

Modelos para Informação Espacial

O que é um modelo?

Definição de Dicionário (*e.g., Porto Editora*)

imagem ou desenho que representa o objecto que se pretende reproduzir esculpindo, pintando ou desenhando;

esquema teórico em matéria científica representativo de um comportamento, de um fenómeno ou conjunto de fenómenos;

representação miniatura de um sistema que permita analisar as suas propriedades de interesse.

... e o que é um modelo conceptual (de dados)?

1. é o que resulta de se executarem directivas SQL da DDL.
2. é o que resulta de se executarem directivas SQL da DML.
3. é o que se obtém ao desenhar um diagrama entidade-associação
4. é a forma mais simples de especificar chaves primárias e estrangeiras
5. é o conjunto de todos os dados armazenados numa colecção de tabelas

Que opção, ou opções, melhor descrevem a ideia de modelo conceptual?

Modelo – conceptual e lógico – no ambiente Relacional

- Modelo conceptual
 - independente de qualquer possível implementação
 - descreve as entidades (objectos) de interesse
 - descreve as relações entre essas entidades
- Modelo lógico
 - resultado de traduzir o modelo conceptual num esquema relacional
 - a tradução segue regras bem definidas
 - a tradução recorre à DDL (*Data Definition Language*) do SQL

Em que se suporta um modelo (lógico) de dados?

- Relacional – suporta-se num conceito muito simples:
 - esquema de relação (em geral designado por tabela)
- Espacial – suporta-se em três conceitos essenciais:
 - Tema (*Theme*)
 - Objecto-Geográfico (*Geographic-Object*)
 - Componente-Espacial (*Spatial-Component*)

```
theme = { geographic-object }
```

```
geographic-object = ( description, spatial-component )
```

```
                  | ( description, geographic-object )
```

objecto atómico

objecto complexo

Modelo de Dados Espacial – *Theme*

- Tema (*Theme*)
 - idêntico à relação do modelo relacional, i.e., tem instâncias de esquema
 - e.g., rios, freguesias, cidades, países
- Instância do Esquema de Tema (*Theme Schema*)
 - conjunto dos Objectos-Geográficos que materializam o tema em causa
 - em linguagem corrente, para simplificar, designa-se apenas por tema
- ... mapa
 - resultado de se representar um tema sobre papel, ou num monitor
 - inclui, entre outros aspectos, cores, uma escala, uma legenda
 - e.g., mapas de estradas, de caminhos de ferro, de estado atmosférico

Primeiro passo no desenho de um modelo espacial:

Construir um esquema conceptual para cada tema (*theme*) de interesse

Modelo de Dados Espacial – *Geographic-Object*

- Objecto-Geográfico (*Geographic-Object*)
 - ao nível conceptual é o objecto essencial a ser considerado
 - corresponde a uma entidade do mundo real
- ... inclui dois aspectos:
 - Descrição (*Description*)
 - Componente-Espacial (*Spatial-Component*)
- ... Descrição (*Description*)
 - lista ordenada de atributos (tal como no relacional)
 - são também designados por atributos alfa-numéricos
 - e.g., “nome” e “densidade populacional” de uma cidade

Um tema (*theme*) é:

uma colecção de objectos-geográficos (*geographic-objects*)

Modelo de Dados Espacial – *Spatial-Component*

- Componente-Espacial (*Spatial-Component*)
 - pode incluir geometria, e.g., localização geográfica, forma
 - pode incluir relações topológicas, e.g., adjacência
- ... e.g., uma “cidade” pode ter como valor do *spatial-component*:
 - um polígono no espaço 2D
- Objecto-Espacial (*Spatial-Object*), é usado para designar
 - apenas a Componente-Espacial de um Objecto-Geográfico
 - pode ser considerado isoladamente
 - e.g., por ser partilhado por diversos Objectos-Geográficos
 - e.g., uma fronteira entre dois, ou mais, países

Cada objecto-geográfico inclui uma componente-espacial

Uma componente-espacial isolada designa-se por objecto-espacial

Geographic-Object: atómico (atomic), complexo (complex)

- Entre as entidades geográficas do mundo real
 - existem composições intrínsecas
 - e.g., a Europa composição de Países; Portugal composição de Distritos
- Objecto-Geográfico Complexo (*Complex Geographic-Object*)
 - é composto por outros Objectos-Geográficos (complexos ou atómicos)
- Objecto-Geográfico Atómico (*Atomic Geographic-Object*)
 - não é composto por outro Objecto-Geográfico

e.g., no tema que corresponde às unidades administrativas da Europa o Objecto-Geográfico (complexo) “Portugal” é composto por 18 Objecto-Geográfico (atómico) “Distrito”, no Continente e 2 Objecto-Geográfico (atómico) “Região Autónoma” nas ilhas dos Açores e Madeira

Modelo Geográfico

```
theme = { geographic-object }  
geographic-object = ( description, spatial-component )  
                   | ( description, geographic-object )
```

- Um tema é um conjunto homogêneo de Objectos-Geográficos
 - i.e., objectos com a estrutura
- e.g., tema “Cidade” modelado por “nome”, “população” e “geometria”
 - descrição: “nome”, “população”
 - componente-espacial: geometria

Um modelo geográfico representa uma forma de descrever e manipular temas e seus objectos num sistema de gestão de bases de dados

Exemplo – tema, objecto-geográfico, componente-espacial

Cada país tem um nome, uma capital e um determinado número de habitantes; um país ocupa uma região geográfica.

Cada país organiza-se em unidades administrativas (unidAdm) que têm um nome e ocupam uma região geográfica (dentro do país)

Cada linguagem (ou família de linguagens) tem um nome e é falada num conjunto de regiões geográficas.

Que temas se podem identificar na descrição acima?

1. Um único tema – país e unidAdm associados à linguagem aí falada
2. Dois temas – tema A) país e unidAdm, tema B) linguagem
3. Três temas – tema A) país, tema B) unidAdm, tema C) linguagem

Manipulação do Modelo Geográfico – “Álgebra de Tema”

tema A	{	PAÍS(<u>nomeP</u> , capital, população, geo:regiao)
		UNID_ADM(<u>nomeR</u> , <u>nomeP</u> , geo:regiao)
tema B	{	LINGUAGEM(<u>nomeL</u> , geo:regiao)

- Manipulação através da “Álgebra de Tema” (*Theme Algebra*)
 - projecção (*projection*)
 - selecção (*selection*)
 - união (*union*)
 - sobreposição (*overlay*)
 - agregação (*merger*)
 - selecção geométrica (*geometric selection*) que inclui
 - ◊ *window-query, point-query e clipping*
- operadores idênticos aos da Álgebra Relacional

... “Tema” – possível concretização deste conceito

tema A { PAÍS(nomeP, capital, população, geo:regiao)
UNID_ADM(nomeR, nomeP, geo:regiao)
tema B { LINGUAGEM(nomeL, geo:regiao)

