos é mais comum que exista um computador que irá centralizar os dados para serem acessados por diversos usuários. A Figura 4 destaca essas várias possibilidades, como ter o cliente e servidor tanto em computadores distintos quanto no mesmo computador.

Figura 4: Arquitetura cliente/servidor física em duas camadas.

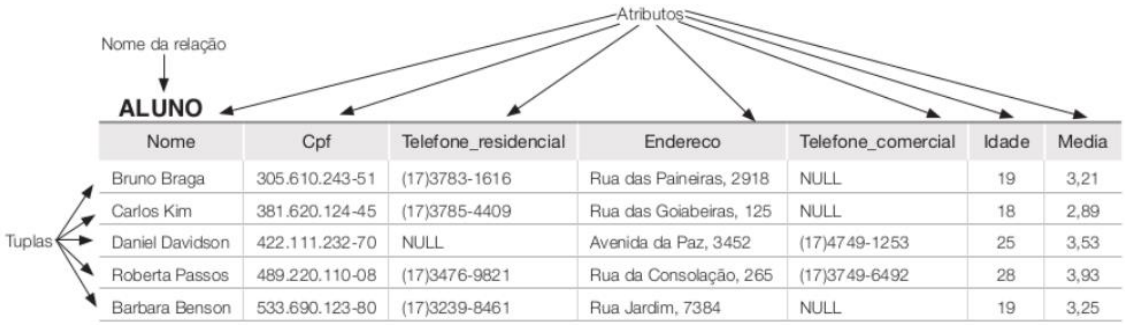


Fonte: (ELMASRI e RAMEZ, 2005)

1.1.2 Modelos de banco de dados

Além de classificar os gerenciadores de banco de dados pela arquitetura, é comum usar como base a forma que eles organizam a informação. Os primeiros SGBDs organizavam os dados geralmente em forma de árvores (modelo hierárquico) ou estruturas de rede (arestas e vértices). Atualmente, o modelo mais utilizado é o relacional (SGBR-R), que surgiu com base na álgebra relacional proposta pelo pesquisador da IBM Ted Codd, onde ele propôs conceitos como relações, tuplas e atributos. Nesse modelo, um banco é representado como uma coleção de tabelas, denominadas também de relações, como ilustrado na Figura 5.

Figura 5. Conceitos principais do modelo relacional.

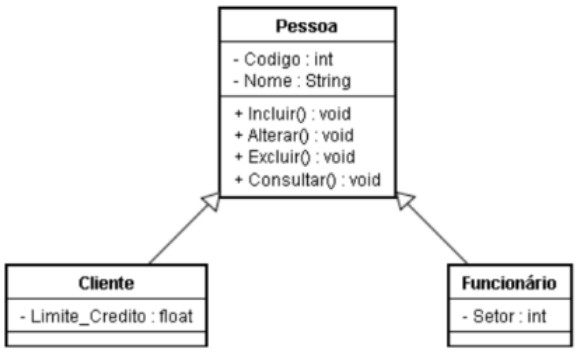


Nome	Cpf	Telefone_residencial	Endereco	Telefone_comercial	Idade	Media
Bruno Braga	305.610.243-51	(17)3783-1616	Rua das Paineiras, 2918	NULL	19	3,21
Carlos Kim	381.620.124-45	(17)3785-4409	Rua das Goiabeiras, 125	NULL	18	2,89
Daniel Davidson	422.111.232-70	NULL	Avenida da Paz, 3452	(17)4749-1253	25	3,53
Roberta Passos	489.220.110-08	(17)3476-9821	Rua da Consolação, 265	(17)3749-6492	28	3,93
Barbara Benson	533.690.123-80	(17)3239-8461	Rua Jardim, 7384	NULL	19	3,25

Fonte: (ELMASRI e RAMEZ, 2005)

Com a popularização das linguagens orientadas a objetos, esperava-se que os gerenciadores de banco de dados suportassem os conceitos desse paradigma, como herança, composição e agregação. Elmasri e Ramez (2005) definem os bancos de dados orientados a objetos como aquele que lida com objetos, suas propriedades e operações. Os objetos com a mesma estrutura e comportamento pertencem a uma classe, e as classes são organizadas em hierarquias. As operações de cada classe são especificadas com procedimentos pré definidos, chamados métodos. A Figura 6 ilustra um exemplo de três classes, sendo que as classes Cliente e Funcionário são subclasses de Pessoa.

Figura 6. Exemplo de um diagrama de classe.



Fonte: AUTOR (2021)

Os sistemas de banco de dados orientados a objetos não se tornaram populares, porém eles influenciaram os sistemas conhecidos como objeto-relacional

(SGBR-OR) ou relacional estendido. Essa nova família de sistemas permite a criação de tipos de dados complexos, que vão além dos tipos convencionais como texto, número e datas, mantendo a compatibilidade do modelo relacional e, sendo assim, muito mais fácil a migração. Um exemplo de sistema gerenciador de banco de dados objeto-relacional é o PostgreSQL que será usado neste fascículo.

Os bancos de dados relacionais e os objeto-relacionais são predominantes, mas atualmente têm surgido algumas alternativas a esse modelo que estão sendo agrupados com a denominação de banco de dados não relacional, ou NoSQL. Esses sistemas agrupam as informações baseados em documentos, grafos, chave valor, triplas entre outros modelos. Inclusive, alguns deles possuem algum suporte para armazenamento e consulta de dados espaciais. Um exemplo é o mongoDB[1] que permite armazenar dados no formato GeoJSON, como no fragmento a seguir:

```
{
 "type": "Feature",
 "geometry": {
  "type": "Point",
  "coordinates": [-44.3090446,-2.5583674]
 },
 "properties": {
  "name": "Universidade Federal do Maranhão"
 }
}
```

Na Seção 1.2 é apresentado um tutorial da instalação do PostgreSQL e a extensão espacial denominada de PostGIS.

1.2 Instalando o PostgreSQL/PostGIS

O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional, que é executado sob uma arquitetura cliente-servidor. Ele surgiu como

[1] https://www.mongodb.com/

parte do projeto POSTGRES, escrito pela Universidade da Califórnia em Berkely de 1986. Passou por várias evoluções até ter o atual nome em 1996, quando passou a ser um projeto Open Source coordenado pela PostgreSQL Global Development Group. Atualmente, é considerado o SGBD de código aberto mais avançado[2]. Ele é formado por uma coleção de programas, incluindo o servidor e alguns programas utilitários. É importante destacar o PG Admin 3, que é usado para administrar o servidor e também como cliente, para acessar e gerenciar os dados.

Atenção!

Não deixem de ler a Seção 1.1, onde são apresentados vários conceitos importantes como o de banco de dados, sistemas gerenciadores de banco de dados, arquitetura cliente-servidor e modelo objeto-relacional.

Para conhecer melhor, é importante instalá-lo para poder explorar seus recursos. Então, a seguir será apresentado um tutorial de instalação deste SGBD. A versão utilizada no tutorial é a 9.5 e pode ser baixada diretamente pelo site:

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

Nesse tutorial, vamos usar a versão 9.5, por ela ser ainda compatível com o PG Admin 3. As versões mais recentes vêm com o PG Admin 4 que tem uma interface um pouco diferente e funciona como uma aplicação Web. Porém, a versão 3 é a que possui maior quantidade de materiais na Web além de possíveis dificuldades para a instalação das versões mais recentes do PostgreSQL em versões do Windows mais desatualizadas. Para esse tutorial, usou-se a versão 9.5.24. Após baixar e executar o programa, será aberta tela inicial da instalação como na Figura 7. A partir

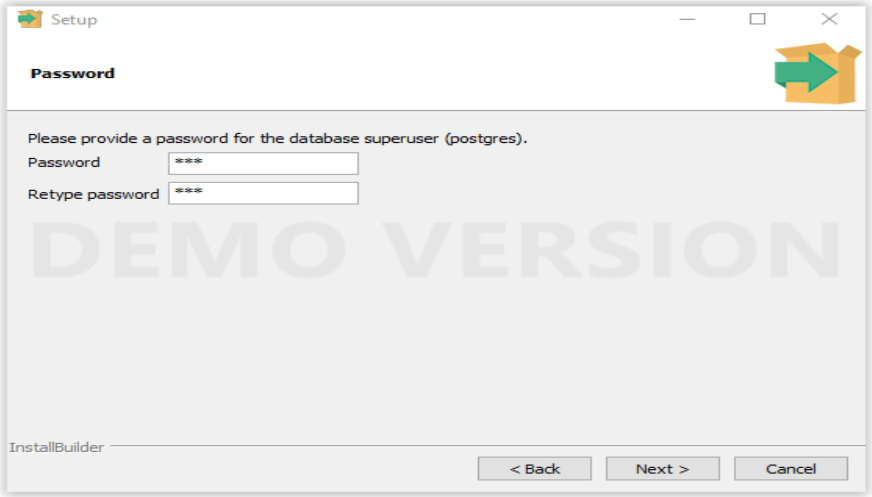
[2] https://www.postgresql.org/docs/current/history.html

daí a instalação segue de modo similar a outros instaladores de programas para o sistema operacional Windows.

Figura 7: Tela inicial do instalador do PostgreSQL

Nas telas seguintes, podem usar os diretórios indicados para o instalador e para onde serão armazenados os dados. Depois de selecionados os diretórios, será necessário definir a senha do usuário administrador, como na Figura 8. Em ambientes institucionais e corporativos deve-se usar uma senha forte, mas para uso pessoal, pode-se usar uma senha simples e fácil de lembrar e digitar.

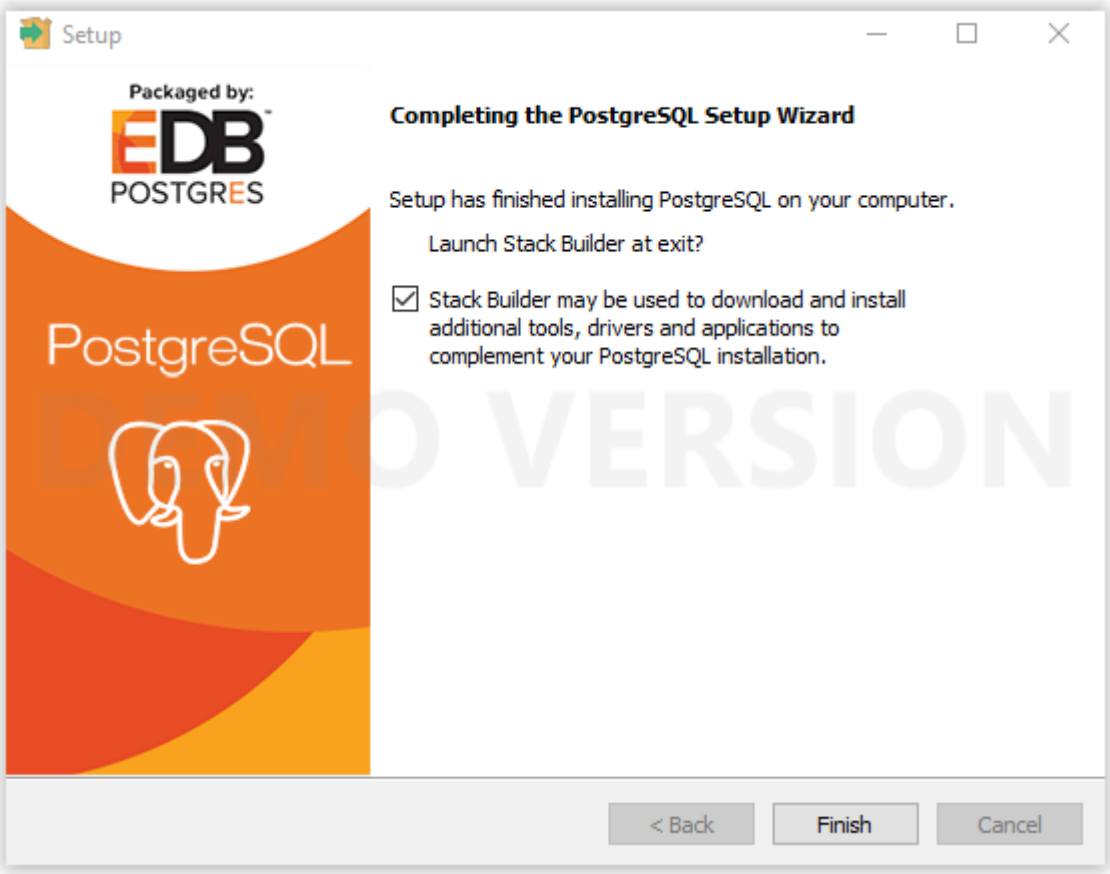
Figura 8. Tela para entrada da senha do administrador.



Como o PostgreSQL usa a arquitetura cliente/servidor, é necessário definir uma porta de acesso. Em um mesmo computador, é possível ter vários servidores em execução, então a porta é um número que permite especificar com qual servidor um dado cliente deseja se comunicar. A porta padrão do PostgreSQL é 5432, então na próxima tela é desejável manter esse valor.

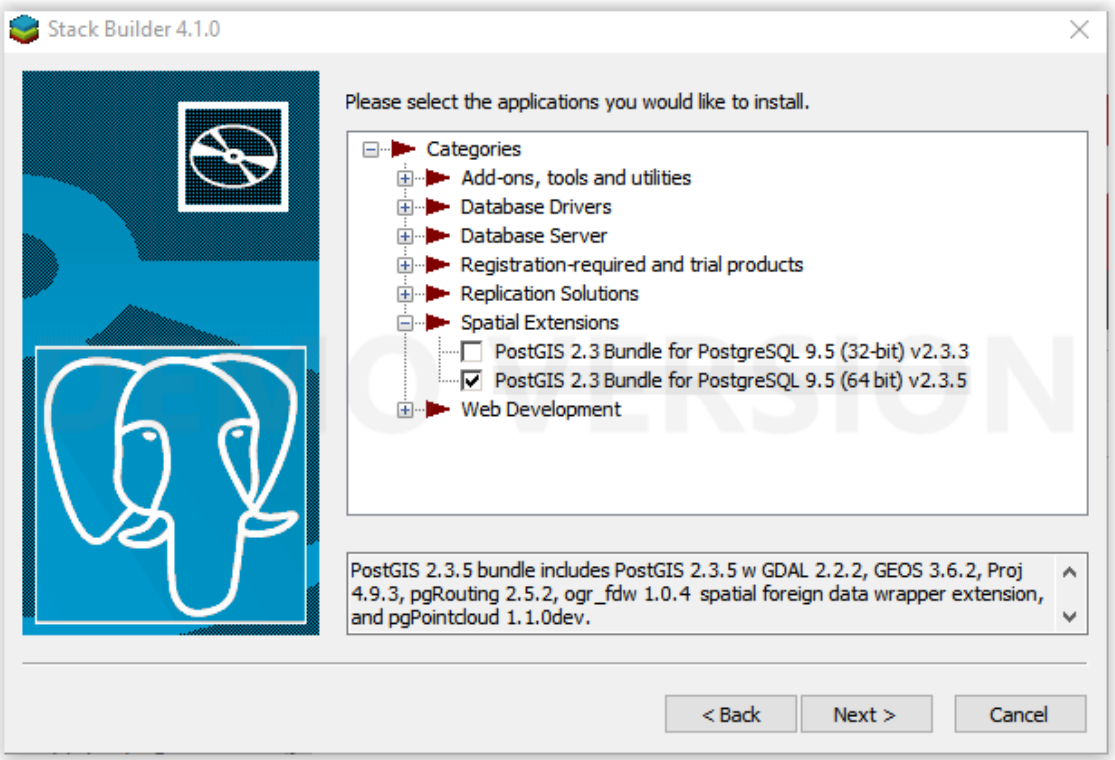
Para as próximas telas, basta clicar em “Next”. Após a instalação, aparecerá a tela de conclusão da instalação como na Figura 9.

Figura 9. Tela de conclusão da instalação do PostgreSQL



Na tela de conclusão, mantenha selecionado a opção “Stack Builder”. Essa opção iniciará a instalação dos adicionais, incluindo o PostGIS. Para este tutorial, será preciso selecionar apenas o PostGIS como na Figura 10.

Figura 10. Tela para seleção do PostGIS



Para concluir a instalação, basta selecionar “Next” nas próximas telas, então terá os programas PostgreSQL, PgAdmin III, PostGIS e “PostGIS Shapefile importer” instalados e prontos para serem usados nos próximos capítulos.

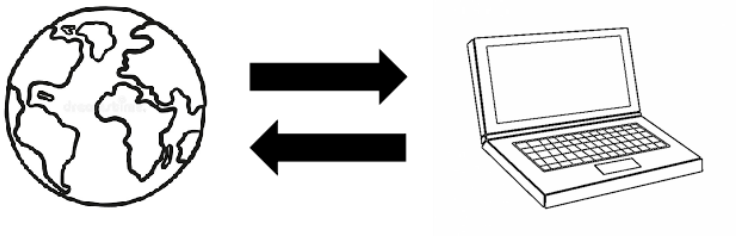
1.3 Projeto de banco de dados geográficos

Aqui é importante diferenciar o uso pessoal do uso institucional de sistemas de informação geográfica. Em uso pessoal e para casos simples, é comum que não seja necessário o uso de banco de dados geográficos em determinados SIGs. Isso acontece, por exemplo, quando se está trabalhando com poucos dados e eles estão normalmente em alguns poucos arquivos *shapefile*. Entretanto, em ambientes corporativos e institucionais, é possível que o volume de dados ultrapasse vários gigabytes de memória, excedendo em muito o limite de um arquivo *shapefile*. Além disso, esses dados podem ser acessados por vários usuários e para diferentes usos.

Nestes casos, a necessidade de projetar um banco de dados geográfico se torna fundamental.

A atividade de projetar um banco de dados envolve a necessidade de representar o "mundo real" através de estruturas computacionais, Figura 11. Esse mapeamento é necessário tanto para dados convencionais quanto para dados geográficos.

Figura 11. Do mundo real para um sistema computacional



Fonte: AUTOR (2021)

Um modelo de (banco de) dados pode ser definido como uma descrição dos tipos de informações que são armazenadas em SGBD (HEUSER, 2009), ou seja, o modelo de dados é uma descrição formal da estrutura de um banco de dados, às vezes chamado de metadados, esquema ou definição de dados.

A criação de modelo de dados para grandes sistemas pode ser uma tarefa complexa e que demanda profissionais especializados. Porém, aqui vamos usar um exemplo mais simples, com o qual vocês podem estar familiarizados. Por exemplo, considere as informações que um sistema acadêmico deve manter como: discentes, docentes, disciplinas, turmas, conteúdos, notas, aprovações entre outras.

Saiba mais!

Como representar todos esses conceitos através de estruturas computacionais?

Ao invés de já pensar em um sistema acadêmico, pode-se começar por algo mais simples, como o controle de notas de um dado professor em uma dada turma. Alguém familiarizado com um pacote de escritório, poderia facilmente criar uma planilha para controlar as notas dessa turma e identificar a aprovação dos alunos, como na Figura 12.

Figura 12: Fragmento de uma planilha simples de controle de notas

	A	B	C	D	E	F	G	H
1	Matricula	Nome	Nota 1	Nota 2	Media	Reposição	Media Final	Aprovado
2	2018111	João Marcos Oliveira	5	4	4,5	5	5	Não
3	2018643	Maria Antônia Silva	8	7	7,5	-	7,5	Sim
4								

Fonte: AUTOR (2021)

Ao fazer isso, pode-se dizer que essa pessoa implicitamente criou um "modelo de dados". Ela definiu quais são os dados relevantes para o objetivo específico, nesse caso apenas identificar os alunos aprovados.

Agora, imagine um centro de ensino com objetivos e necessidades bem maiores:

- Gerar histórico do discente, com as aprovações, reprovações, média em cada disciplina.
- Encontrar facilmente informações de contato de cada discente e docente.
- Ter acesso às mudanças em matriz curricular de um dado curso
- Gerar relatórios por turma e docente.

É fácil prever que uma única tabela de uma planilha não seria suficiente. Por exemplo, para ter um controle dos contatos dos alunos poderia ser usado uma segunda tabela como na Figura 13. Evitando a redundância (repetição) dos dados, e facilitando a sua modificação. Caso o discente mude o seu telefone, por exemplo, a atualização iria ocorrer em um único lugar apenas.

Figura 13: Fragmento de duas planilhas simples para controle de notas e de contato.

	A	B	C	D	E
1	Matricula	Nome	Email	Endereço	Telefone
2	2018111	João Marcos Oliveira	joa@gmail.com	Rua Planejada , 123	99999877799
3	2018643	Maria Antônia Silva	mariaa@gmail.com	Rua da Esperança, 345	98886789964
4					

	A	B	C	D	E	F	G	H
1	Matricula	Nome	Nota 1	Nota 2	Media	Reposição	Media Final	Aprovado
2	2018111	João Marcos Oliveira	5	4	4,5	5	5	Não
3	2018643	Maria Antônia Silva	8	7	7,5	-	7,5	Sim
4								

Fonte: AUTOR (2021)

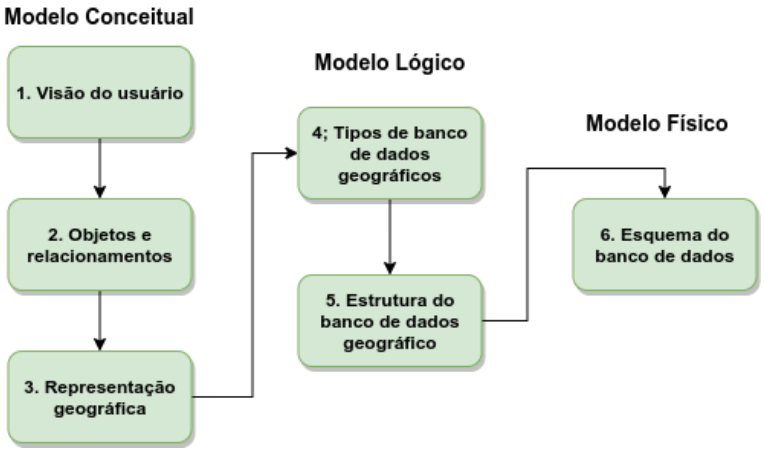
De modo prático, pode-se dizer que a modelagem inclui quais tabelas, suas relações e os tipos de dados de cada coluna. No exemplo da Figura 13, para o nome foi usado texto, para matrícula um número inteiro e para notas números “reais”, ou seja, números com casas decimais.

Um dado domínio ou problema pode ser modelado de diversas formas, sendo algumas melhores que outras para determinados casos, o que torna essa uma das tarefas mais complexas para os profissionais da área de computação. Porém, para esse fascículo serão feitas algumas simplificações, considerando o uso em geoprocessamento. Para esse uso, um usuário geralmente precisa responder algumas perguntas como:

- Qual representação geográfica? É um geo-objeto ou geo-campo?
- Quais são os atributos que representam esses dados?
- Qual representação computacional (tipos de dados) de cada atributo?
- Quantas “classes” de objetos representará? Existe alguma relação ou restrição entre elas?

O projeto do banco de dados geográfico tem muitas similaridades com qualquer outro, envolvendo a criação de modelos conceituais, lógicos e físicos (LONGLEY et. al 2015). A Figura 14 apresenta esses três níveis, como 6 passos distintos.

Figura 14: Níveis e etapas em um projeto de banco de dados.



Fonte: Adaptado de (LONGLEY et. al 2015)

Observe que na Figura 14 não foi destacado o nível do "mundo real", que é geralmente representado nos livros textos de banco de dados. No caso de geoprocessamento, esse nível consistiria dos fenômenos geográficos reais como rios, cidades e montanhas. Longley et. al (2015) definem os fenômenos geográficos como aqueles que possuem o tempo e espaço bem definidos. Longley et. al (2015, pag. 81) exemplificam essa ligação entre atributos, espaço e tempo com a seguinte afirmação: "*A temperatura ao meio-dia local na altitude 34 graus e 35 minutos norte, longitude, 120 graus 0 minutos oeste, era 19 graus Celsius*".

Sem querer tentar ser filosófico — mas talvez o mundo real seja intangível, ou seja, impossível de ser representado computacionalmente, em muitos casos — os fenômenos são convenções e abstrações que criamos no domínio de discurso, como o Everest e o Brasil. Então, pode-se considerar a atividade do projeto de banco de dados a partir do nível conceitual. A seguir serão destacadas cada um destes níveis.

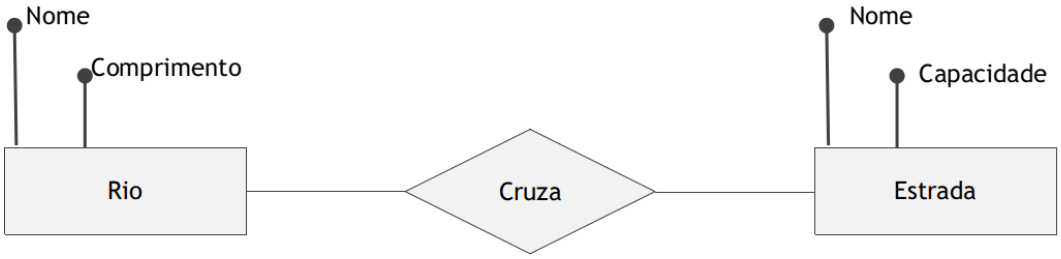
1.3.1 Modelo conceitual

Longley et. al (2015) argumentam que a visão do usuário é o primeiro passo no nível conceitual. Nele os dados serão identificados e organizados, de modo a

entender o domínio ou problema de estudo através de entrevistas e leituras de documentos e relatórios.

O segundo passo é a definição de objetos e seus relacionamentos. De modo similar a outros domínios, nessa etapa poderão ser usados diagramas mais simples, que ainda sejam independentes de um tipo de banco de dados. Por exemplo, a Figura 15 apresenta um diagrama de entidade e relacionamento (DER) simples:

Figura 15: Exemplo simples de um diagrama de entidade e relacionamento.



Esses primeiros diagramas ajudam a entender o domínio, porém na última etapa desse nível é importante que seja selecionada a representação geográfica. A literatura considera duas visões conceituais para representação de dados geográficos:

- **Objetos discretos** (geo-objetos): representam o mundo como entidades com limites bem definidos sobre um espaço vazio (LONGLEY et. al 2015).
- **Campos contínuos** (geo-campos): representam o mundo real como um número finito de variáveis, cada qual definida em uma posição do espaço (LONGLEY et. al 2015)

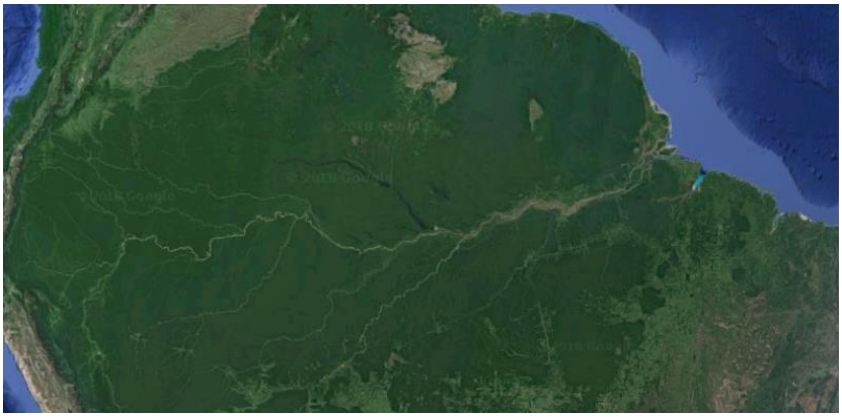
Um exemplo comum de objetos discretos são os mapas geopolíticos, onde os limites são definidos e aceitos por acordos (inter)nacionais, como na Figura 16.

Figura 16. Exemplo de um mapa geo-político.

Fonte: Google Maps, 2021.

Um exemplo comum de campos contínuos são as imagens de satélite, dado que os sensores embutidos nesses equipamentos leem um valor de reflectância para cada posição imageada, como na Figura 17.

Figura 17: Exemplo de uma imagem de satélite.



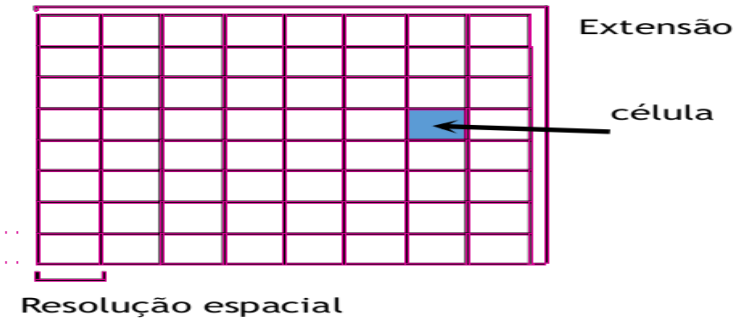
Fonte: Google Earth,2021.

1.3.2 Modelo lógico

Longley et. al (2015) destacam que o nível lógico envolve ajustar o modelo conceitual com suas representações para estruturas de dados, como matrizes e vetores. Em geral, os geo-campos são implementados através de estruturas matriciais, conhecidas também como *rasters*. Essas estruturas dividem o mundo em

matrizes de células e especificam atributos para elas (LONGLEY et. al 2015). A Figura 18, destaca algumas características específicas nessa representação, como a extensão e a resolução espacial.

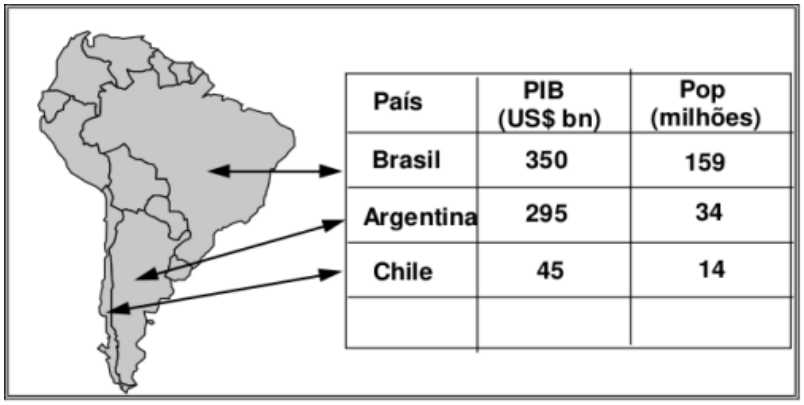
Figura 18. Representação matricial.



Fonte: AUTOR (2021)

Os geo-objetos são geralmente representados através de estruturas vetoriais. Essas estruturas representam o mundo como coleções de feições geométricas (ponto, linha e área) com um ou mais atributos associados. Nessa representação, a componente espacial utiliza primitivas geométricas que são associadas a dados convencionais, como textos e números. A Figura 19 destaca um mapa geopolítico da América do Sul, com informações como o produto interno bruto e a população de cada país.

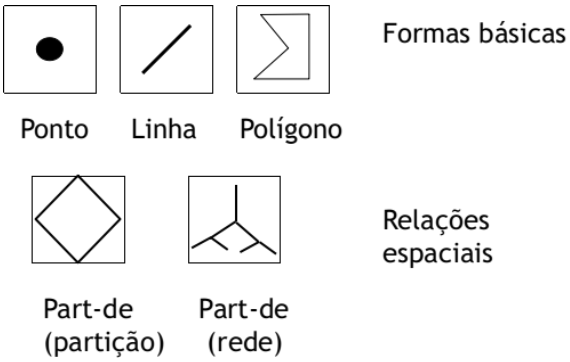
Figura 19. Representação vetorial.



Fonte: (CÂMARA et. al. 2001)

Com a definição dessas estruturas, pode-se incluí-las nos diagramas de relacionamento ou de classes, através de pictogramas (SHEKKAR e CHAWLA, 2003). A Figura 20 destaca alguns desses pictogramas.