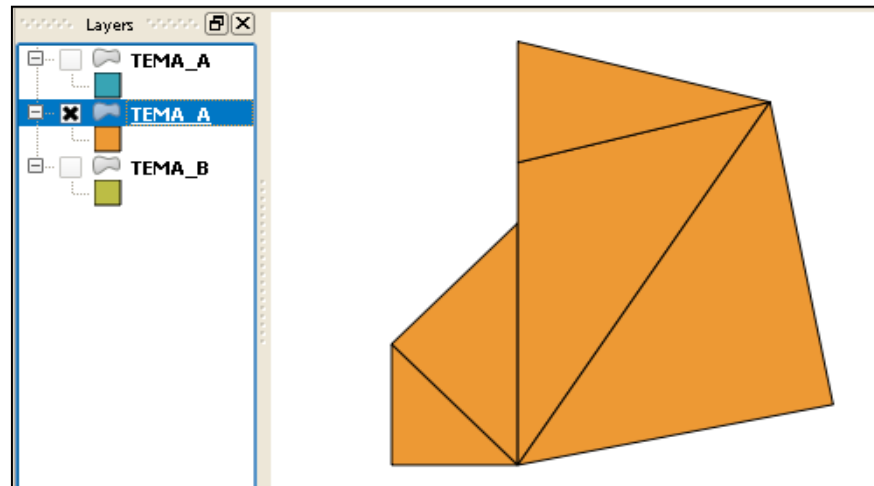
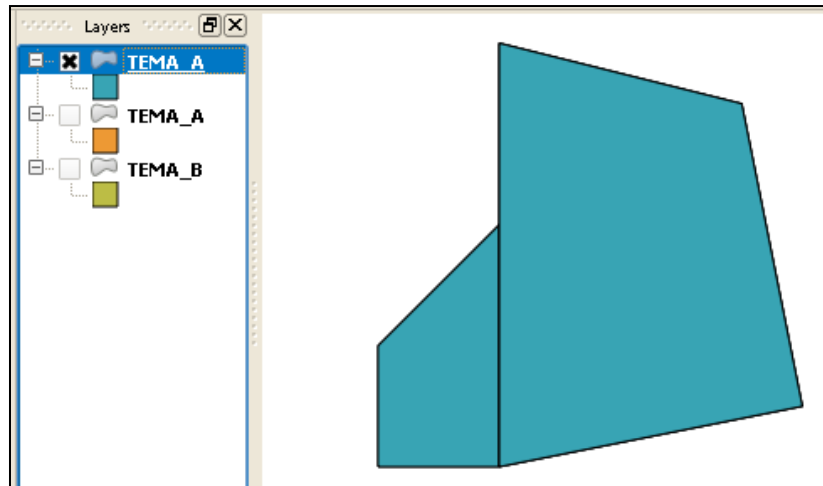
- cada “Tema” concretizado (representado) por uma vista
- cada vista podendo ter várias geometrias (logo “layers”)

```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

relação topológica a
satisfazer; notar que não
refere integridade referencial

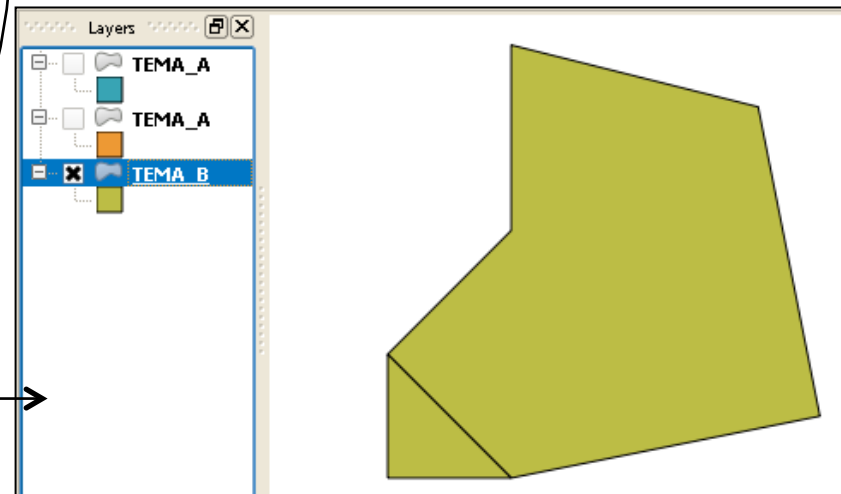
```
CREATE VIEW TEMA_B( nomeLinguagem, geo_regiao ) AS  
SELECT nomeLinguagem, geo_regiao  
FROM LINGUAGEM;
```

... possível concretização da noção de “Tema”



a componente espacial de
cada um dos temas (A e B):

- (A) país
- (A) unidade administrativa
- (B) linguagem



Projectão de Tema (*Theme Projection*)

$$\pi: \text{tema} \times \{ A_1, \dots, A_n \} \rightarrow \text{tema}$$

$\{ A_1, \dots, A_n \}$ é um subconjunto dos atributos da Descrição do tema;
 π devolve tema com mesma componente-espacial e atributos $\{ A_1, \dots, A_n \}$.

Seja:

$T \equiv$ instância do esquema do tema th , i.e. colecção de objectos-geográficos de th .

$\{ A_1, \dots, A_n \} \equiv$ atributos alfa-numéricos que descrevem th .

$geo \equiv$ componente-espacial de th .

Uma **projectão** sobre T escreve-se (mesma notação da álgebra relacional):

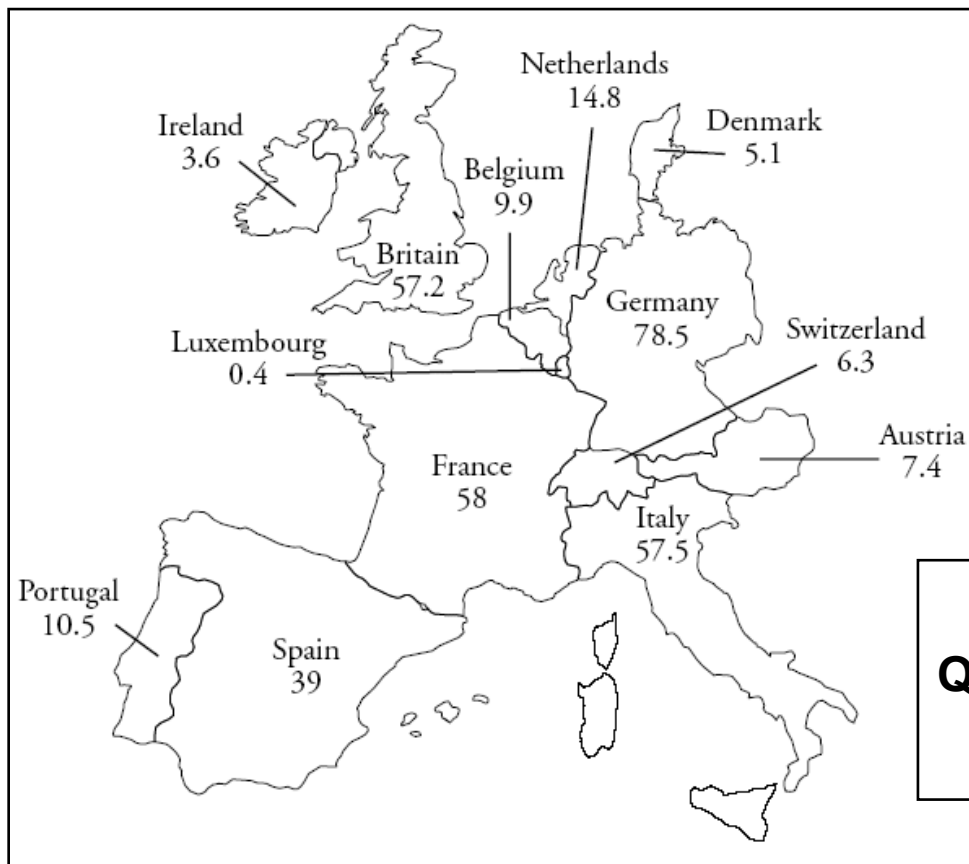
$$\pi_{A_1, \dots, A_n, geo}(T)$$

Projectão de Tema – exemplo

T \equiv instância do esquema do tema *países-da-europa*

A_1 \equiv nome; A_2 \equiv população

geo \equiv polígono representando as fronteiras de cada país



?

Qual o resultado da projecção:

$\pi_{A_2, geo}(T)$

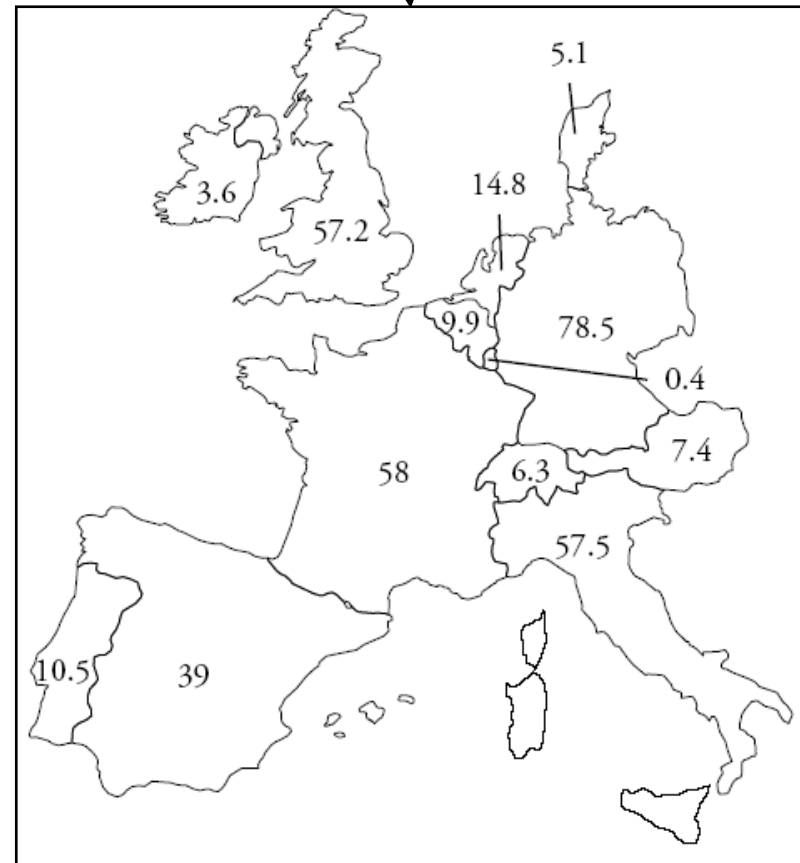
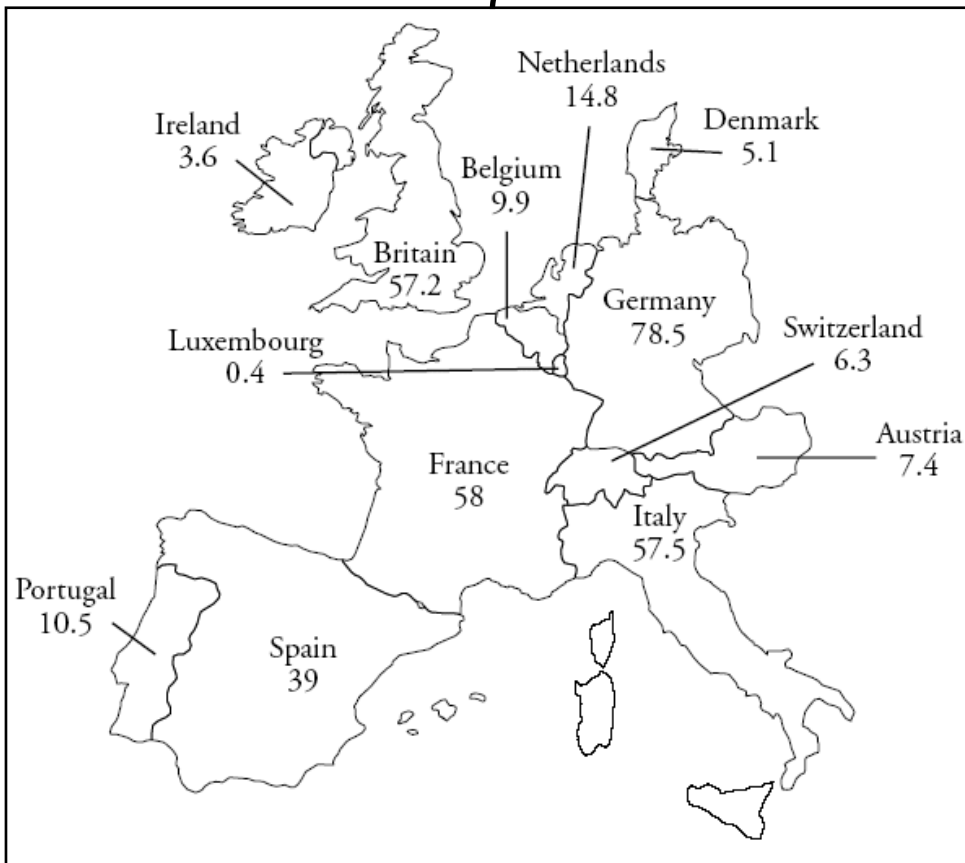
Projecção de Tema – exemplo (cont.)

T ≡ instância do esquema do tema *países-da-europa*

A₁ ≡ nome; A₂ ≡ população

geo ≡ polígono das fronteiras de cada país

$\pi_{A_2, \text{geo}}(T)$



... uma projeção usando o exemplo do “Tema A”

```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

π nomePais, geo_pais, geo_unidAdmin (TEMA_A)

```
SELECT nomePais, geo_pais, geo_unidAdmin  
FROM TEMA_A;
```

Seleccção de Tema (*Theme Selection*)

$$\sigma: \text{tema} \times p(A_1, \dots, A_n) \rightarrow \text{tema}$$

$p(A_1, \dots, A_n)$ é um predicado sobre a descrição do tema;
 σ mantém atributos e devolve tema com componente-espacial satisfazendo p .