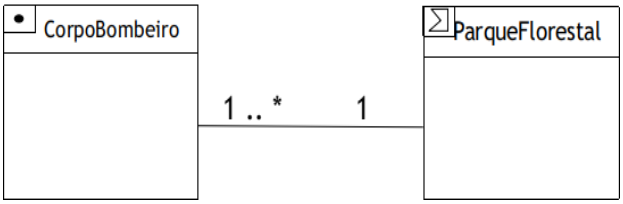
Figura 20. Pictogramas para as formas e relações espaciais.



Fonte: (SHEKKAR e CHAWLA , 2003)

Esses pictogramas poderiam ser usados em diagramas de classes como na Figura 21, onde cada corpo de bombeiro é modelado como um ponto e está localizado em um parque florestal modelado por um polígono.

Figura 21. Exemplo simplificado de um diagrama de classe usando pictogramas.

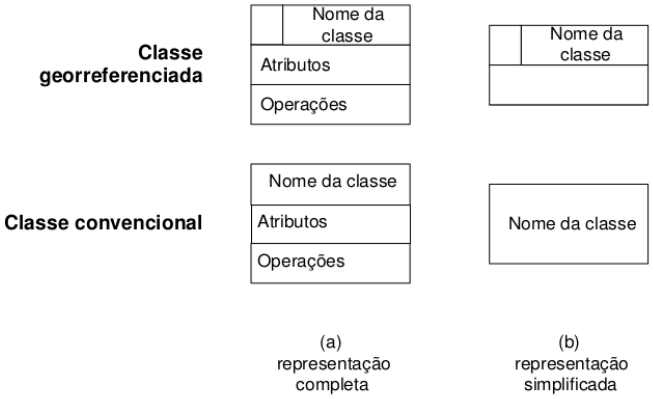


Fonte: (AUTOR, 2021)

O uso de pictogramas ajuda a tornar explícito o formato de representação e da relação espacial. Porém, além dos pictogramas, Borges et al. (2001) propuseram um modelo mais completo denominado de OMT-G. Esse modelo é uma extensão do OMT (Object Modeling Technique – Técnica de Modelagem de Objetos) proposto por Rumbaugh et al. (1991). Nele, podemos modelar dados espaciais e não espaciais, através de classes convencionais e georreferenciadas. No caso das classes

georreferenciadas a um espaço para a inclusão de um pictograma, como pode ser observado na Figura 22.

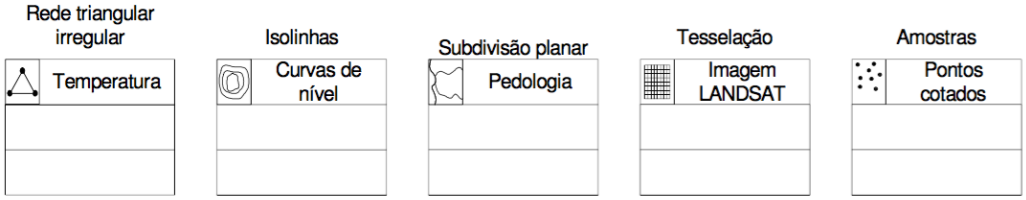
Figura 22. Representação de classes convencionais e georreferenciadas na OMT-G.



Fonte: (BORGES et al. 2001)

As classes georreferenciadas são especializadas em classes do tipo geo-campo e geo-objeto, seguindo o conceito descrito na seção 2.1. O modelo OMT-G define cinco classes descendentes de geo-campo: isolinhas, subdivisão planar, tesselação, amostragem e malha triangular, Figura 23.

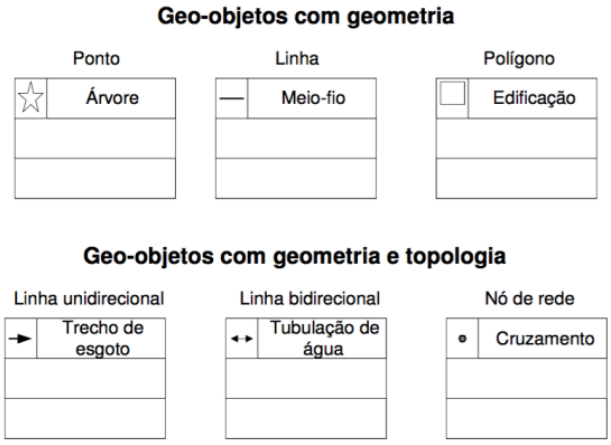
Figura 23. Geo-campos.



Fonte: (BORGES et al. 2001)

Duas classes descendentes de geo-objeto: geo-objeto com geometria e geo-objeto com geometria e topologia, Figura 24.

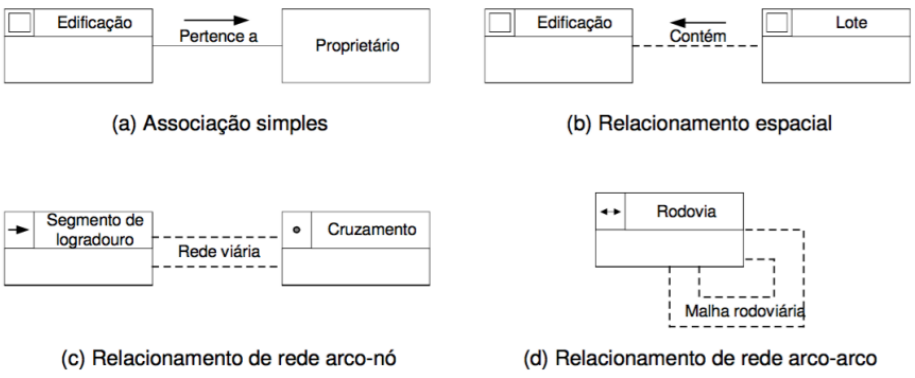
Figura 24. Geo-objetos.



Fonte: (BORGES et al. 2001)

Além das classes, o modelo OMT-G descreve três categorias de relacionamentos, entre elas: associações simples, relacionamentos topológicos em rede e relacionamentos espaciais, Figura 25.

Figura 25.Relacionamentos.



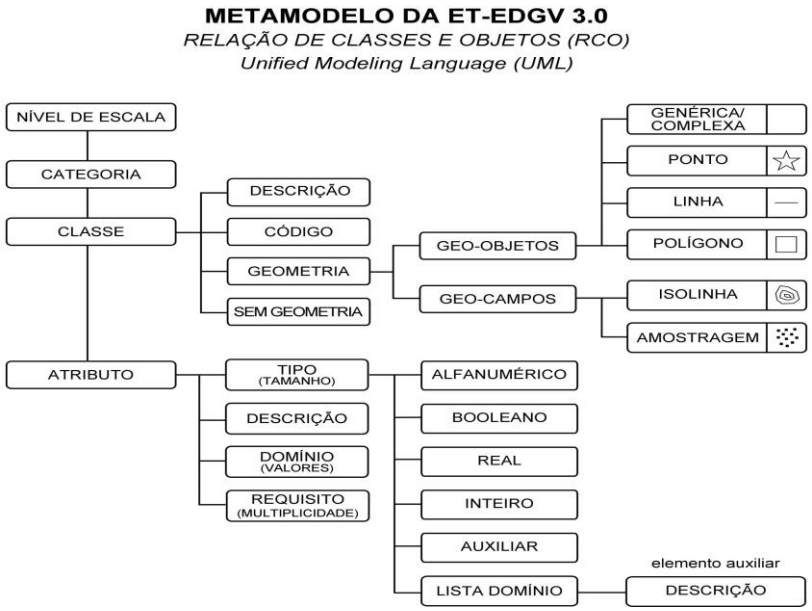
Fonte: (BORGES et al. 2001)

Saiba mais!

Esse fascículo está mais prático e simplificado. Porém, conhecer o modelo OMT-G é importante para quem quer trabalhar com geoprocessamento no Brasil. Para saber mais sobre o OMT-G, acesse http://www.dpi.inpe.br/livros/bdados/cap3.pdf

Como comentado anteriormente, a modelagem é uma atividade complexa que exige muita experiência e conhecimento da área de domínio. Além disso, um mesmo problema pode ser modelado de diversas maneiras. Essas diferenças podem inviabilizar o compartilhamento de dados entre os produtores e usuários de dados e informações cartográficas. Então, a Comissão Nacional de Cartografia (CONCAR) homologou um conjunto de Especificações Técnicas para Estruturação de Dados Geoespaciais Vetoriais (ET-EDGV) para ser um modelo conceitual e semântico do mapeamento de referência brasileiro e padronizar estruturas de dados (DSG,2017). A atual versão da ET-EDGV (3.0) tem enfoque na definição da estrutura de dados vetoriais e teve como base o modelo OMT-G.

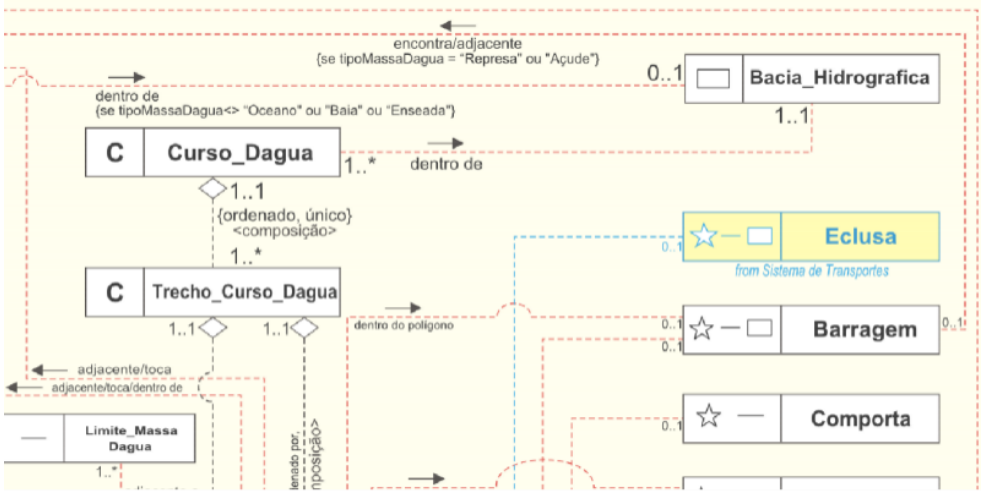
Figura 26.Metamodelo da ET-EDGV 3.0.



Fonte: (DSG,2017)

Dentro dessa especificação é possível encontrar modelos de dados para diferentes categorias, como: Hidrografia, Energia e Comunicações, Relevo, Vegetação, Sistemas de Transportes, Administração Pública entre outros. Cada uma dessas categorias possuem diversas classes e relações que seguem as definições da OMT-G, como apresentada na Figura 27.

Figura 27. Fragmento do diagrama conceitual da categoria Hidrologia.



Fonte: (DSG,2017)

1.3.3 Modelo físico

A Diretoria de Serviço Geográfico (DSG) disponibiliza um plugin para o QGis (DsgTools) que auxilia na criação de banco de dados de acordo com as ET-EDGV. Esse plugin fornece já o **modelo físico** do banco de dados. Longley et. al (2015) destacam que o modelo físico é a definição do esquema de banco de dados utilizando a linguagem de definição de dados, que é descrita na Seção 1.4.

Saiba mais!

Pesquise mais e explore essa especificação, dado que em determinadas instituições esse modelo pode ser mais que recomendado. As especificações técnicas podem ser baixadas em:
http://www.geoportal.eb.mil.br/portal/index.php/inde2?id=139

Atenção!

O plugin DsgTools pode ser instalado diretamente pelo QGIS. Mais informações em https://github.com/dsgoficial/DsgTools

1.4 SQL: Criando, armazenando e recuperando dados

Na Seção 1.1 foi destacado que os sistemas gerenciadores de banco de dados atuais utilizam arquitetura cliente/servidor, onde eles são servidores especializados no armazenamento e na recuperação de dados. Eles fornecem esses serviços para diversos outros programas clientes rodando em diferentes sistemas operacionais e escritos em distintas linguagens de programação. Para permitir essa comunicação entre o cliente e servidor, surgiu a SQL (Structured Query Language – Linguagem de Consulta Estruturada) na década de 70 proposta por um pesquisador da IBM e adotada pela maioria dos SGBD-R e SGBD-OR. Antes de se tornar padrão, ela foi originalmente denominada de SEQUEL (Structured English QUEry Language) e era a interface para um sistema de banco de dados relacional experimental, chamado SYSTEM R (ELMASRI e RAMEZ, 2005). Desde então, a linguagem tem sofrido sucessivas atualizações, como apresentado na Tabela 1:

Tabela 1. Evolução da linguagem 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
2003	**SQL:2003**	Pequenas atualizações ao SQL:1999 e inclusão do SQL/XML
2006	**SQL:2006**	Avanços na integração entre XML e SQL, como a inclusão de código XQuery em uma consulta SQL

Além das versões destacadas na Tabela 1, mais recentemente foram lançadas as versões SQL:2008, SQL:2011, SQL:2016 e SQL:2019.

A linguagem SQL pode ser entendida como formada por 3 sub-linguagens:

- **SQL DDL** é a linguagem de definição de dados, fornecendo comandos para definir e modificar esquemas de tabelas, remover tabelas, criar índices e definir restrições de integridade. Os comandos mais comuns são: create table, alter table e drop table.
- **SQL DML** é a linguagem de manipulação de dados, fornecendo comandos para consultar, inserir, modificar e remover dados. Os comandos mais comuns são: select, insert, update e delete.
- **SQL DCL** é a linguagem de controle de acesso, fornecendo comandos para permitir e definir quais dados um determinado usuário tem acesso, e qual o tipo de acesso. Os comandos mais comuns são: grant e revoke.

A linguagem SQL é geralmente transparente aos usuários de sistemas de informação geográfica, porém conhecer a sintaxe de alguns comandos básicos permite a esses usuários um maior controle sobre a base de dados. Além disso, será possível extrair algumas informações de modo muito mais rápido e sem a necessidade de usar um SIG. Veremos a seguir alguns comandos básicos para criar tabelas, inserir e consultar dados dessas.

Saiba mais!

Esse fascículo foca nos comandos mais básicos de definição e manipulação de dados. Não será tratado o controle de acesso que é uma tarefa normalmente executada por administradores de banco de dados. Para saber mais sobre controle de acesso, pesquise na Web e acesse o seguinte tutorial: https://docs.aws.amazon.com/pt_br/AmazonRDS/latest/UserGuide/Appendix.PostgreSQL.CommonDBATasks.html

Atenção!

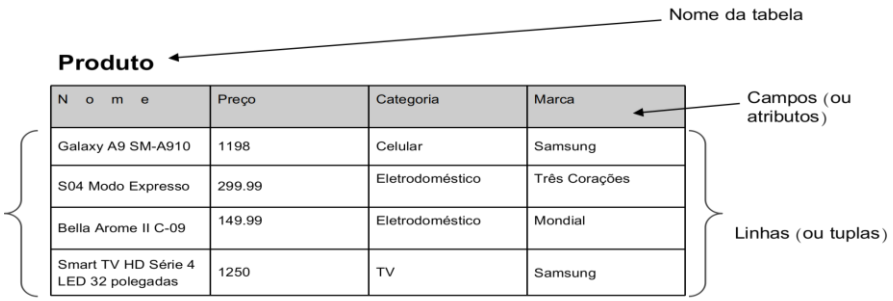
Lembrem-se que esse fascículo visa dar uma visão básica sobre banco de dados para usuários de sistema de informação geográfica. Para saber mais sobre a linguagem SQL, os mais curiosos podem acessar um excelente tutorial em: https://www.w3schools.com/sql/

1.4.1 Criando tabelas de dados

Ao usar um SIG, o comando de criação de tabelas é normalmente realizado de modo implícito quando usamos algumas interfaces de importação de dados. Porém, no Capítulo 4 veremos essa operação para destacar melhor os tipos de dados providos pelo PostGIS.

Antes de apresentar esse comando é importante destacar os seguintes conceitos ilustrados na Figura 28.

Figura 28: Conceitos principais associados a uma tabela



N o m e	Preço	Categoria	Marca
Galaxy A9 SM-A910	1198	Celular	Samsung
S04 Modo Expresso	299.99	Eletrodoméstico	Três Corações
Bella Arome II C-09	149.99	Eletrodoméstico	Mondial
Smart TV HD Série 4 LED 32 polegadas	1250	TV	Samsung

Uma tupla ou linha é usualmente referida como registro, enquanto a tabela é uma coleção de registros. Todo campo tem um domínio, ou seja, ele precisa ter um tipo de dado definido. Os tipos de dados convencionais são:

- Caracteres: CHAR(20), VARCHAR(50)
- Números: INT, BIGINT, SMALLINT, FLOAT
- Dinheiro: MONEY,
- Data: DATETIME.

Então, para criar uma tabela é necessário definir o seu nome e campos. A sintaxe básica do comando create table é:

```
CREATE TABLE < nome tabela > (
  < nome coluna1 > < tipo coluna1 > ,
  < nome coluna2 > < tipo coluna2 > ,
  ...
);
```

Por exemplo, a tabela da Figura 28 poderia ser criada com o comando a seguir:

```
CREATE TABLE Produto
(
  Nome varchar(255),
  Preco float,
  Categoria varchar(100),
  Marca varchar(100)
);
```

Saiba mais!

É possível experimentar esse comando em um banco de dados criado através do PGAdmin instalado na Seção 1.2, ou *online* através do seguinte endereço: https://www.w3schools.com/sql/trysql.asp?filename=trysql_create_table

O comando create table apenas prepara o banco de dados para receber os dados, que serão inseridos através do comando insert into como será apresentado na próxima seção.

1.4.2 Armazenando dados

O armazenamento ou a inserção das tuplas é realizada através do comando insert into, onde a sintaxe básica é apresentada a seguir:

```
INSERT INTO table_name (coluna1, coluna2, coluna3, ...)
VALUES (valor1, valor2, valor3, ...);
```

Quando estão sendo adicionados valores para todos os campos, não é necessário especificar os nomes das colunas. Por exemplo, uma dada tupla da Figura 28 poderia ser adicionada através do seguinte comando:

```
INSERT INTO Produto
VALUES ('Galaxy A9 SM-A910',1198,'Celular', 'Samsung'        );
```

Depois das inserções, na próxima seção é apresentado o comando mais utilizado, que permite a recuperação ou consulta de dados.

1.4.3 Recuperando dados

A recuperação ou consulta de dados é realizada pelo comando select, onde a sua forma mais básica é:

```
SELECT      <lista de campos>
FROM    <lista de tabelas>
WHERE       <condição>.
```

Onde:

- *<lista atributos>* é uma lista de nomes de atributo cujos valores devem ser recuperados pela consulta. Quando deseja recuperar todos os atributos basta usar o símbolo *.
- *<lista tabelas>* é uma lista dos nomes das tabelas exigidos para processar a consulta.
- *<condição>* é uma expressão condicional (booleana ou lógico) que identifica as tuplas a serem recuperadas pela consulta. É uma expressão que pode ser avaliada como verdadeira ou falsa.

Na condição será comum o uso de operadores relacionais como:

- =, compara se dois valores são iguais
- <>, compara se dois valores são diferentes
- >, compara se, dados dois valores, o primeiro é maior que o segundo. Ainda pode-se usar o >= que considera como verdadeiro quando é maior ou igual.
- <, compara se, dados dois valores, o primeiro é menor que o segundo. Ainda pode-se usar o <= que considera como verdadeiro quando é maior ou igual.

Além dos operadores relacionais é comum o uso de operadores lógicos que permitem agrupar os resultados de outras operações.

- AND, dados dois valores ele retorna verdadeiro apenas quando ambos são verdadeiros.
- OR, dados dois valores ele retorna falso apenas quando ambos são falsos.
- NOT, dado um valor verdadeiro ele retorna falso e vice-versa.

O resultado do comando SQL pode ser entendido como uma nova tabela temporária, que só existe em memória. Por exemplo, ainda usando o exemplo da Figura 28, é possível recuperar todos as tuplas onde a categoria seja eletrodoméstico:

```
SELECT *
FROM produto
WHERE categoria = 'Eletrodoméstico'
```

O resultado dessa consulta seria uma tabela como a apresentada na Figura 3.2:

Figura 29: Resultado do comando select.

Nome	Preço	Categoria	Marca
S04 Modo Expresso	299.99	Eletrodoméstico	Três Corações
Bella Arome II C-09	149.99	Eletrodoméstico	Mondial

Nesse exemplo foi recuperado todos os atributos, mas seria possível recuperar apenas alguns como a seguir:

```
SELECT Nome, Marca
FROM produto
WHERE preco > 1000
```

Nesse caso, o retorno incluiria apenas o nome e a marca como na Figura 30.

Figura 30: Resultado do comando select onde o preço é maior que 1000.

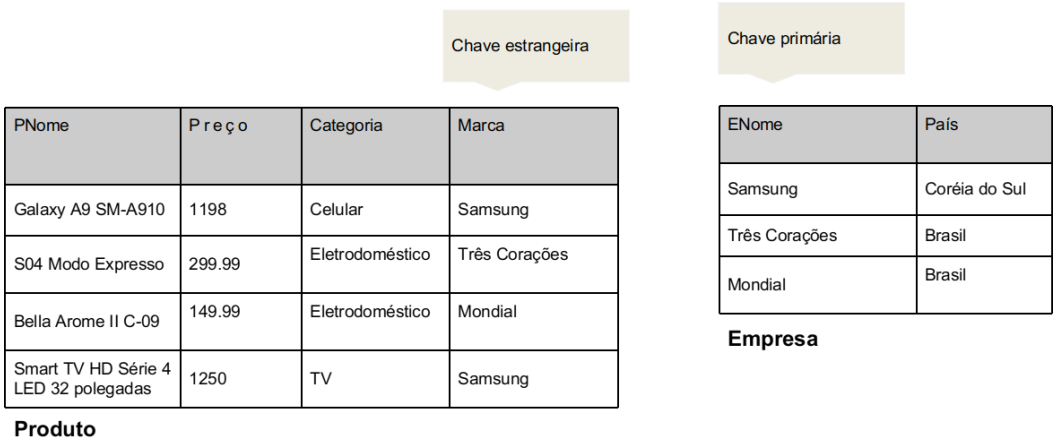
Nome	Marca
Galaxy A9 SM-A910	Samsung
Smart TV HD Série 4 LED 32 polegadas	Samsung

O comando select pode se tornar bem complexo, porém esse fascículo tem como foco usuários de sistema de informação geográfica. Então, está se buscando destacar apenas exemplos mais simples que podem ainda ser úteis a esses usuários. Mesmo assim, é necessário ainda apresentar exemplos de consultas que relacionam duas ou mais tabelas.

Uma das vantagens dos bancos de dados relacionais é a forma que eles permitem relacionar duas ou mais tabelas. Para isso, usam-se os conceitos de **chave primária** e **chave estrangeira**. A primeira é utilizada como identificador único em uma tabela, como CPF de usuário ou o código de um produto. Já a chave estrangeira é uma referência em uma tabela a uma chave primária de outra tabela. Isso ocorre porque nos bancos de dados relacionais é comum criarmos várias tabelas para evitar a redundância. Por exemplo, uma tabela Produto poderia incluir um campo com o nome do país sede de uma dada marca. No entanto, ao fazer isso, haveria uma maior redundância de dados, pois seria necessário entrar com essa informação para cada novo produto cadastrado. Além disso, existe o risco de erros de digitação como em uma dada entrada. Por exemplo, em algumas entradas digitar Brasil e em outras digitar Brazil com **z**. Isso resultaria em consultas imprecisas no futuro.

Ao invés disso, é comum separar as informações da empresa em outra tabela. Desse modo poderia ser incluído até outras informações, como ano de fundação, CNPJ entre outras. Na Figura 31 são mostradas duas tabelas, a de produto similar a que foi apresentada anteriormente e a de empresa que agora possui o nome do país sede. Na tabela Empresa o nome é a chave primária que irá se relacionar com o atributo Marca na tabela Produto.

Figura 31: Exemplo de uso de chave estrangeira e chave primária.



Com essas duas tabelas é possível realizar consultas usando informações de ambas, por exemplo: **Quais os produtos de empresas com sede no Brasil?**

Uma forma simples de responder a essa pergunta é incluir a restrição de que a chave primária da tabela Empresa precisa coincidir com a chave estrangeira na tabela Produto, como no código abaixo:

```
SELECT          PNome, Preço
FROM            Produto, Empresa
WHERE           Marca=ENome
        AND Pais='Brasil'
```

Outra forma, seria através da cláusula join como a seguir:

```
SELECT          PNome, Preço
FROM            Produto,
WHERE       Pais='Brasil'
JOIN  Empresa ON Marca=ENome ;
```

O SQL suporta várias operações de agregação para os dados básicos como o sum, count, min, max e avg. Por exemplo, consultar o preço médio dos produtos:

```
SELECT      avg(Preco)
```

```
FROM                    Produto
```

Ou contar a quantidade de produtos de uma dada marca:

```
SELECT          count(*)
FROM                    Produto
where marca = 'samsung
```

Atenção!

Na Seção 2.1, serão apresentadas outras operações de agregação específicas para dados espaciais.

Saiba mais!

Pesquisem sobre as cláusulas group-by, order e having. Uma fonte interessante é o tutorial da w3schools: https://www.w3schools.com/sql/sql_groupby.asp

ATIVIDADE DE APRENDIZAGEM DA UNIDADE 1

1. Conceitue e exemplifique dados, banco de dados e sistema gerenciadores de banco de dados.

2. Quais são os conceitos utilizados pelos principais modelos de banco de dados utilizados atualmente?

3. Foram criadas duas tabelas em um Banco de Dados utilizando os seguintes comandos:

```
CREATE TABLE aluno ( matricula int , nmaluno  varchar(80), cdcurso int)
CREATE TABLE curso (cdcurso int , nmcurso  varchar(80))
```

Qual a consulta SQL que retorna o nome do aluno e do curso?

4. Observe o banco de dados composto pelas relações a seguir, em que atributos sublinhados indicam a chave primária, e atributos em itálico apontam as chaves estrangeiras.

Usuario (CdUser, Nome, Sexo, *CdGrp*)
Post (CodPost, Mensagem, Data, *CdUser*)
Grupo (CdGrp, Nome)

As quatro afirmações abaixo com relação ao banco de dados acima podem ser verdadeiras (V) ou falsas (F). Marque a resposta correta:

I – um usuário pode estar alocado em mais de um grupo;
II – um usuário pode estar associado a mais de um Post;
III – um grupo pode conter mais de um usuário;
IV – o nome de um grupo é único;

a) V, V, V, V

b) V, F, V, V
c) F, V, V, F
d) F, F, V, V

5. A Fig 1 representa uma tabela de nome FUNC. A Fig 2 mostra o resultado de uma consulta executada pelo SGDB PostgreSQL.

Fig 1. Tabela base

Mat	Nome	Cargo	Sal
20161	Maria	Professor	1000
20164	Carlos	Medico	3000
20171	Marcos	Engenheiro	2200
20174	José	Enfermeiro	2000
20153	Pedro	Programador	1000
20181	Lucas	Topografo	1500

Fig 2. Tabela resultante da consulta

Nome	Cargo	Sal
Carlos	Medico	3000
Marcos	Engenheiro	2200
José	Enfermeiro	2000

Qual é a consulta em SQL que foi executada sobre a tabela mostrada na Fig 1 para obter o resultado mostrado na Fig 2.

6. Foi criado uma tabela em um Banco de Dados utilizando o seguinte comando:

```
CREATE TABLE met (
      cidade          varchar(80),
      temp_baixa        int,
      temp_alta         int,
      precip          real)
```

Escreva um comando SQL que insere uma nova linha na tabela met.

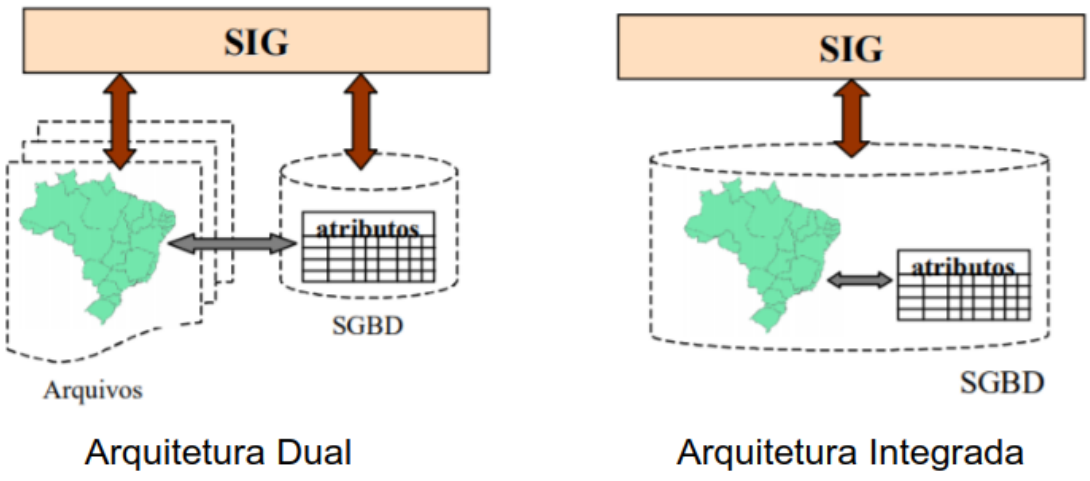
UNIDADE 2

Banco de Dados Geográficos e Sistemas de Informação Geográfica

2.1 Arquitetura de sistemas de banco de dados geográficos

Gerenciadores de banco de dados geográficos permitem o acesso aos dados através de uma linguagem de consulta e manipulação como vimos na Seção 1.4. Entretanto, usuários finais irão acessar os dados geográficos normalmente através de aplicações *desktops* ou *web services*, usando assim de forma transparente o banco de dados. Neste capítulo, iremos focar nos sistemas de informação geográfica para *desktop*, como os atuais QGIS, TerraView Spring, ArcGIS e gVSIG. Nesses sistemas, existem basicamente duas formas de integrar os Sistemas de Informações Geográficas (SIGs) com SGBDs (FERREIRA et. al, 2005).

Figura 32: Arquitetura de sistemas de informação geográfica



Fonte: (FERREIRA et. al, 2005)

A arquitetura dual, mostrada na Figura 32, surgiu porque os primeiros gerenciadores de bancos de dados tinham suporte apenas a tipos de dados básicos

como números, textos e datas. Então, os SIGs precisavam armazenar as componentes espaciais dos objetos separadamente em arquivos próprios. Ferreira et. al (2005) apresentam algumas dificuldades relacionadas a arquitetura dual:

- No controle e manipulação das componentes espaciais;
- Em manter a integridade entre a componente espacial e a componente alfanumérica;
- No processamento entre a parte convencional e a parte espacial;
- Na interoperabilidade, dado que cada sistema tem seu próprio formato.

Como solução a algumas destas dificuldades, esses sistemas de desktop passaram a utilizar uma arquitetura integrada através de extensões espaciais. Essa arquitetura consiste em utilizar extensões espaciais desenvolvidas sobre um banco de dados objeto relacional. Essas extensões podem incluir tipos de dados, métodos de acesso e operadores específicos a uma dada necessidade. Exemplos de extensões, podemos citar o Oracle Spatial, o PostGIS do PostgreSQL e o Spatial Lite e Geopackage baseadas no SQLite. Dentre as características que tornam essas extensões interessantes, podemos citar (FERREIRA et. al, 2005):

- Fornecem tipos de dados espaciais;
- Estendem o SQL para incluir operações sobre esses dados espaciais; e,
- Possuem métodos de acesso e armazenamento para otimizar esse tipo de consulta.

Para garantir uma maior interoperabilidade entre os tipos e operações suportadas por diferentes extensões, o Open Geoscience Consortium (OGC) propôs a Simple Features for SQL (SFSQL) que especifica geometrias vetoriais, operações topológicas e métricas. Ela ainda define um esquema de tabelas para metadados de informações espaciais, por exemplo, a que armazena os dados sobre os sistemas de referência espacial.