Seja:

$T \equiv$ instância do esquema do tema th , i.e. colecção de objectos-geográficos de th .

$p(A_1, \dots, A_n) \equiv$ predicado sobre os atributos alfa-numéricos que descrevem th .

$geo \equiv$ componente-espacial de th .

Uma **selecção** sobre T escreve-se (mesma notação da álgebra relacional):

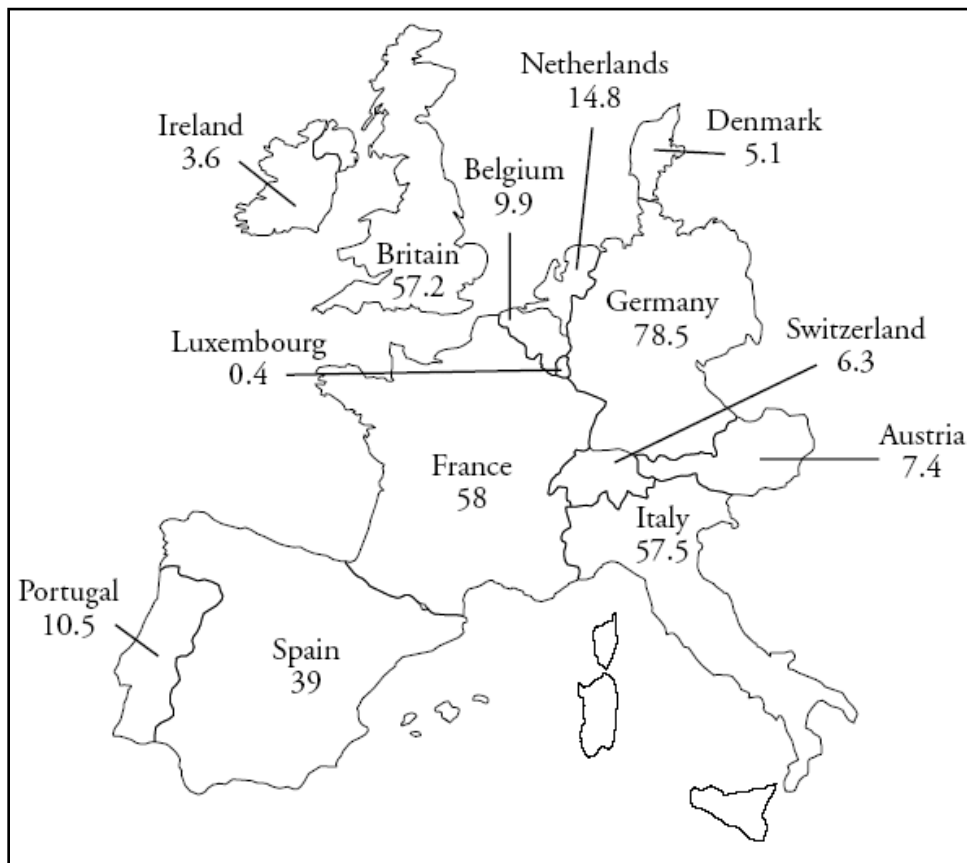
$$\sigma_{p(A_1, \dots, A_n)}(T)$$

Seleccção de Tema – exemplo

T ≡ instância do esquema do tema *países-da-europa*

A₁ ≡ nome; A₂ ≡ população

geo ≡ polígono representando as fronteiras de cada país



**Qual o nome e
população dos países
com 50 milhões (50M),
ou mais, habitantes?**

**i.e., qual o resultado da
projectção:**

$\sigma_{A2 \geq 50M} (T)$

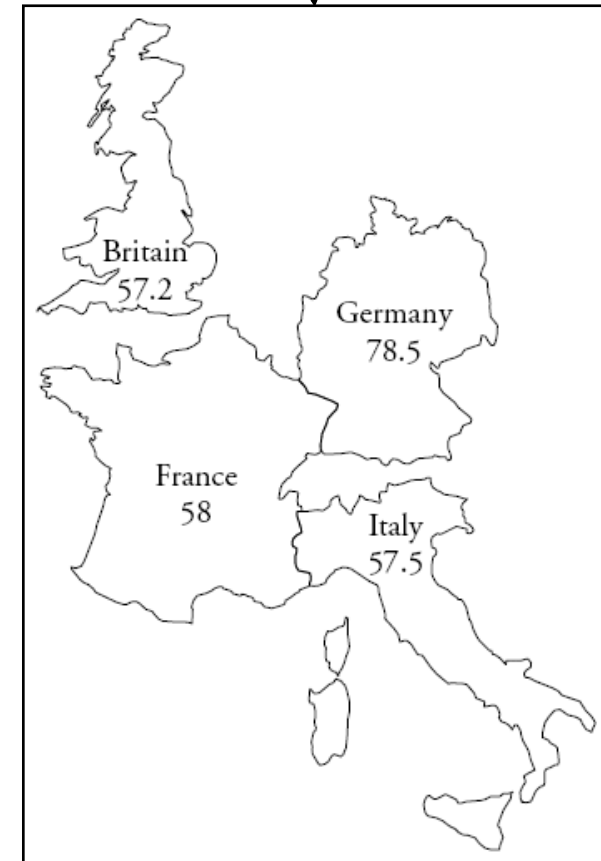
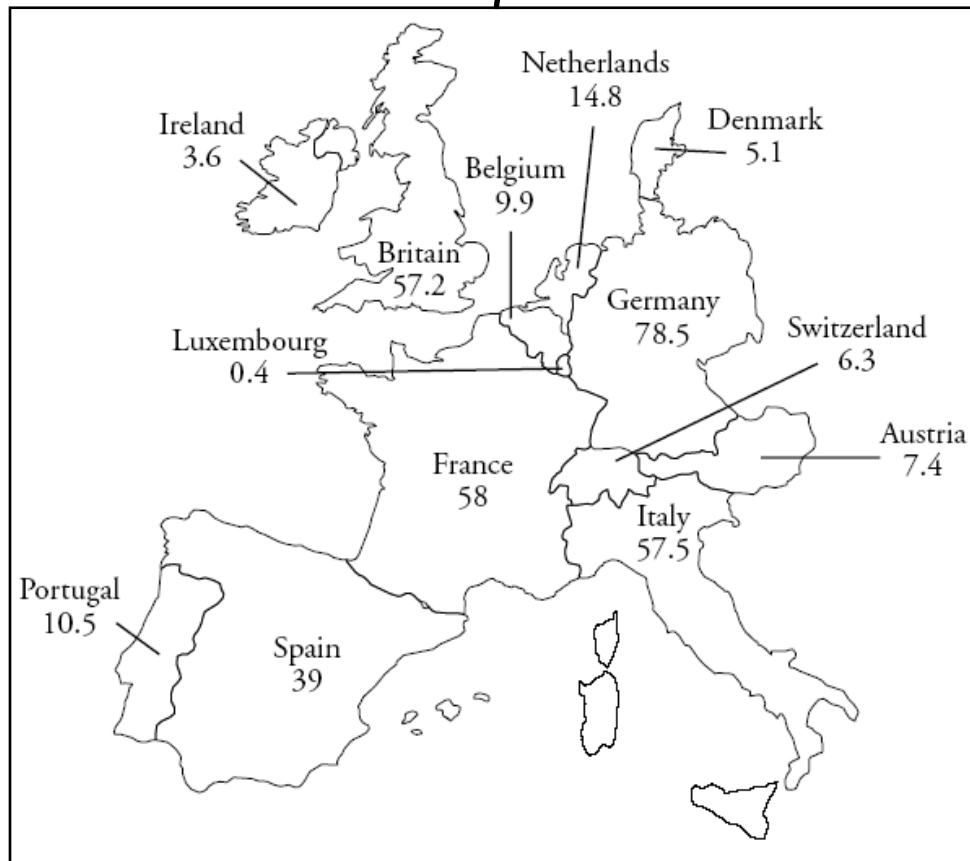
Seleccção de Tema – exemplo (cont.)

T ≡ instância do esquema do tema *países-da-europa*

A₁ ≡ nome; A₂ ≡ população

geo ≡ polígono das fronteiras de cada país

$\sigma_{A_2 \geq 50M}(T)$



... uma seleção usando o exemplo do “Tema A”

```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

$\sigma_{\text{capital}='Lisboa'} (\text{TEMA_A})$

```
SELECT *  
FROM TEMA_A  
WHERE capital = 'Lisboa';
```

União de Tema (*Theme Union*)

$$\cup : \text{tema} \times \text{tema} \rightarrow \text{tema}$$

efectua a união dos conjuntos de objectos-geográficos;
a descrição (atributos alfa-numéricos) tem que ser compatíveis em união.

Seja:

$T1 \equiv$ instância do esquema do tema $th1$

$T2 \equiv$ instância do esquema do tema $th2$

$\{A_{1-T1}, \dots, A_{n-T1}\}$ e $\{A_{1-T2}, \dots, A_{n-T2}\} \equiv$ descrição, respectivamente, de $th1$ e $th2$

$geo1$ e $geo2 \equiv$ componente-espacial, respectivamente, de $th1$ e de $th2$

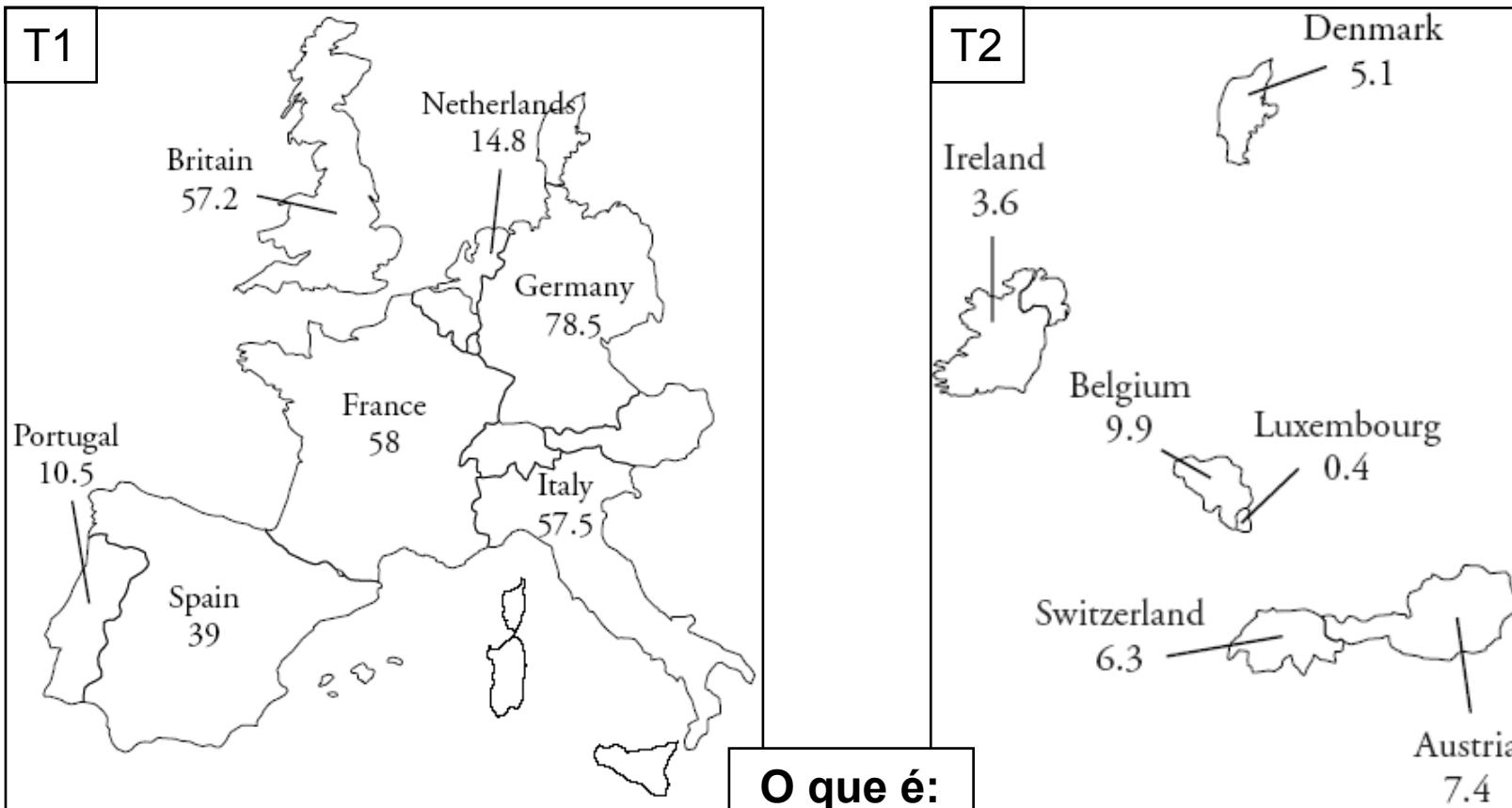
A **união** de $T1$ com $T2$ escreve-se (mesma notação da álgebra relacional):

$$T1 \cup T2$$

onde cada atributo A_{i-T1} é compatível com A_{i-T2}

União de Tema – exemplo

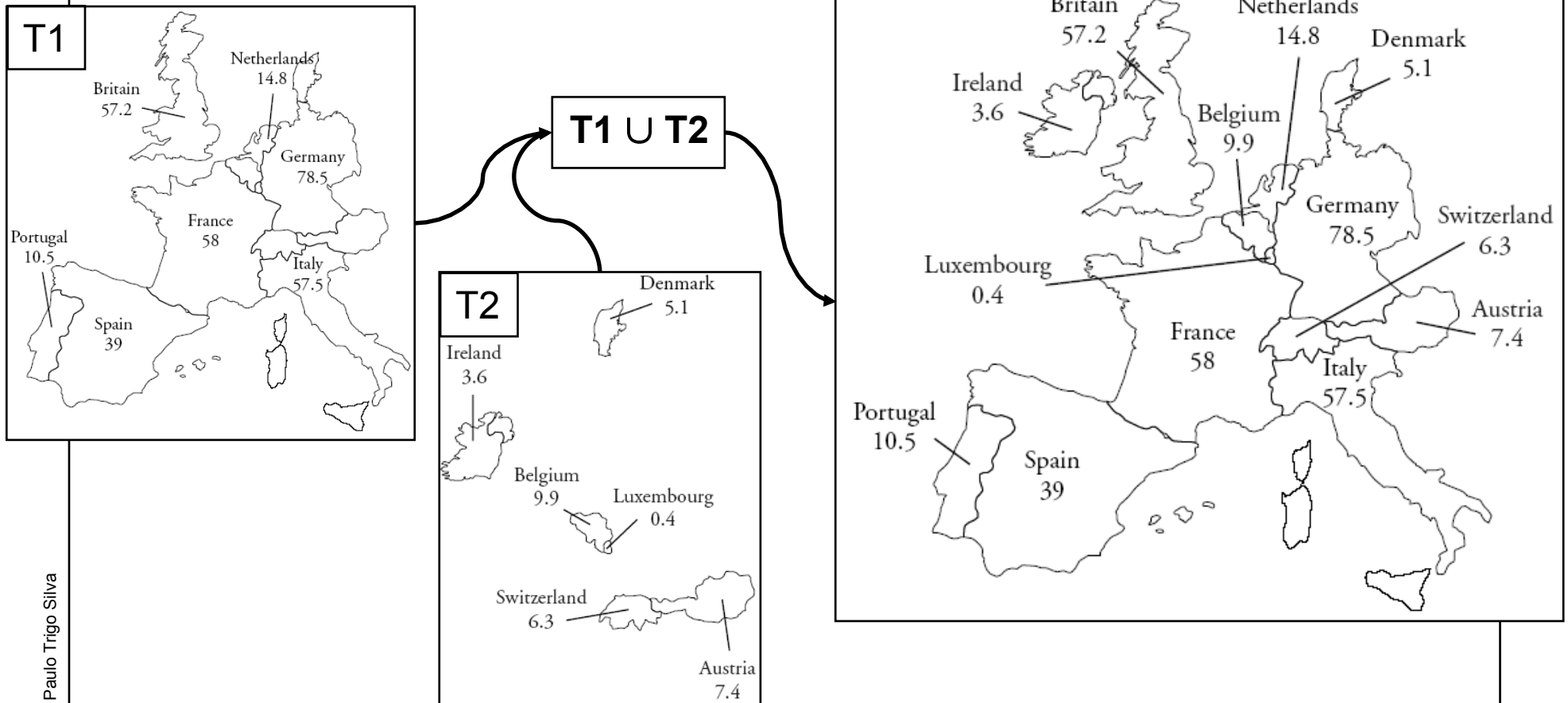
T1 \equiv *países-da-europa* com 10M, ou mais, habitantes; T2 \equiv com menos de 10M
em T1 e em T2: A₁ \equiv nome; A₂ \equiv população
geo \equiv polígono representando as fronteiras de cada país



O que é:
T1 \cup T2 ?

União de Tema – exemplo (cont.)

T1 = *países-da-europa* com 10M, ou mais, habitantes; T2 = com menos de 10M
em T1 e em T2: A₁ = nome; A₂ = população
geo = polígono representando as fronteiras de cada país



... uma união usando o exemplo dos “Tema A” e “Tema B”

```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

```
CREATE VIEW TEMA_B( nomeLinguagem, geo_regiao ) AS  
SELECT nomeLinguagem, geo_regiao  
FROM LINGUAGEM;
```

$\pi_{\text{nomePais, geo_unidAdmin}} (\text{TEMA_A}) \cup \text{TEMA_B}$

```
SELECT nomePais, geo_unidAdmin  
FROM TEMA_A  
UNION  
SELECT nomeLinguagem, geo_regiao  
FROM TEMA_B;
```

primeiro projeção para tornar
compatíveis em união.

Sobreposição de Tema (*Theme Overlay*)

$$G^{\bowtie} : \text{tema} \times \text{tema} \rightarrow \text{tema}$$

efectua a junção espacial dos conjuntos de objectos-geográficos;
a descrição é a união das descrições dos temas participantes.