A seguir veremos os tipos de dados e operações espaciais definidas pela SFSQL, bem como os métodos de indexação implementados pelas principais extensões.

2.1.1 Tipos de dados espaciais

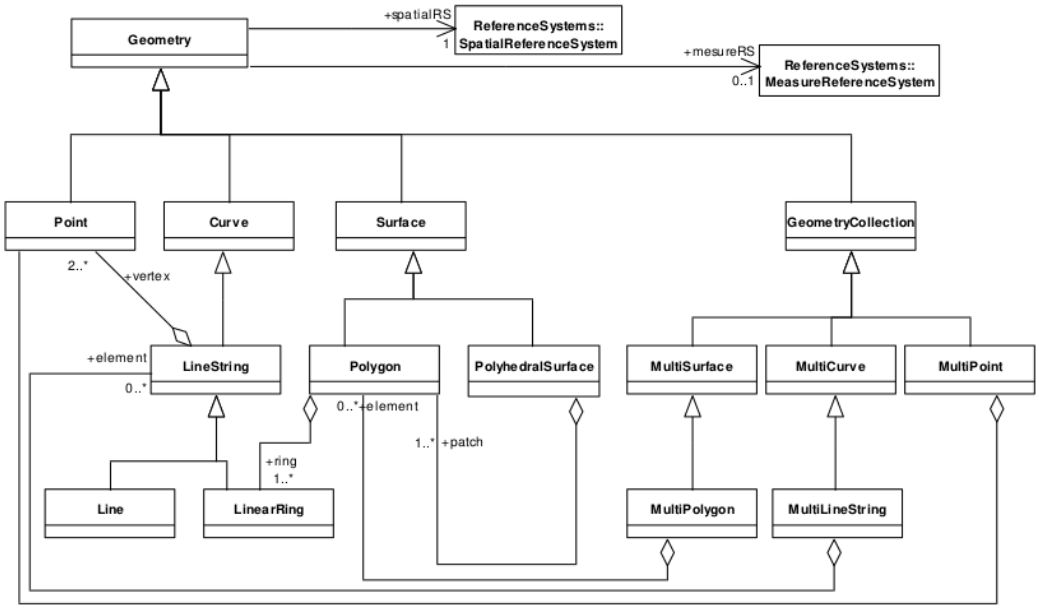
Vimos na Seção 1.3, que as representações de geo-objetos são mapeadas para estruturas de dados vetoriais enquanto as de geo-campos são mapeadas para matrizes. Nesse fascículo, será dado um enfoque maior aos dados vetoriais. O modelo de dados utilizado pelos bancos de dados para representar essa estrutura é um modelo de geometrias simples definido pelo OGC (*Open Geospatial Consortium*) , Figura 4.2.

Observe na Figura 33 que o tipo *geometry* precisa estar sempre associado a um sistema de referência. Para isso é usado uma tabela com todos os códigos EPSG existentes. O código EPSG vem do nome do Grupo Europeu de Pesquisa Petrolífera (European Petroleum Survey Group), que associou códigos numéricos a vários Sistemas de Referência de Coordenadas (SRC). No caso do PostGIS, a criação dessa tabela é uma das ações que ocorrem após a execução do seguinte comando:

```
create extension postgis;
```

Então, após a execução desse comando é criada a tabela spatial_ref_sys que pode ser consultada como na Figura 34.

Figura 33: Hierarquia dos tipos geométricos



Fonte: (HERRING, J. 2010)

Você sabia??

A Figura 33 representa os dados vetoriais de acordo com um modelo orientado a objetos onde a classe Geometry é a raiz na qual são derivadas as demais.

Figura 34: Exemplo de códigos EPSG data tabela spatial_ref_sys

Query Editor | Query History | Scratch Pad

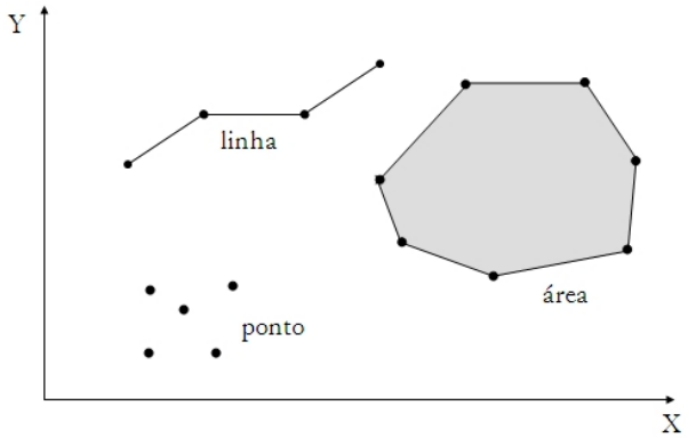
```
1  select * from spatial_ref_sys;
```

Data Output | Explain | Messages | Notifications

	srid [PK] integer	auth_name character varying (256)	auth_srid integer	srtext character varying (2048)	proj4text character varying (2048)
1	3819	EPSG	3819	GEOGCS["HD1909",DATUM["Hu...	+proj=longlat +ellps=bessel +to...
2	3821	EPSG	3821	GEOGCS["TWD67",DATUM["Tai...	+proj=longlat +ellps=aust_SA +...
3	3824	EPSG	3824	GEOGCS["TWD97",DATUM["Tai...	+proj=longlat +ellps=GRS80 +t...
4	3889	EPSG	3889	GEOGCS["IGRS",DATUM["Iraqi_...	+proj=longlat +ellps=GRS80 +t...
5	3906	EPSG	3906	GEOGCS["MGI 1901",DATUM["...	+proj=longlat +ellps=bessel +to...
6	4001	EPSG	4001	GEOGCS["Unknown datum bas...	+proj=longlat +ellps=airy +no_d...
7	4002	EPSG	4002	GEOGCS["Unknown datum bas...	+proj=longlat +ellps=mod_airy ...
8	4003	EPSG	4003	GEOGCS["Unknown datum bas...	+proj=longlat +ellps=aust_SA +...
9	4004	EPSG	4004	GEOGCS["Unknown datum bas...	+proj=longlat +ellps=bessel +n...
10	4005	EPSG	4005	GEOGCS["Unknown datum bas...	+proj=longlat +a=6377492.018 ...
11	4006	EPSG	4006	GEOGCS["Unknown datum bas...	+proj=longlat +ellps=bess_nam...
12	4007	EPSG	4007	GEOGCS["Unknown datum bas...	+proj=longlat +a=6378293.645...

Os atuais formatos espaciais e sistemas de informação geográfica dão suporte, principalmente, aos três tipos primitivos: ponto, área (polígono) e linha. O ponto é o mais básico de todos, de onde os demais são construídos, como apresentado na Figura 35. Inclusive, curvas são aproximadas através da adição de vários vértices a uma linha.

Figura 35: Tipos geométricos primitivos.



Fonte: (CÂMARA, 2005)

Ao usar um banco de dados geográfico, usamos todos os conceitos que vimos relacionados aos dados convencionais, mas com a possibilidade de incluir

esses novos tipos de dados. Por exemplo, podemos criar uma tabela para armazenar os limites da Universidade Federal do Maranhão e seus edifícios como abaixo:

```
CREATE TABLE ufma (
fid INTEGER NOT NULL PRIMARY KEY,
geom     geometry(POLYGON,4326),
nome     CHARACTER VARYING(30)
);
```

Observe que o primeiro atributo é apenas um identificador, a chave primária da tabela. O segundo atributo é do tipo *geometry*, especificamente um *polygon*. O número 4326 é o código EPSG. Ele indica que estamos usando o datum WGS84 e com coordenadas geográficas.

Para a inserção no banco de dados, a OpenGIS especificou dois formatos para expressar objetos espaciais: Well-Known Text (WKT) e o Well-Known Binary (WKB). Esses formatos incluem informações sobre o tipo do objeto e as coordenadas que o formam. Basicamente é o nome da representação seguido de pares de coordenada, por exemplo:

- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT((0 0),(1 2))
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))

Esses formatos suportam apenas geometrias 2D e sem incluir o SRID associado. Então, o PostGIS implementou uma extensão a esses formatos, ou seja, todo WKB / WKT válido é um EWKB / EWKT válido. Basicamente, o PostGIS EWKB / EWKT adiciona coordenadas 3D, 4D e a especificação do SRID. Por exemplo:

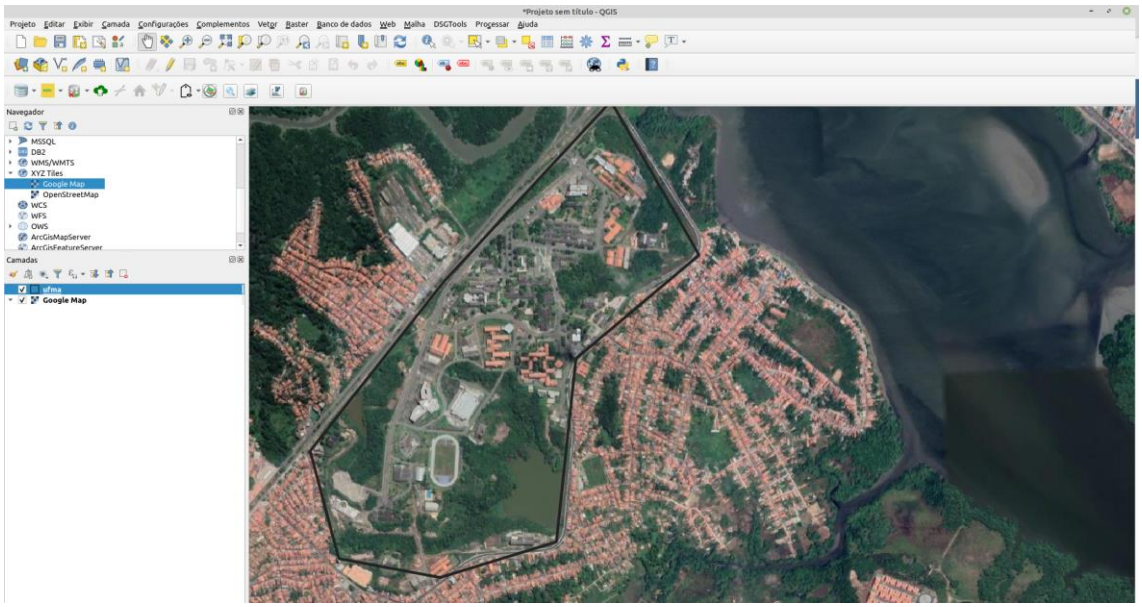
- POINT(0 0 0) -- XYZ
- SRID=32632;POINT(0 0) -- XY com SRID
- POINTM(0 0 0) -- XYM
- POINT(0 0 0 0) -- XYZM
- TRIANGLE ((0 0, 0 9, 9 0, 0 0))
- TIN(((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0)))

Ao usar o PostGIS podemos usar esse formato já que ele permite especificar o SRID diretamente na literal:

```
INSERT INTO ufma  VALUES
            (1,'SRID=4326;
            POLYGON ((-44.3109882 -2.5559354,-44.3151081 -2.5611445,
                                    -44.3142283 -2.5644456,-44.3111169 -2.5650244,
                                    -44.307555 -2.5639312,-44.3069541 -2.5583149,
                                    -44.3031776 -2.5551423,-44.3058813 -2.5504263,
                                    -44.3109882 -2.5559354))',
'Contorno');
```

Após a inserção, podemos visualizar esse dado em um sistema de informação geográfica como o QGIS. A Figura 36 apresenta o polígono sobre uma camada do Google Maps.

Figura 36: Visualizando um dado espacial no QGIS



Fonte: AUTOR (2021)

Atenção!

Observe que esse polígono é apenas uma simplificação, a representação real da UFMA exigiria muito mais vértices.

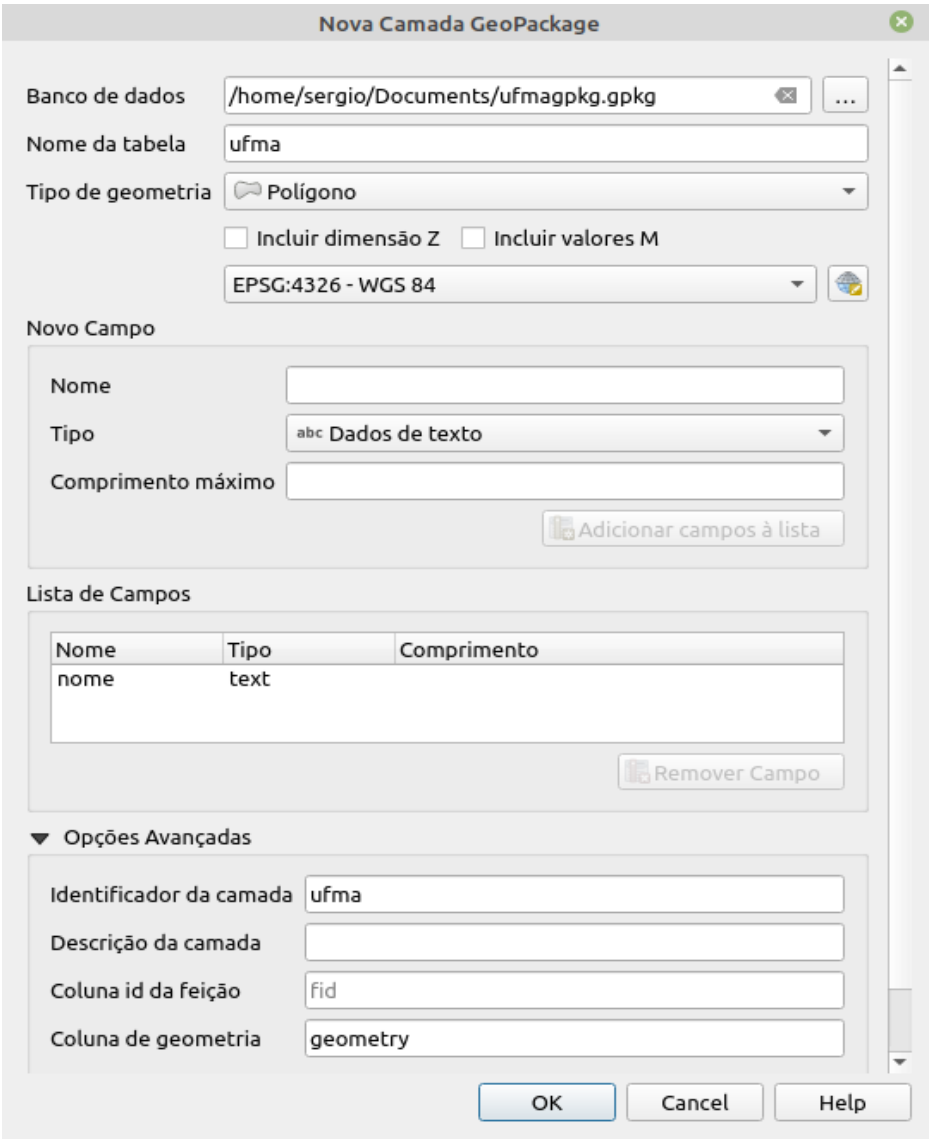
Alternativamente aos sistemas gerenciadores de banco de dados como o PostgreSQL e o Oracle, existem formatos mais leves que não requerem a instalação de um servidor de banco de dados e são baseados no SQLITE[3]. Como exemplos, temos o Spatial Lite[4] e o Geopackage, onde o primeiro é o mais antigo e o segundo foi proposto e é mantido pela OGC. O Geopackage já é suportado pelas versões mais

[3] https://www.sqlite.org/index.html

[4] https://www.gaia-gis.it/fossil/libspatialite/home

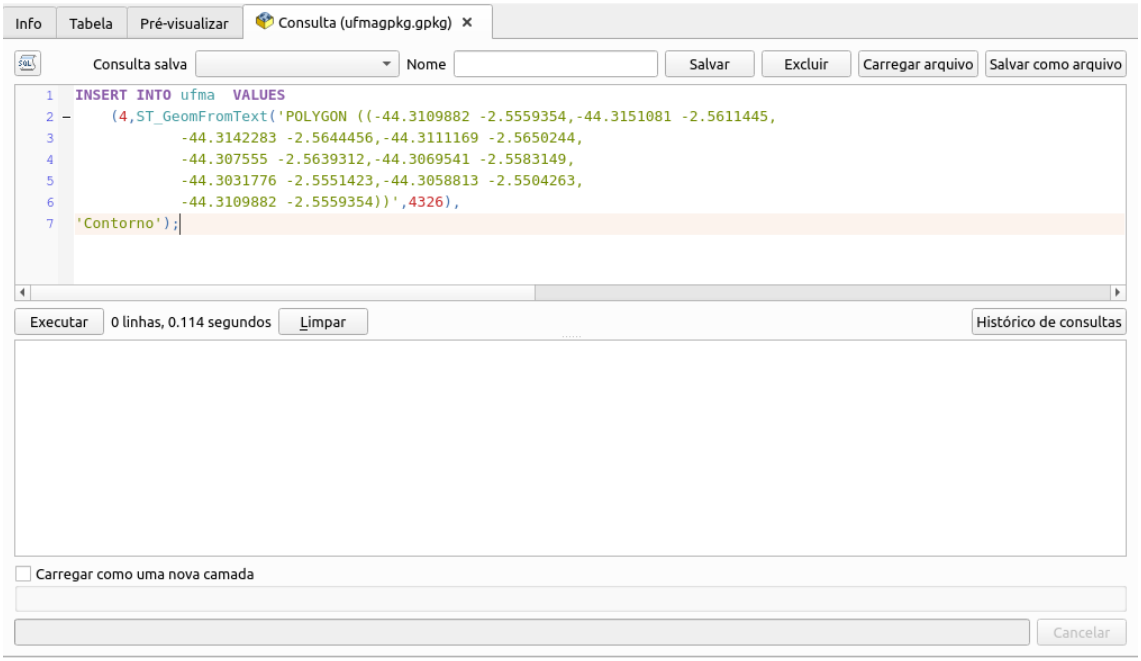
recentes dos sistemas de informação geográfica, como o QGIS. Por exemplo, no QGIS é possível criar uma base de dados Geopackage como na Figura 37.