Seja:

$T1 \equiv$ instância do esquema do tema $th1$

$T2 \equiv$ instância do esquema do tema $th2$

$\{A_{1-T1}, \dots, A_{n-T1}\}$ e $\{A_{1-T2}, \dots, A_{n-T2}\} \equiv$ descrição, respectivamente, de $th1$ e $th2$

$geo1$ e $geo2 \equiv$ componente-espacial, respectivamente, de $th1$ e de $th2$

A **sobreposição** de $T1$ e $T2$ escreve-se (notação idêntica à álgebra relacional):

$$T1 \mathrel{G^{\bowtie}} T2$$

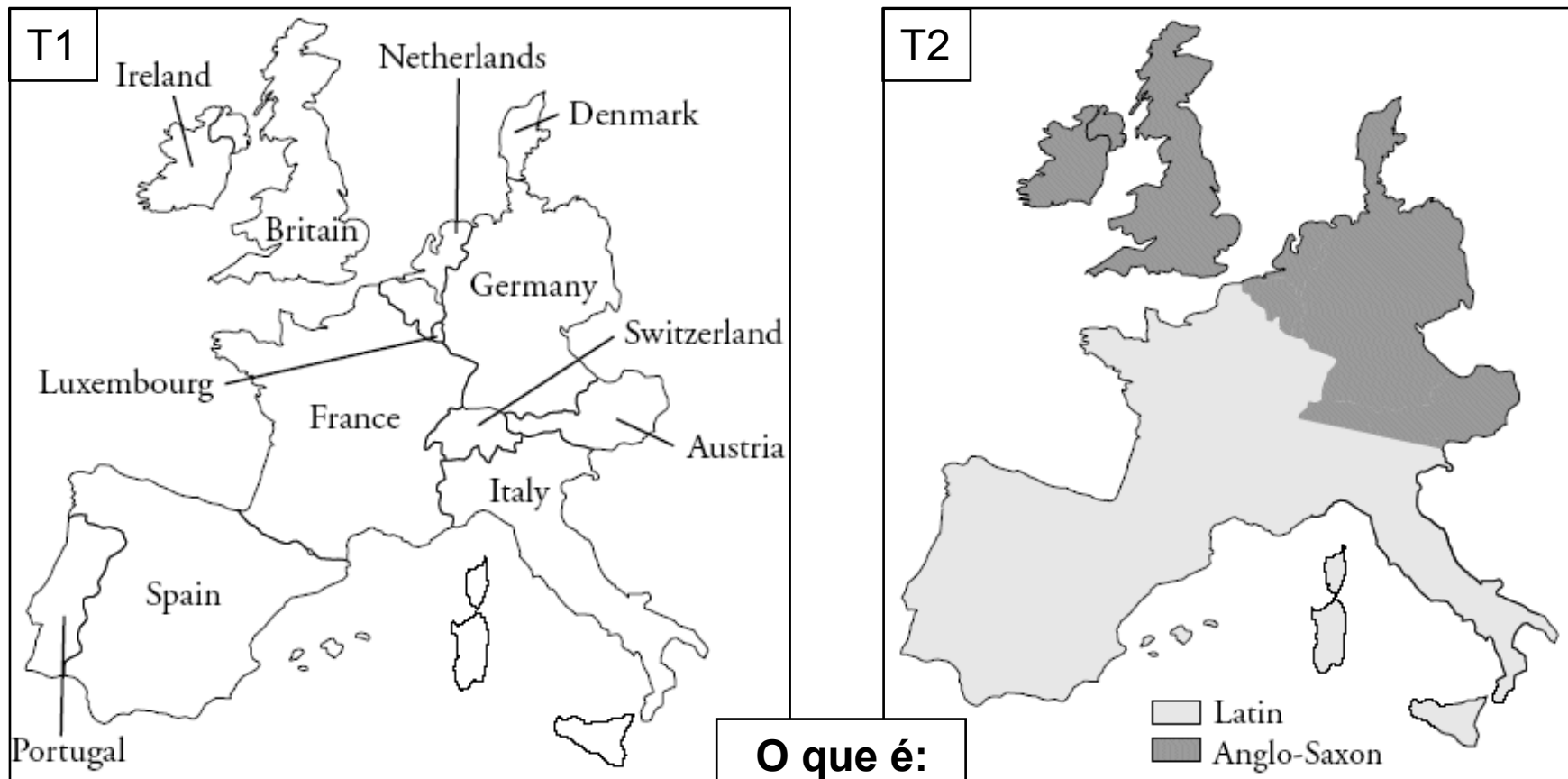
resultando a descrição composta pelos atributos $\{A_{1-T1}, \dots, A_{n-T1}, A_{1-T2}, \dots, A_{n-T2}\}$

Sobreposição de Tema – exemplo

T1 \equiv *países-da-europa*; T2 \equiv *linguagens nos países-da-europa*

em T1: A₁ \equiv nome; em T2: A₁ \equiv nome

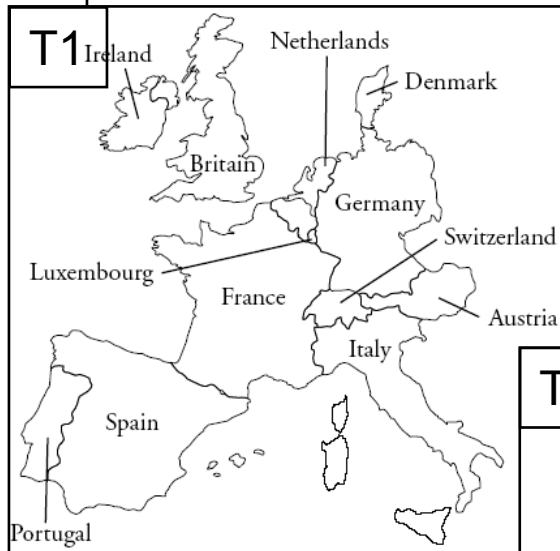
geo \equiv polígono representando as fronteiras de cada país



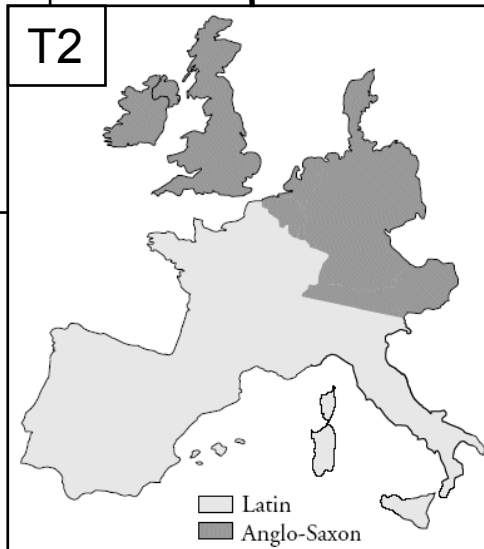
O que é:
T1 \bowtie T2 ?

Sobreposição de Tema – exemplo (cont.)

T1 = *países-da-europa*; T2 = *linguagens nos países-da-europa*
em T1: A₁ = nome; em T2: A₁ = nome
geo = polígono representando as fronteiras de cada país



T1 \bowtie **T2**



Um exemplo do novo tema é a parte norte da Suíça cuja descrição é:
“Switzerland” e “Anglo-Saxon”



... uma sobreposição usando os “Tema A” e “Tema B”

```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

```
CREATE VIEW TEMA_B( nomeLinguagem, geo_regiao ) AS  
SELECT nomeLinguagem, geo_regiao  
FROM LINGUAGEM;
```

TEMA_A \bowtie TEMA_B

```
SELECT nomePais, capital, populacao, nomeUnidAdmin,  
       nomeLinguagem,  
       geo_pais, geo_unidAdmin, geo_regiao  
FROM TEMA_A, TEMA_B  
WHERE ST_Intersects( geo_regiao, geo_unidAdmin );
```

→ relação topológica a satisfazer

Agregação de Tema (*Theme Merger*)

$${}_G\mathfrak{S}: \text{tema} \times p(A_1, \dots, A_n) \rightarrow \text{tema}$$

$p(A_1, \dots, A_n)$ é um predicado sobre a descrição do tema;
merger realiza a união geométrica da componente-espacial de vários
objectos-geográficos pertencentes ao mesmo tema e satisfazendo p

Seja:

$T \equiv$ instância do esquema do tema th

$\{A_1, \dots, A_n\} \equiv$ descrição de th

$geo \equiv$ componente-espacial de th

A **agregação** de T escreve-se (diferente da álgebra relacional):

$${}_G\mathfrak{S}_{p(A_1, \dots, A_n)}(T)$$

Agregação de Tema – exemplo

$T \equiv \text{países-da-europa}$

$A_1 \equiv \text{nome}$

$\text{geo} \equiv \text{polígono representando as fronteiras de cada país}$



Notar que é diferente da união de temas que aceita dois temas como argumentos e os combina num único tema

?

O que é:

$G \sim A_1 = \text{'Former West germany' OR } A_1 = \text{'Former East Germany' } (T)$

— Agregação de Tema – exemplo (cont.) —

$T \equiv \text{países-da-europa}$

$A_1 \equiv \text{nome}$

$\text{geo} \equiv \text{polígono representando as fronteiras de cada país}$

$G \mathcal{S} A_1 = \text{'Former West germany'} \text{ OR } A_1 = \text{'Former East Germany'} (T)$



... uma agregação usando o “Tema A”

```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

geo_pais \mathcal{I} **nomePais='P1' OR nomePais='P2') (TEMA_A)**

```
SELECT ST_Union( geo_pais )  
FROM TEMA_A  
WHERE nomePais='P1' or nomePais='P2';
```

Seleccção Geométrica (*Geometric Selection*): *window query*

$$\square \sigma : \text{tema} \times \text{figura-geométrica} \rightarrow \text{tema}$$

intersecta o tema com uma figura-geométrica, em geral um rectângulo;
devolve apenas os objectos-geométricos, do tema, contidos na figura-geométrica.

Seja:

$T \equiv$ instância do esquema do tema th

$\{ A_1, \dots, A_n \} \equiv$ descrição de th

$geo \equiv$ componente-espacial de th

A **selecção geométrica – *window query*** de T escreve-se:

$$\square \sigma \langle (x1,y1), (x2, y1), (x2, y2), (x1, y2), (x1, y1) \rangle (T)$$

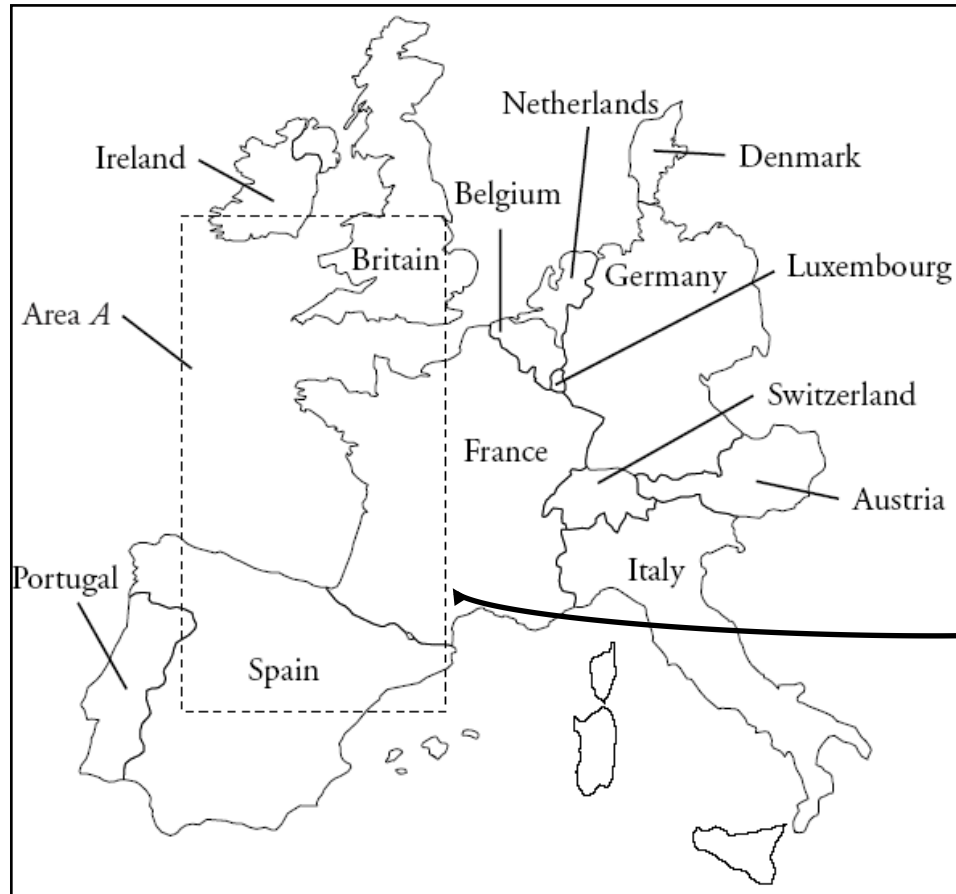
numa janela de interrogação (*window query*) do tipo rectângulo a
figura-geométrica tem o formato $\langle (x1,y1), (x2, y1), (x2, y2), (x1, y2), (x1, y1) \rangle$

Seleccção Geométrica: *window query* – exemplo

$T \equiv \text{países-da-europa}$

$A_1 \equiv \text{nome}$

$\text{geo} \equiv \text{polígono representando as fronteiras de cada país}$



?

O que é:

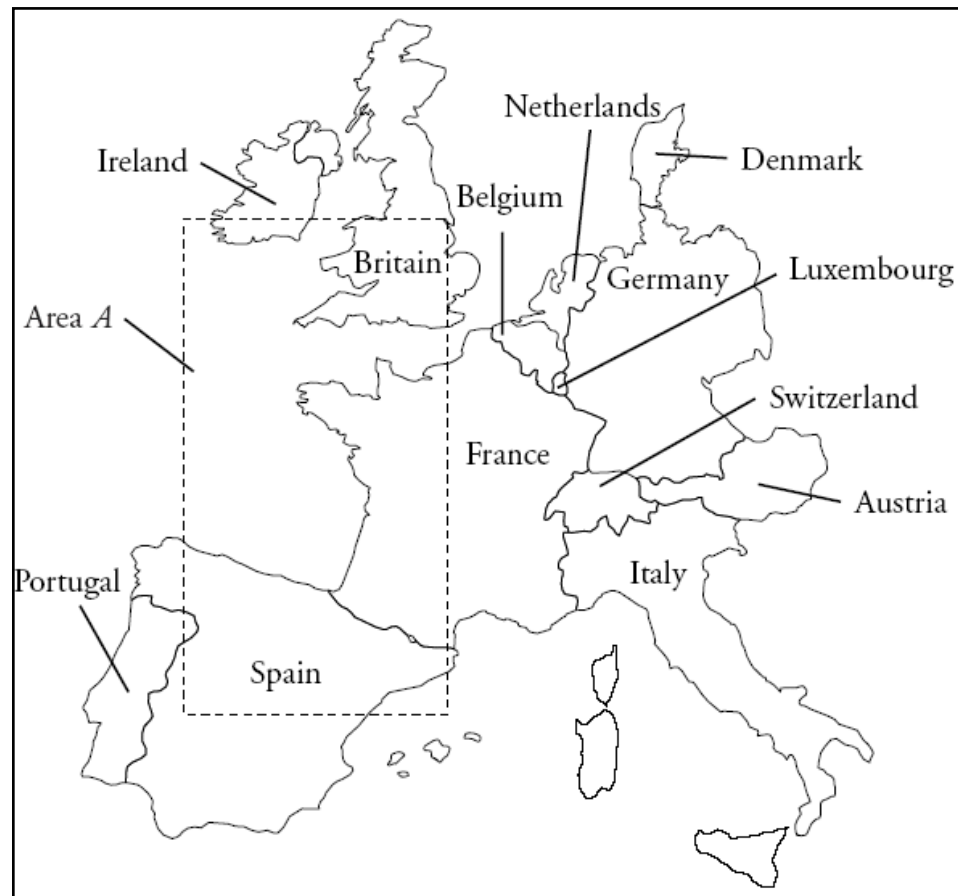
$\sigma_{\text{figura-geométrica}}(T)$

Seleccção Geométrica: *window query* – exemplo (cont.)

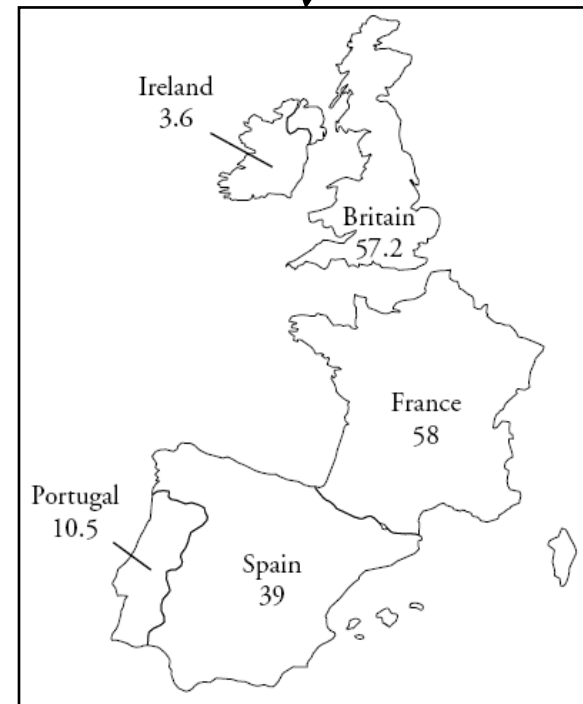
$T \equiv \text{países-da-europa}$

$A_1 \equiv \text{nome}$

$\text{geo} \equiv \text{polígono representando as fronteiras de cada país}$



$\sigma_{\text{figura-geométrica}}(T)$



... uma *window query* usando o “Tema A”

```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

□ $\sigma_{\langle 0\ 0, 0\ 20, 10\ 20, 10\ 0, 0\ 0 \rangle} (\text{TEMA_A})$