Figura 37: Criando uma base de dados com o Geopackage



Observe que, nesse caso, não é preciso executar explicitamente o comando SQL *create table*, já que nessa tela da Figura 37 já definimos o nome da tabela. Na lista de campos só foi adicionado o atributo nome, pois já é padrão a criação de uma chave primária e da geometria, de acordo com o tipo selecionado. Então, agora podemos inserir a mesma área usada no exemplo anterior usando o comando insert into, como na Figura 38:

Figura 38: Inserindo um dado em uma base geopackage.



Observem que nesse caso foi necessário usar o formato WKT com a função ST_GeomFromText para poder informar o código EPSG. Dado que a extensão EWKT não é um formato padrão, só é possível utilizá-la em bases de dados do PostGIS.

```
INSERT INTO ufma  VALUES
        (4,ST_GeomFromText('POLYGON ((-44.3109882 -2.5559354,-44.3151081 -2.5611445,
                        -44.3142283 -2.5644456,-44.3111169 -2.5650244,
                        -44.307555 -2.5639312,-44.3069541 -2.5583149,
                        -44.3031776 -2.5551423,-44.3058813 -2.5504263,
                        -44.3109882 -2.5559354))',4326),
'Contorno');
```

2.1.2 Operações espaciais

As consultas espaciais baseiam-se em relacionamentos espaciais de vários tipos, como métricos, direcionais e topológicos. Além disso, vale destacar que nesse fascículo estamos tratando das operações sobre geo-objetos. Nesse contexto, RIGAUX et al. (2003) classificaram as operações sobre geo-objetos em sete categorias:

- Operação unária booleana: mapeia geometrias em valores booleanos como verdadeiro e falso.
- Operação unária escalar: mapeia geometrias em valores escalares.
- Operação unária espacial: mapeia geometrias em geometrias.
- Operação binária booleana: também chamada de relacionamento espacial, mapeia pares de geometrias em valores booleanos. Pode ainda ser subdividida em relacionamento topológico, direcional e métrico. No topológico, o relacionamento não é alterado por transformações como translação, rotação e escala. No direcional, o relacionamento expressa uma noção de direção, como acima, abaixo, ao sul, etc. E no relacionamento métrico, expressa-se a noção de distância entre geometrias.
- Operação binária escalar: mapeia pares de geometrias em valores escalares, como por exemplo, distância entre duas geometrias.
- Operação binária espaciais: mapeia pares de geometrias em outras geometrias, como por exemplo, operações de união e intersecção.
- Operação n-ária espacial: mapeia várias geometrias em outras geometrias.

A SF-SQL definiu várias operações a serem suportadas pelos bancos de dados. O PostGIS implementa centenas dessas operações que podem ser agrupadas de diferentes maneiras. As mais básicas, são aquelas que realizam a construção ou a impressão das geometrias a partir de formatos abertos como KML(Keyhole Markup Language), GeoJSON, WKT e SVG. Por exemplo, dado a geometria do contorno da UFMA inserida anteriormente, ela poderia ser impressa no formato KML como a seguir:

```
select st_askml(geom) from ufma;
```

Que teria como saída:

```
<Polygon>
<outerBoundaryIs><LinearRing>
<coordinates>
-44.310988199999997,-2.5559354 -44.315108100000003,-2.5611445 -44.314228300000003,-2.5644456 -
44.311116900000002,-2.5650244 -44.307555000000001,-2.5639312 -44.306954099999999,-2.5583149 -
44.303177599999998,-2.5551423 -44.305881300000003,-2.5504263 -44.310988199999997,-2.5559354
</coordinates>
</LinearRing></outerBoundaryIs>
</Polygon>
```

Para outros formatos, pode-se usar as seguintes operações:

- ST_AsBinary: retorna a representação do binário bem conhecido (WKB) da geometria/geografia sem os metadados SRID.
- ST_AsEWKT: retorna a representação de texto bem conhecida (WKT) da geometria com os meta dados SRID.
- ST_AsText: retorna a representação de texto bem conhecida (WKT) da geometria/geografia sem os metadados do SRID.
- ST_AsSVG: retorna uma geometria em dados SVG path, dado um objeto de geometria ou geografia.

Atenção!

O ST presente nos nomes das operações refere a *spatial-temporal*, ou seja, operações que lidam com as dimensões de espaço e tempo.

A função ST_GeomFromText usada anteriormente é um exemplo de construtor que usa o formato WKT como entrada, mas poderia ser usado outros formatos, por exemplo, o KML:

```
INSERT INTO ufma  VALUES
            (4,ST_GeomFromKML('<Polygon>
        <outerBoundaryIs><LinearRing>
```

```
      <coordinates>
        -44.310988199999997,-2.5559354 -44.315108100000003,-2.5611445
        -44.314228300000003,-2.5644456 -44.311116900000002,-2.5650244
        -44.307555000000001,-2.5639312 -44.306954099999999,-2.5583149
        -44.303177599999998,-2.5551423 -44.305881300000003,-2.5504263
        -44.310988199999997,-2.5559354
      </coordinates>
    </LinearRing></outerBoundaryIs>
  </Polygon>'),
'Contorno');
```

Outras funções incluem:

- ST_GeomFromGeoJSON: utiliza como entrada uma representação geojson de uma geometria e como saída um objeto de geometria PostGIS.
- ST_GeomFromEWKT: retorna um valor ST_Geometry específico da representação de texto estendida bem conhecida (EWKT).
- ST_GeomFromGML: utiliza como entrada uma representação GML de geometria e como saída um objeto de geometria PostGIS.

Em diversas aplicações é comum a necessidade de identificar as relações topológicas entre os objetos. Por exemplo, para saber quais municípios são cruzados por um dado rio, ou ainda, quantos lotes estão dentro de uma dada quadra:

```
SELECT COUNT(*)
FROM lotes lt, quadras qd
WHERE ST_Contains(qd.geom, lt.geom)
AND qd.nome = 'Q1';
```

As operações topológicas mais comuns são:

- ST_Disjoint: verifica se as geometrias não se "intersectam espacialmente", ou seja, se elas não dividem nenhum espaço.
- ST_Contains: verifica se nenhum ponto de B estiver no exterior de A, e pelo menos um ponto do interior de B estiver no interior de A.

- ST_Equals: verifica se as geometrias representam a mesma geometria. A direcionalidade é ignorada.
- ST_Intersects: verifica se as geometrias se intersectam espacialmente.
- ST_Touches: verifica se as geometrias têm pelo menos um ponto em comum, mas seus interiores não se intersectam.
- ST_Crosses: verifica se as geometrias fornecidas têm alguns, não todos, pontos em comum.
- ST_Transform: retorna uma nova geometria com suas coordenadas transformadas em diferentes referências espaciais.

Algumas operações métricas muito utilizadas são:

- ST_Perimeter: retorna o comprimento do limite de uma geometria. A medição das unidades de geometria está na referência espacial e a da geografia em metros.
- ST_Distance: retorna a distância cartesiana 2D entre duas geometrias em unidades projetadas (baseado em referência espacial).
- ST_Area: retorna a área da superfície se ela for um polígono ou multipolígono na unidade especificadas pelo SRID.

Saiba mais!

Em https://postgis.net/docs/reference.html#Spatial_Relationships são detalhados várias operações espaciais.

2.1.3 Indexação espacial

Os gerenciadores de banco de dados são conhecidos por fornecer respostas rápidas a consultas complexas. Porém, para isso ocorrer, os dados devem ser indexados de modo eficiente. Os bancos de dados convencionais utilizam estruturas de dados específicas, mas que não são eficientes para dados espaciais. O PostGIS

fornece uma indexação eficiente para dados espaciais que foi implementada sobre o GIST (Generalized Search Tree) existente no PostgreSQL. A sintaxe básica para criação de um índice é a seguinte:

```
CREATE INDEX sp_idx_name ON nome_tabela
USING GIST (coluna_geometrica);
```

Considerando uma tabela com as sedes dos municípios e outra de estados, podemos criar índices em ambas como a seguir:

```
CREATE INDEX sedes_gix ON sedes USING GIST (geom);
CREATE INDEX estados_gix ON estados USING GIST (geom);
```

Os índices espaciais são usados em consultas que envolvam predicados espaciais apresentados na Seção 2.1.2.

2.2 Integração entre banco de dados geográficos e SIGs

Antes de apresentar a integração entre banco de dados geográficos e SIGs, a Seção 2.2.1 apresenta como instalar e usar um sistema de informação geográfica, especificamente, o QGIS.

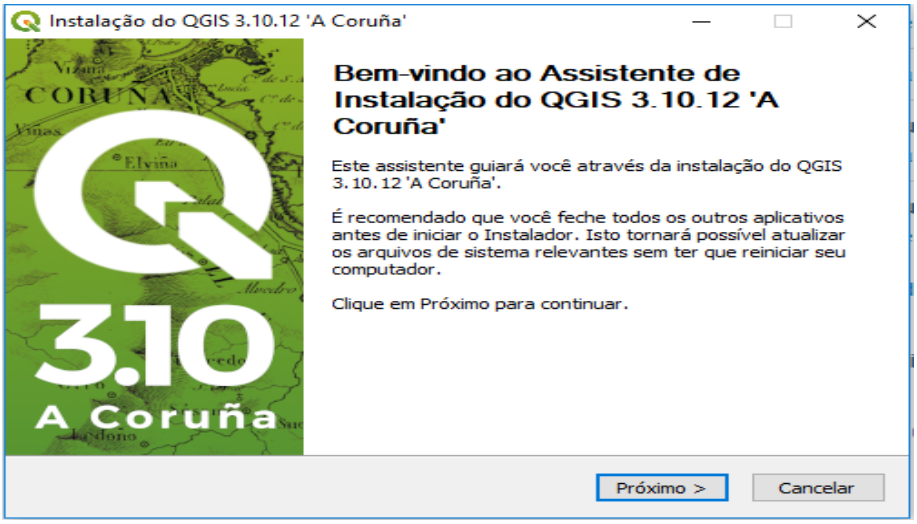
2.2.1 Instalando e usando um sistema de informação geográfica

Esta seção apresenta um breve tutorial sobre como instalar o QGIS. Primeiramente, vá em https://qgis.org/pt_BR/site/forusers/download.html baixe a versão 3.10 para o sistema operacional desejado. A Figura 39 apresenta a versão indicada para o sistema operacional Windows.

Figura 39: Baixando o instalador do QGIS.

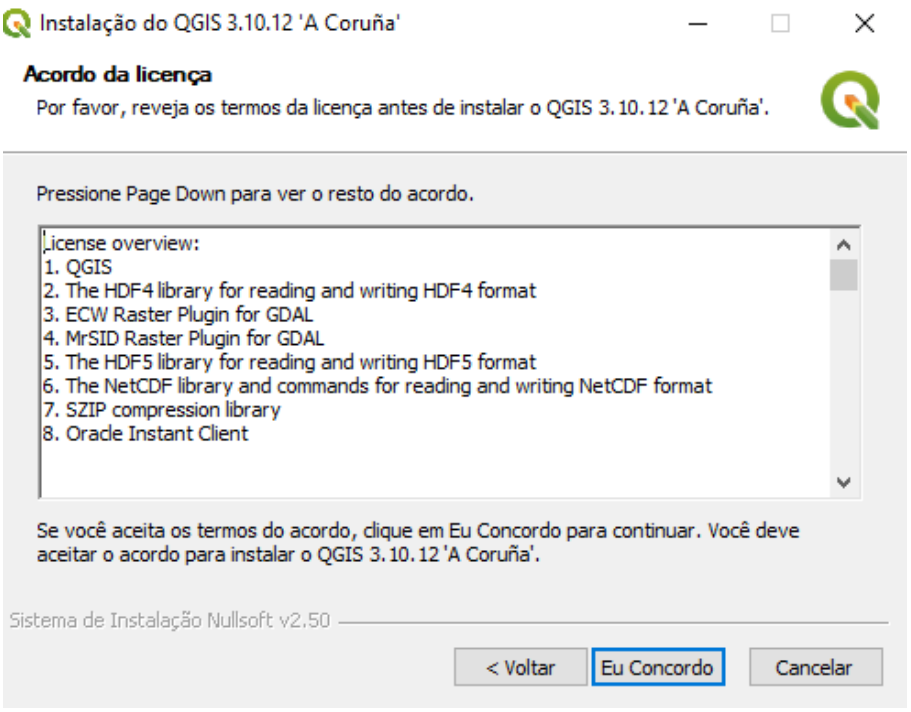
Após baixar, podemos executar o instalador que, nesse caso, é similar a qualquer outro instalador para o Windows como pode ser visto na Figura 40.

Figura 40: Tela inicial do instalador do QGIS



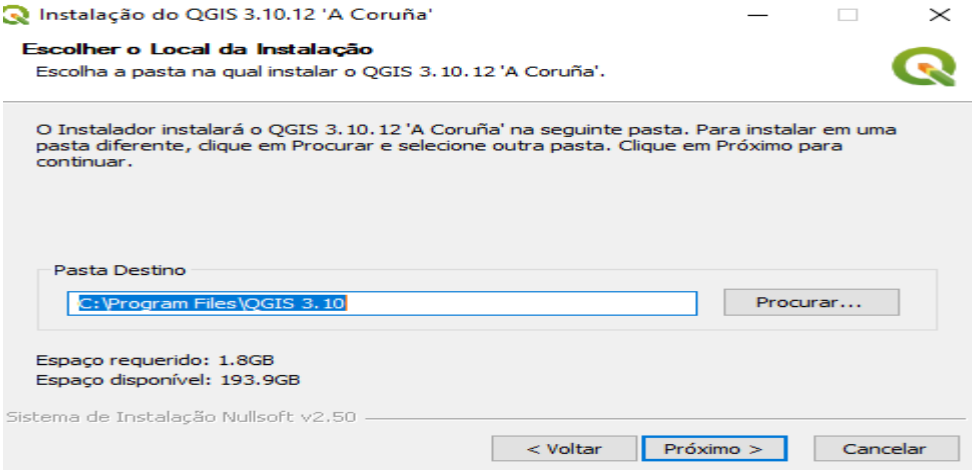
Em uma tela a seguir, será necessário concordar com a licença de uso.