```
SELECT geo_pais, geo_unidAdmin  
FROM TEMA_A  
WHERE ST_Intersects( geo_pais,  
                     ST_GeomFromText( 'POLYGON(  
                                     (0 0, 0 20, 10 20, 10 0, 0 0))' ) );
```

Seleccção Geométrica (*Geometric Selection*): *point query*

$\sigma : \text{tema} \times \text{ponto-geométrico} \rightarrow \text{tema}$

idêntico à *window query* mas, neste caso, devolve todos os objectos cuja geometria contenha o ponto-geométrico indicado.

Seja:

$T \equiv$ instância do esquema do tema th

$\{ A_1, \dots, A_n \} \equiv$ descrição de th

$geo \equiv$ componente-espacial de th

A **selecção geométrica – *point query*** de T escreve-se:

$\sigma_{(x1,y1)} (T)$

um ponto de interrogação (*point query*) tem o formato $(x1,y1)$

Seleccção Geométrica (*Geometric Selection*): *clipping*

■ $\sigma : \text{tema} \times \text{figura-geométrica} \rightarrow \text{tema}$

extrai a região do tema intersectada pela figura-geométrica

o *clipping* é diferente *window query*

no *clipping* a geometria do objecto resultante corresponde exactamente à intersecção da geometria dos objectos-geográficos com a figura-geométrica

Seja:

$T \equiv$ instância do esquema do tema th

$\{A_1, \dots, A_n\} \equiv$ descrição de th

$geo \equiv$ componente-espacial de th

A **selecção geométrica – *clipping*** de T escreve-se:

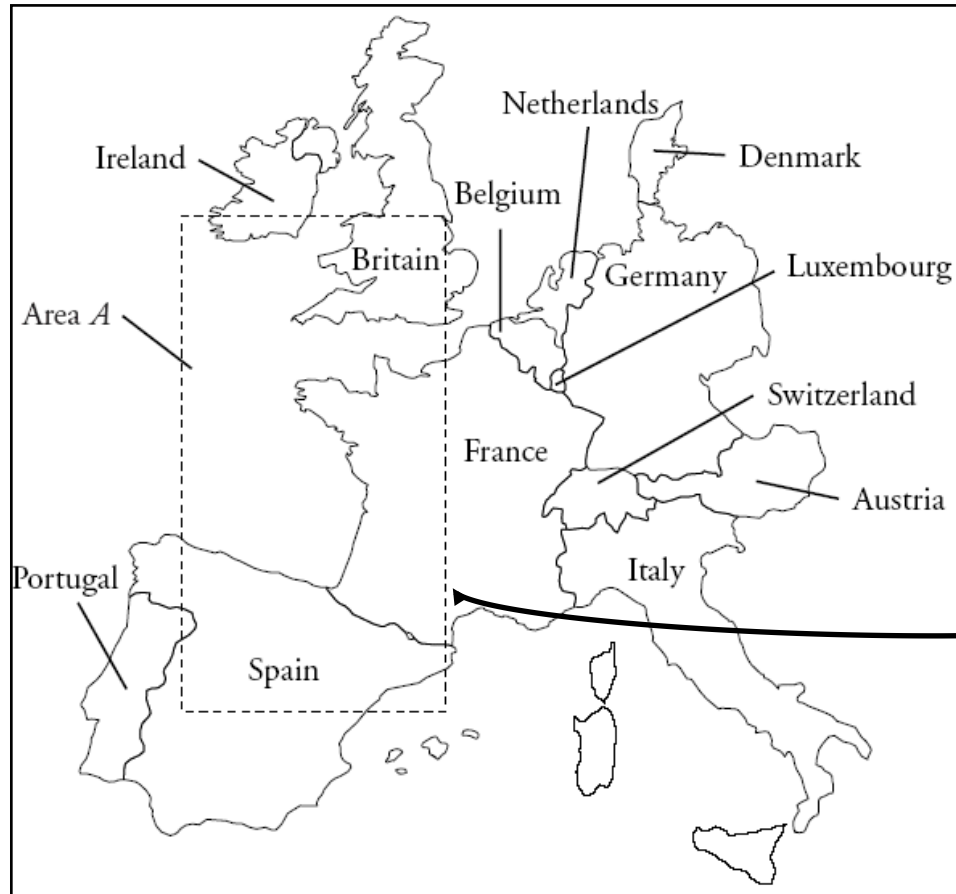
■ $\sigma_{\langle (x1,y1), (x2,y1), (x2,y2), (x1,y2), (x1,y1) \rangle} (T)$

Seleccção Geométrica: *clipping* – exemplo

$T \equiv \text{países-da-europa}$

$A_1 \equiv \text{nome}$

$\text{geo} \equiv \text{polígono representando as fronteiras de cada país}$



?

O que é:

■ $\sigma_{\text{figura-geométrica}}(T)$

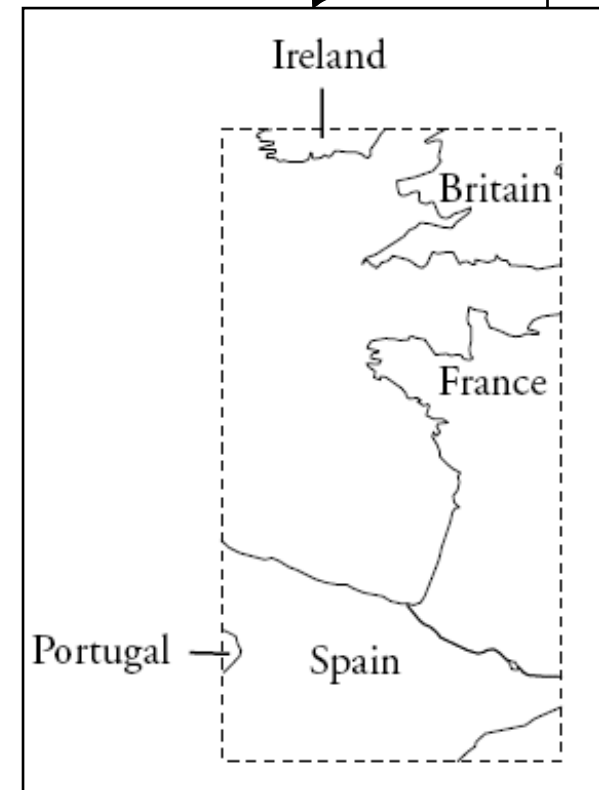
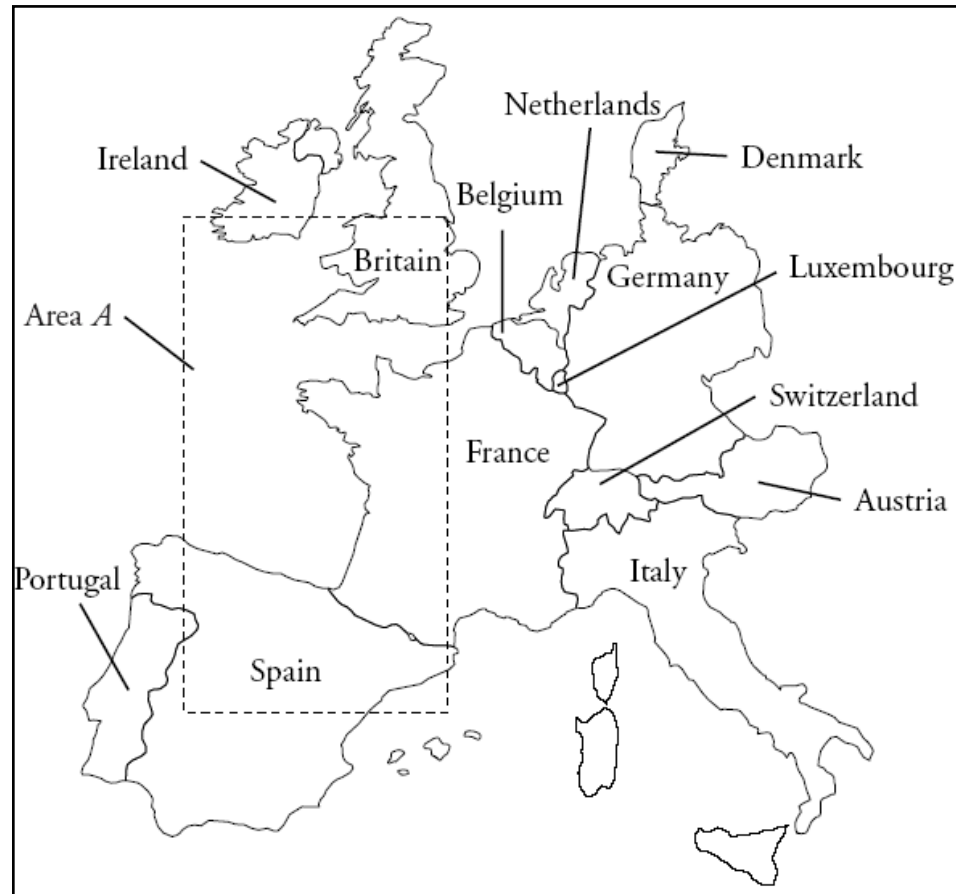
Seleccção Geométrica: *clipping* – exemplo (cont.)

$T \equiv \text{países-da-europa}$

$A_1 \equiv \text{nome}$

$\text{geo} \equiv \text{polígono representando as fronteiras de cada país}$

■ $\sigma_{\text{figura-geométrica}}(T)$



... um *clipping* usando o “Tema A”

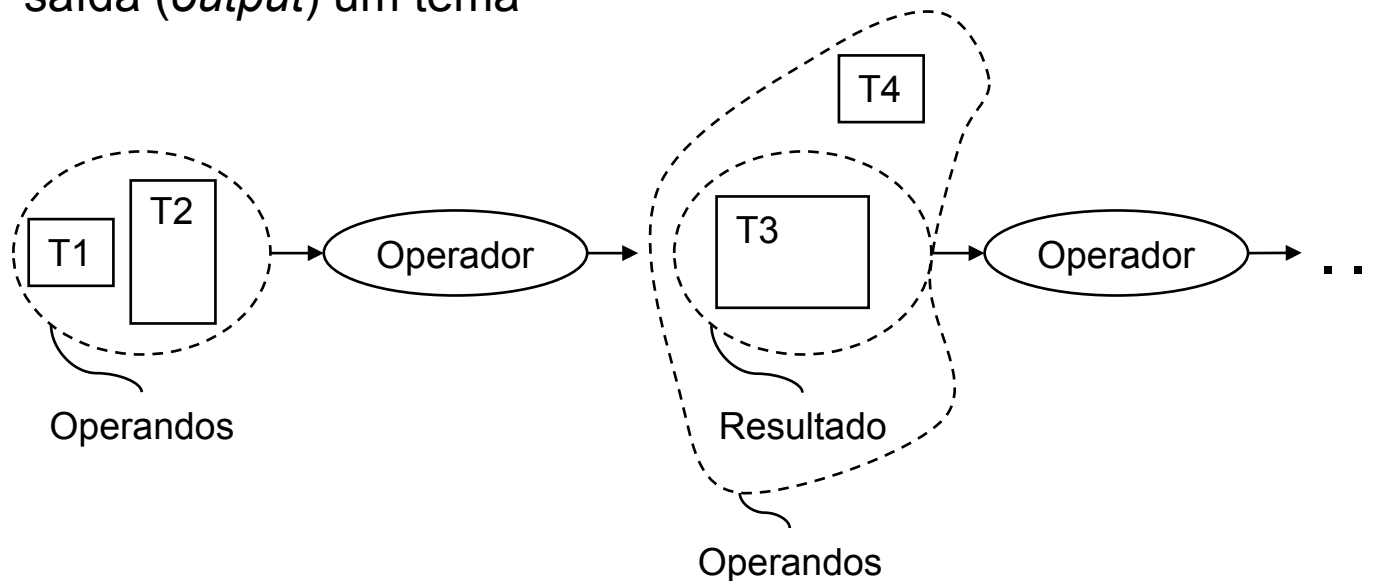
```
CREATE VIEW TEMA_A( nomePais, capital, populacao, nomeUnidAdmin,  
                    geo_pais, geo_unidAdmin ) AS  
SELECT P.nomeP, capital, populacao, nomeR,  
       P.geo_regiao, UA.geo_regiao  
FROM PAIS P, UNID_ADMIN UA  
WHERE ST_Contains( P.g_regiao, UA.g_regiao );
```

■ $\sigma_{<0\ 0, 0\ 20, 10\ 20, 10\ 0, 0\ 0>}$ (TEMA_A)