Figura 41: Tela de licença do QGIS



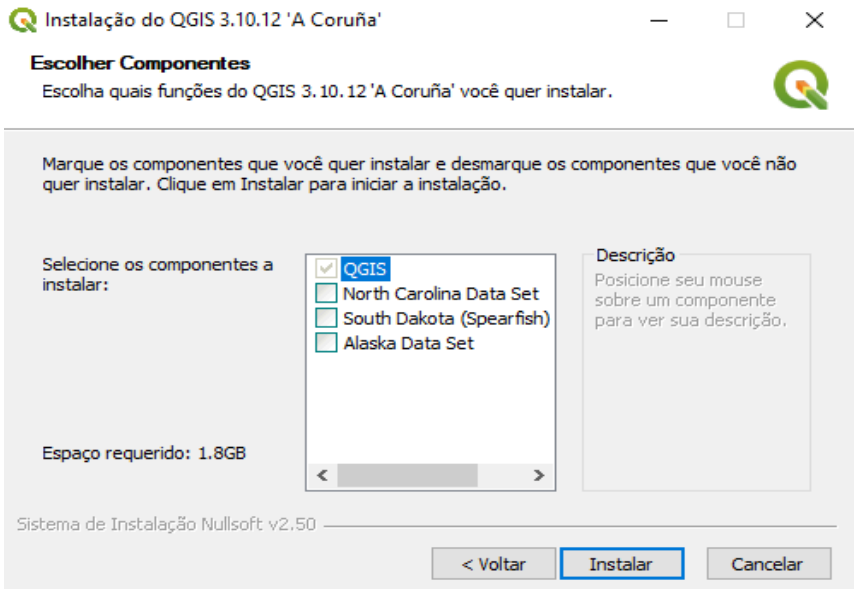
Geralmente, podemos usar a pasta padrão, como na Figura 42.

Figura 42: Local de instalação do QGIS



A Figura 43 apresenta a tela para seleção dos componentes a serem instalados. Não é necessário instalar os dados de exemplo.

Figura 43: Componentes a serem instalados



Após a instalação, podemos testar o QGIS. Para isso baixem os seguintes dados https://bit.ly/3b6qN8s e descompacte-os em alguma pasta do teu computador. Então, em seguida inicie o QGIS e no menu Projeto, selecione abrir para selecionar o projeto denominado de "projetoexemplo".

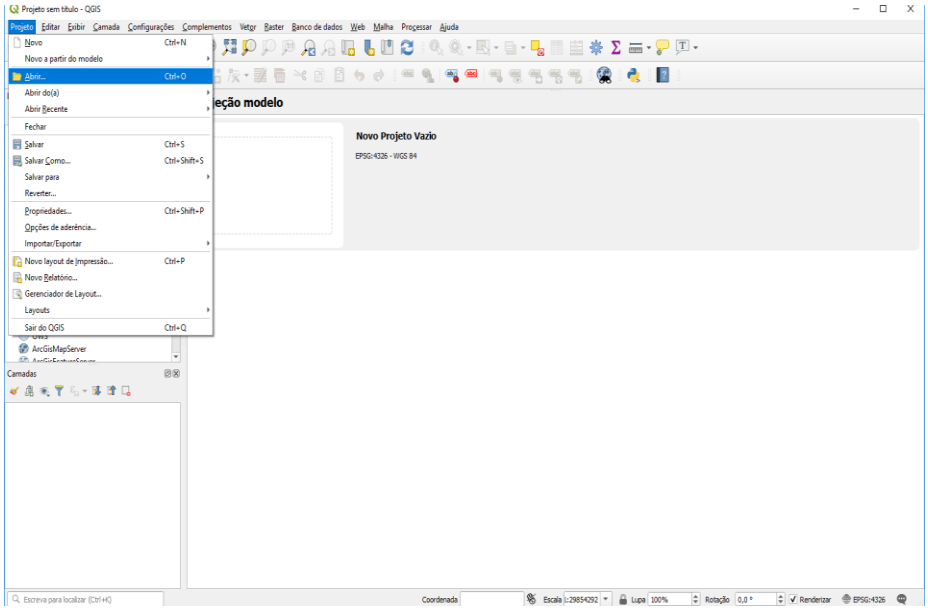
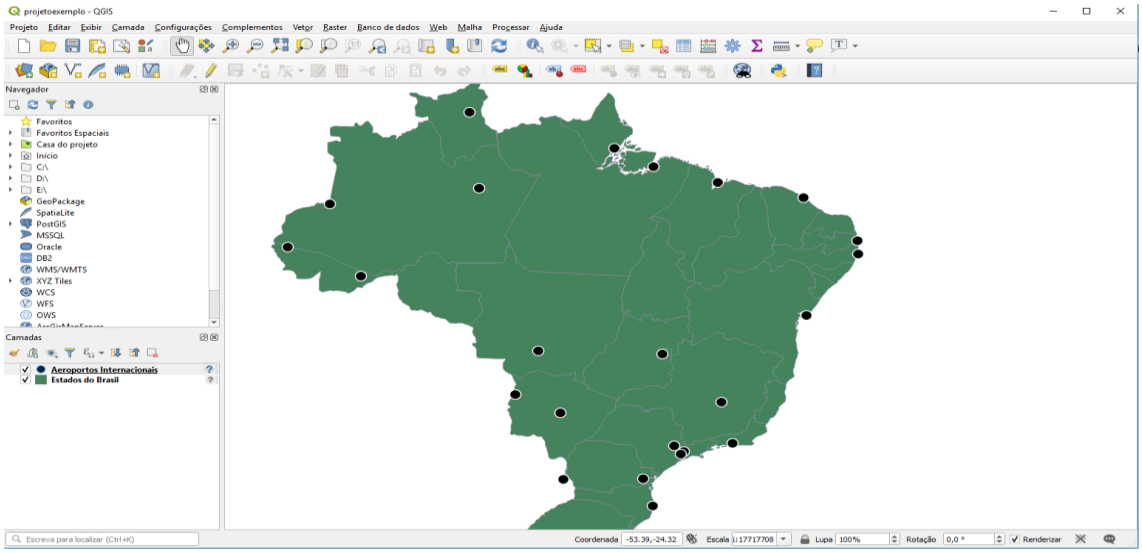


Figura 44: Tela principal do QGIS

Após abrir o projeto, você deverá ver o limite político do Brasil com as sedes dos aeroportos internacionais como na Figura 45.

Figura 45: Projeto exemplo para teste da instalação do QGIS

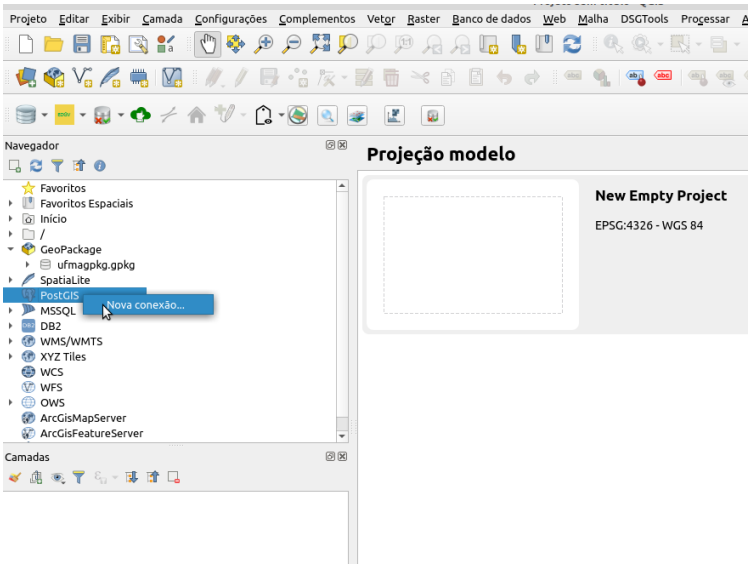


Esse rápido tutorial teve como objetivo apenas ajudá-los na instalação e iniciação com o QGIS. Na próxima Seção seguiremos com a integração entre o QGIS e banco de dados geográfico.

2.2.2 Conectando a um banco de dados

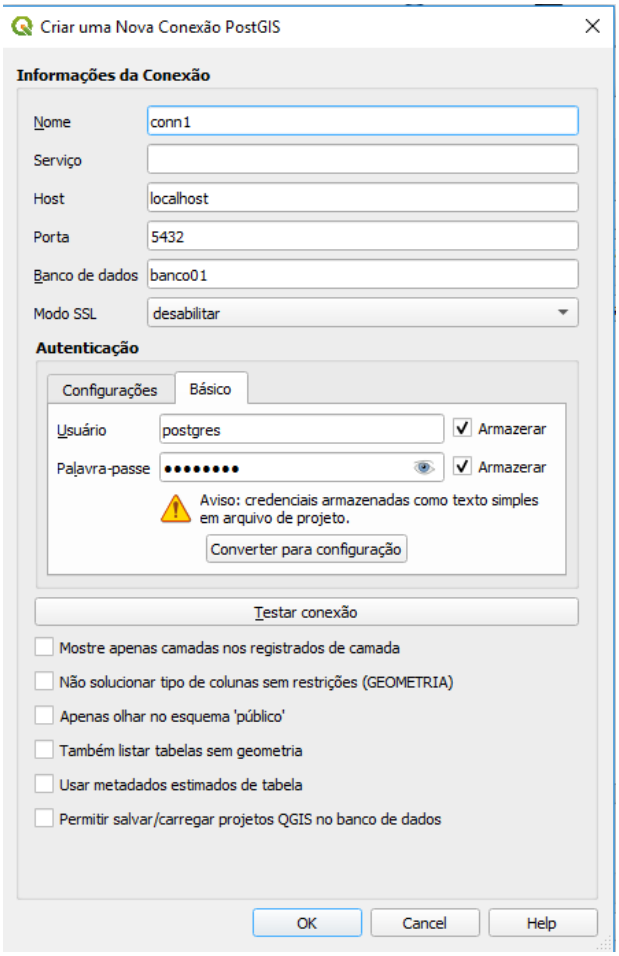
Como vimos na Seção 1.1.1, um sistema gerenciador de banco de dados (SGBD) utiliza uma arquitetura denominada de cliente/servidor. Nesse contexto, um sistema de informação geográfica (SIG) desempenha o papel de cliente enquanto o PostgreSQL executa em modo servidor. Então, dado um banco de dados já criado, o próximo passo é criar uma conexão entre o SIG (cliente) e o SGBD (servidor). No QGIS existem vários caminhos para conseguir criar uma conexão. Na Figura 46, mostra um caminho onde a conexão é criada clicando com o botão direito sobre o ícone do PostGiS no navegador:

Figura 46: Criando uma conexão com o PostGIS



Então, será aberta uma nova janela para entrar com as informações necessárias para a criação da conexão, Figura 47. O nome da conexão deve ser escolhido de forma que facilite encontrar essa conexão no futuro. Em host, deve-se entrar com o endereço do servidor ou localhost caso o SGBD esteja instalado no mesmo computador. Para o nome do banco de dados, deve-se digitar um nome de um banco já existente. Por fim, deve-se entrar com o usuário e senha.

Figura 47: Janela de configuração de uma conexão com o PostGIS



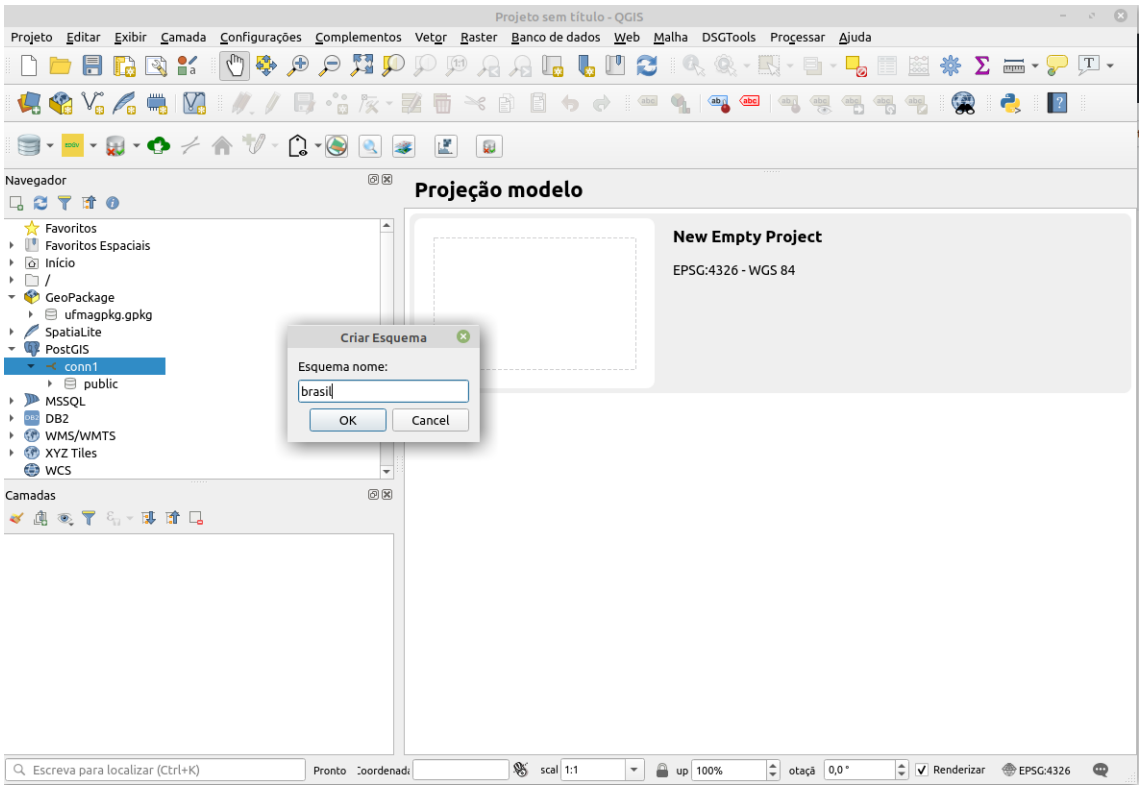
Atenção!

Na Figura 47 foi selecionado as opções para armazenar a senha e o usuário. Essa prática é interessante para evitar ter de digitar o usuário e a senha em outros momentos. Porém, use apenas em trabalhos pessoais e em seu computador. Nunca em ambiente corporativo ou institucional.

2.2.3 Importando e visualizando dados vetoriais

Após criar uma conexão com um banco de dados geográfico, pode-se inserir dados a partir de formatos de intercâmbio de dados ao invés de inserir diretamente a partir do SQL como foi feito na Seção 2.1. Para isso, pode-se clicar no ícone da conexão criada anteriormente e com o botão direito do mouse selecionar "Criar Esquema". É uma boa prática inserir dados em um esquema diferente do público, que é usado pelo PostgreSQL para armazenar alguns metadados. A Figura 48 mostra a criação de um esquema denominado de "brasil".

Figura 48: Criando um novo esquema

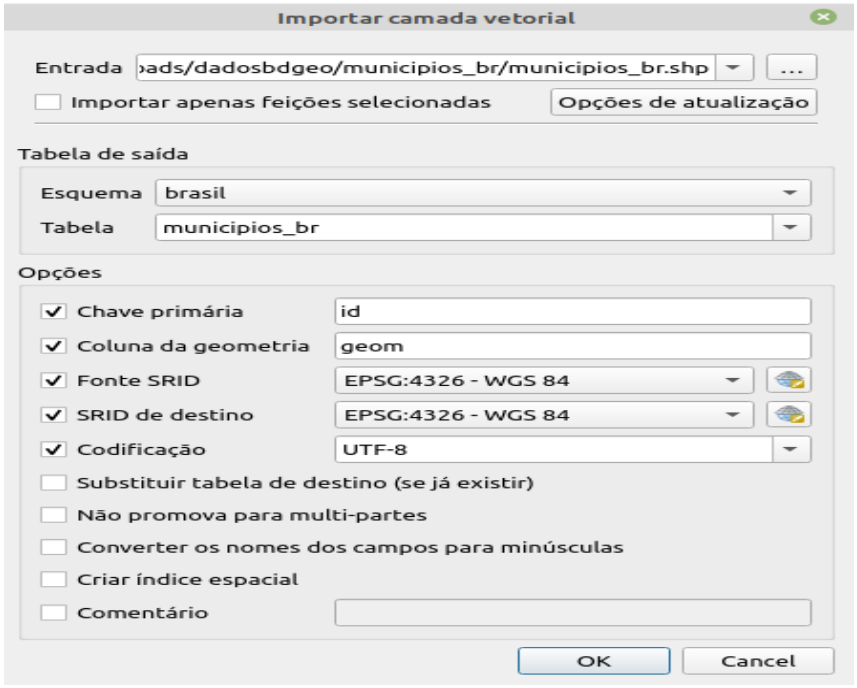


Após a criação do novo esquema, acesse o "Gerenciar banco de dados" no menu "Banco de Dados". Então, clique na conexão criada anteriormente e selecione o esquema Brasil, como na Figura 49.

Figura 49: Janela do gerenciador de banco de dados

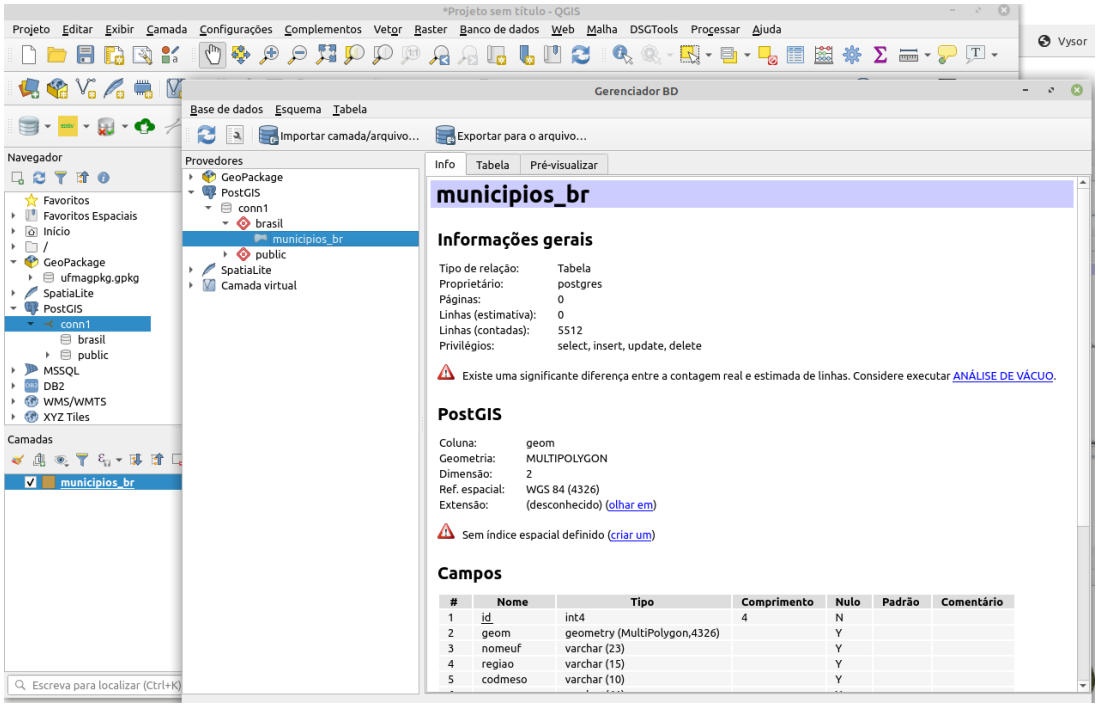
Na janela da Figura 49 basta clicar no ícone “Importar layer”. Na janela que irá surgir, selecione o layer municípios br e preencha os demais campos como na Figura 50.

Figura 50: Janela de importação de dados vetorial



Após a importação, você deverá ver a nova tabela criada como na Figura 51. Então, ao clicar nessa tabela o dado será adicionado a interface do QGIS.

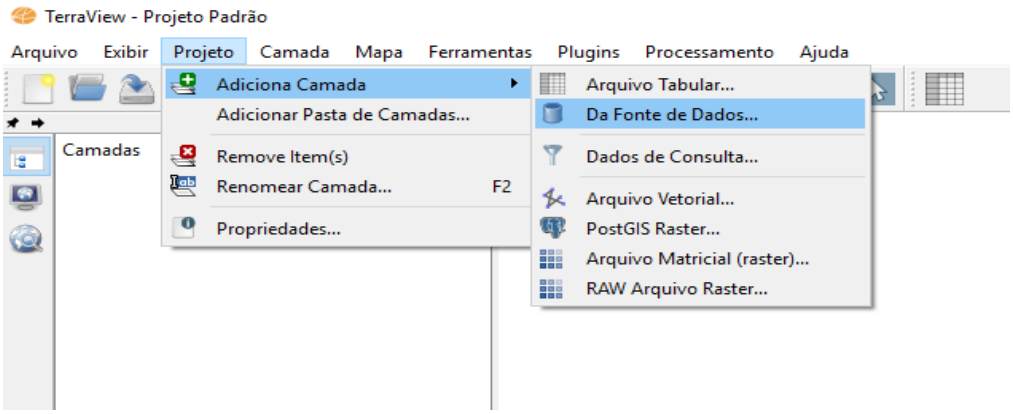
Figura 51: Informações sobre o dado importado.



2.2.4 Integrando com outros SIGs

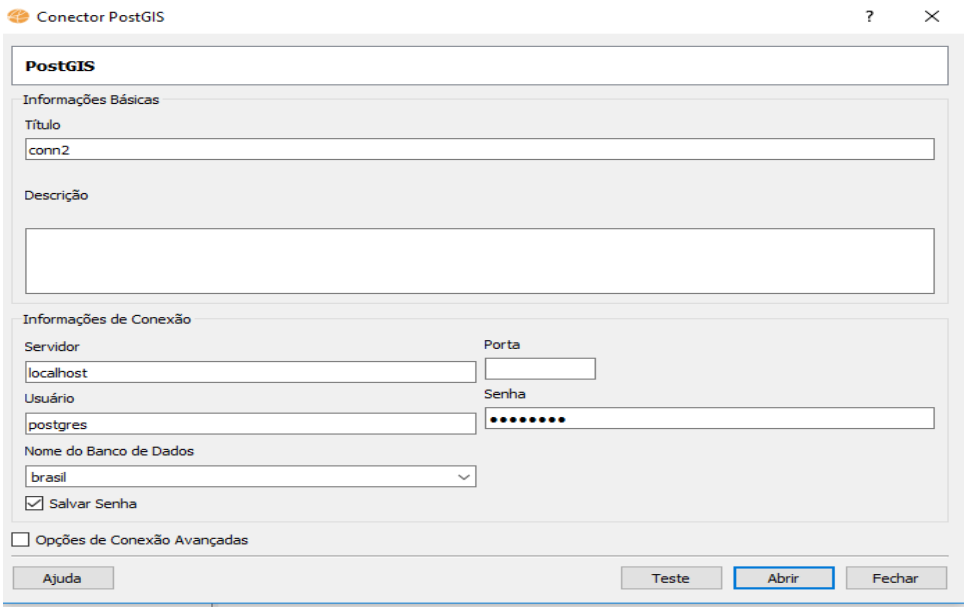
Uma vantagem de usar um banco de dados geográfico como o PostGIS é a possibilidade de acessar os mesmos dados através de diferentes SIGs, dado que eles não estão em um formato específico. Por exemplo, a Figura 52 apresenta a tela principal do TerraView, que é desenvolvido pelo Instituto Nacional de Pesquisas Espaciais.

Figura 52: Tela principal do Terraview



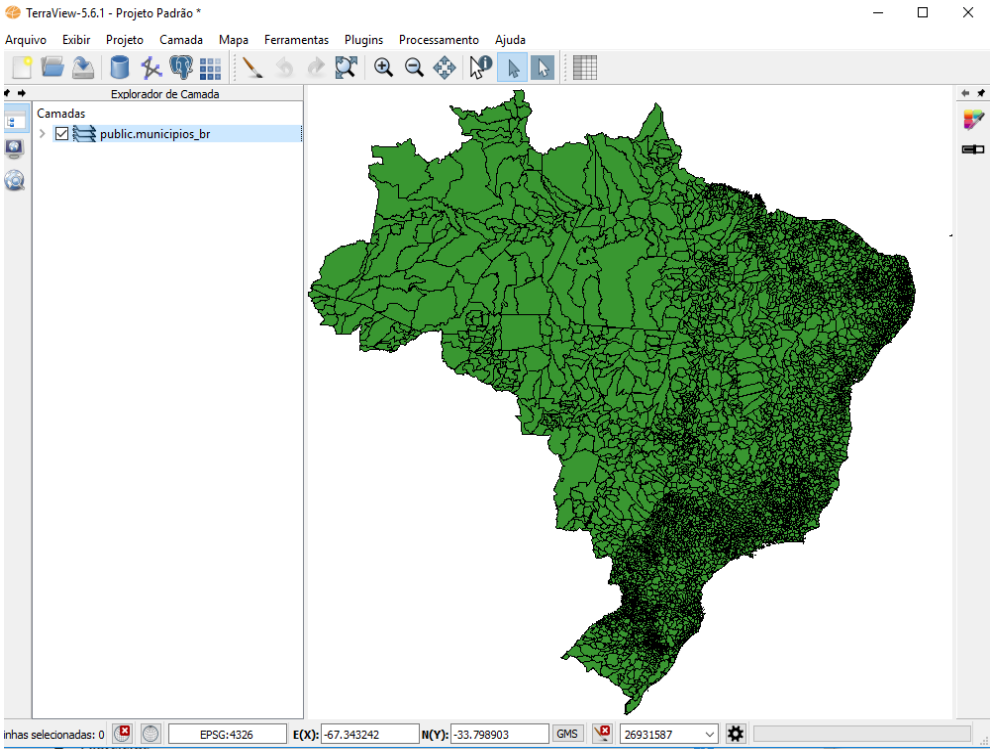
A partir da tela principal, podemos adicionar uma nova camada a partir do menu projeto como na Figura 52. Em seguida, na janela "Seletor Fonte de Dados" selecionamos Postgis para que seja aberta a janela onde entraremos entre com as informações para conexão com o PostGIS. Observe que a janela de configuração da Figura 53 é similar a do QGIS que vimos na Seção 2.2.2.

Figura 53: Tela do conector do PostGIS



Após configurado, clicamos em abrir e deverá ser mostrado o mesmo dado que foi importado na Seção 2.2.3, como na Figura 54.

Figura 54: Visualizando um dado do PostGIS através do Terraview



ATIVIDADE DE APRENDIZAGEM DA UNIDADE 2

1) Explique as duas formas de integrar banco de dados geográficos e sistemas de informação geográfica.

2) Considere um banco de dados espacial de uma dada cidade, que possui as seguintes entidades: bairro e lotes.

```
CREATE TABLE bairro( codigo    integer PRIMARY KEY , bairroNm varchar(20),
geom  geometry(POLYGON, 0));
CREATE TABLE lotes( codigo    integer PRIMARY KEY ,loteNm varchar(20), geom
geometry(POLYGON, 0));
```

Qual a consulta SQL que retorna todos os lotes localizados dentro do bairro Turu?

3)Baixem os arquivos shape, localizados em https://bit.ly/3b6qN8s. Descompacte-os em uma pasta e depois façam as seguintes atividades:

a)Crie um banco de dados geográfico no PostGIS denominado brdata.

b)Importe os arquivos sede_municipio.shp, aeroporto_internacional.shp e ferrovia.shp.

c)Escreva e teste uma consulta que retorne os municípios que estão dentro de um raio de 100km do aeroporto de São Luiz?

d)O nome dos municípios em que suas sedes estão a pelo menos 100KM de distância de algum aeroporto.

e) Qual o tamanho em quilômetros da rede ferroviária do Estado do Maranhão?

Banco de dados geográficos é a base dos sistemas de informação geográfica, como destacado em vários momentos neste fascículo. Mas isso não o torna um tópico fácil, mas talvez um dos mais difíceis para usuários destes sistemas. Isso se deve ao fato de não ser uma área mais conhecida para quem tem alguma formação na área de computação. Além disso, os sistemas de informação geográfica mais populares permitem aos usuários postergarem o uso de banco de dados geográficos. Esses sistemas permitem aos usuários trabalharem diretamente com os arquivos como fonte de dados. Entretanto, usuários mais avançados e que trabalham em ambientes institucionais e corporativos logo perceberão que é preciso migrar para um banco de dados geográficos.

Então, esse fascículo buscou ser uma “porta de entrada” para usuários de sistemas de informação geográfica que estão tendo um primeiro contato com banco de dados. Está longe de esgotar o assunto, mas ele conseguiu englobar vários aspectos importantes como: conceitos básicos, projeto e modelagem, linguagem de consulta, arquitetura e integração com sistemas de informação geográfica.

REFERÊNCIAS

BORGES, K. A. V.; DAVIS JR., C. A.; LAENDER, A. H. F. OMT-G: an object-oriented data model for geographic applications. **GeoInformatica**, v. 5, n.3, p. 221-260, 2001.

CÂMARA, G. (2005). Representação computacional de dados geográficos. In G. D. Casanova, M. A., Câmara, G., Davis, C., Vinhas, L., & Queiroz (Ed.), ***Bancos de Dados Geográficos*** (1st ed., pp. 1–44). Mundogeo. http://www.dpi.inpe.br/livros/bdados/index.html

CÂMARA, G., DAVIS, C., MONTEIRO, A. M. V., & MEDEIROS, J. S. (2001). **Introdução à ciência da geoinformação.** Disponível em: https://www.dpi.inpe.br/gilberto/livro/introd/

DSG. DIRETORIA DO SERVIÇO GEOGRÁFICO – EXÉRCITO BRASILEIRO. ET-EDGV 3.0 – **Especificação Técnica para a Estruturação dos Dados Geoespaciais Vetoriais**, 2017. Disponível em: http://www.geoportal.eb.mil.br/index.php/inde2?id=142.

ELMASRI, RAMEZ; NAVATHE, SHAMKANT B. **Sistemas de Banco de Dados**. 4. ed. São Paulo, Pearson Addison Wesley, 2005.

FERREIRA, K. R., CASANOVA, M. A., QUEIROZ, G. R., & OLIVEIRA, O. (2005). Arquiteturas e linguagens. In M. A. Casanova, G. Câmara, Clodoveu A Davis, L. Vinhas, & G. R. de Queiroz (Eds.), ***Banco de dados geográficos***. MundoGEO.

GÜTING, RALF HARTMUT. An introduction to spatial database systems. **The VLDB Journal—The International Journal on Very Large Data Bases**, v. 3, n. 4, p. 357-399, 1994.

HERRING, J. (2010). Opengis implementation standard for geographic information-simple feature access. *Open Geospatial Consortium, Inc*.

HEUSER, CARLOS ALBERTO. **Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS**. Bookman Editora, 2009.

LONGLEY, PAUL A. et al. **Geographic information science and systems**. John Wiley & Sons, 2015.

RIGAUX, P., SCHOLL, M., VOISARD, A., & WIEGAND, N. (2003). **Spatial Databases with Application to GIS**. *SIGMOD Record*. https://doi.org/10.1145/959060.959081

RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W., **Object-Oriented Modeling and Design**, Prentice-Hall, 1991.

SHEKKAR, S. AND S. CHAWLA (2003). **Spatial databases - a tour**. Upper Saddle River, NJ, USA, Prentice-Hall.