```
SELECT ST_Intersection( g_pais, ST_GeomFromText  
    ( 'POLYGON((0 0, 0 20, 10 20, 10 0, 0 0))' ) ),  
    ST_Intersection( g_unidAdmin, ST_GeomFromText  
    ( 'POLYGON((0 0, 0 20, 10 20, 10 0, 0 0))' ) )  
FROM TEMA_A;
```

Operadores formam uma “Álgebra de Tema”

- Os operadores anteriores têm
 - entrada (*input*) um ou mais temas
 - saída (*output*) um tema



- ... podem considerar-se que formam uma “Álgebra de Tema”
 - permitem escrever expressões sobre temas para descrever outro tema

Outras operações envolvendo Tema

- Operações que utilizem métricas
 - e.g., *Qual a distância entre Lisboa e Madrid?*
- Operações topológicas
 - tratam as relações (topológicas) entre dados
 - e.g., *Quais os países adjacentes a Espanha?*
 - e.g., *Que cidades são acessíveis directamente de Lisboa por comboio?*

Tema descrito usando o Modelo Relacional (MRel)

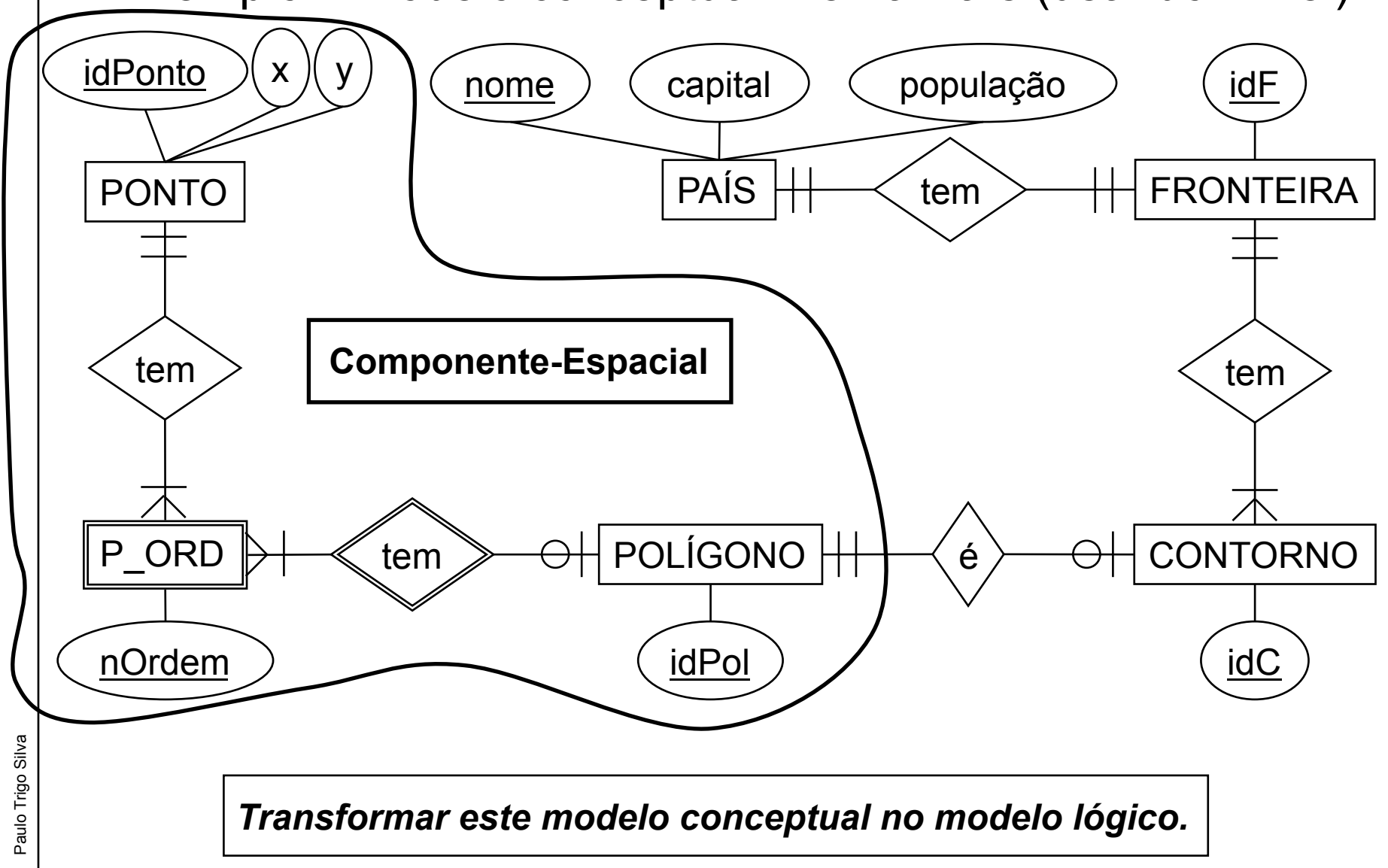
- Um tema pode ser descrito por esquemas de relação (tabelas)
 - um objecto-geográfico serão tuplos (linhas) nesses esquemas
 - algumas das colunas formam a descrição do tema
 - outras colunas representam as componentes-geométricas
- Exemplo de utilização do modelo relacional – tema País

```
PAÍS( nomeP, capital, população, fronteira )
```

O atributo (geométrico) `fronteira` corresponde à fronteira do país.
Um país pode incluir diversas partes logo uma fronteira é formada por contornos.
Cada contorno é um polígono com identificador único.
Cada polígono tem uma lista de pontos (cada ponto é um vértice).
Os vértices de cada polígono representam-se seguindo uma relação de ordem.

Construir um modelo conceptual para representar este conceito de país.

Exemplo – modelo conceptual: Tema País (usando MRel)



Exemplo – modelo lógico: Tema País (usando MRel)

PAÍS(nomeP, capital, população, *idF*)
cc = { nomeP } e { *idF* }

FRONTEIRA(*idF*)

CONTORNO(*idC*, *idPol*, *idF*)
cc = { *idC* } e { *idPol* }

POLÍGONO(*idPol*)

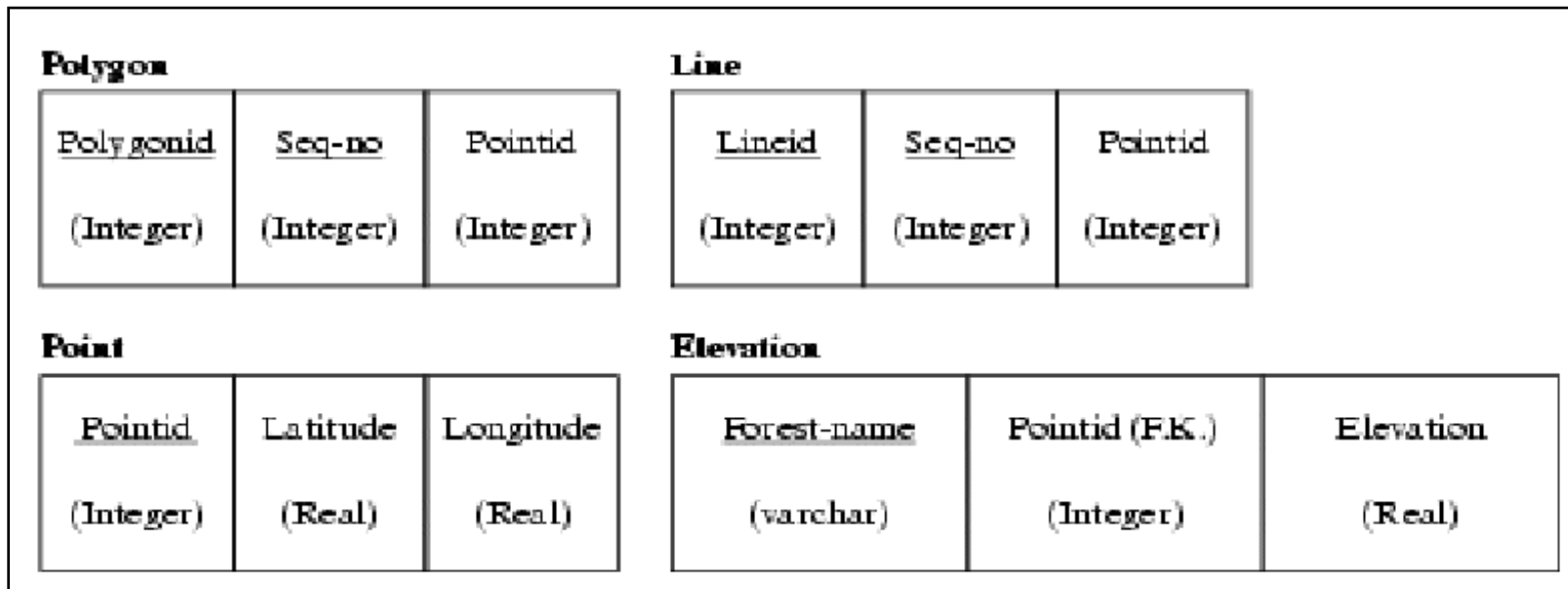
PONTO(*idPonto*, x, y)

P_ORD(*idPol*, nOrdem, *idPonto*)

chaves estrangeiras
com grafia em *itálico*

Componente-Espacial

... outra abordagem com MRel – componente-espacial



Aspectos do Modelo Relacional:

- apenas permite valores atômicos
 - não permite colunas com valores complexos (e.g., polígonos)
- valores complexos têm que ser decompostos em domínios mais simples
 - e.g., um polígono pode ser decomposto em vértices e arestas

Exemplo – obter informação espacial com MRel

Pretende-se saber:

Qual a fronteira de Portugal?

Para resolver esta interrogação é necessário:

- obter cada um dos contornos que caracterizam a fronteira (continente e ilhas)
 - obter o polígono de cada contorno
 - obter as coordenadas de todos os pontos de cada polígono

Escrever a directiva SQL que responde à questão colocada.

Exemplo – obter informação espacial com MRel e SQL

```
SELECT    P.idF, x, y

FROM      PAÍS P, FRONTEIRA F, CONTORNO C,
            POLÍGONO POL, PONTO, P_ORD

WHERE      P.nomeP = 'Portugal' AND
            P.idF = F.idF AND
            F.idF = C.idF AND
            C.idPol = POL.idPol AND
            POL.idPol = P_ORD.idPol AND
            P_ORD.idPonto = PONTO.idPonto

ORDER BY  P.idF, P_ORD.nOrdem ASC
```

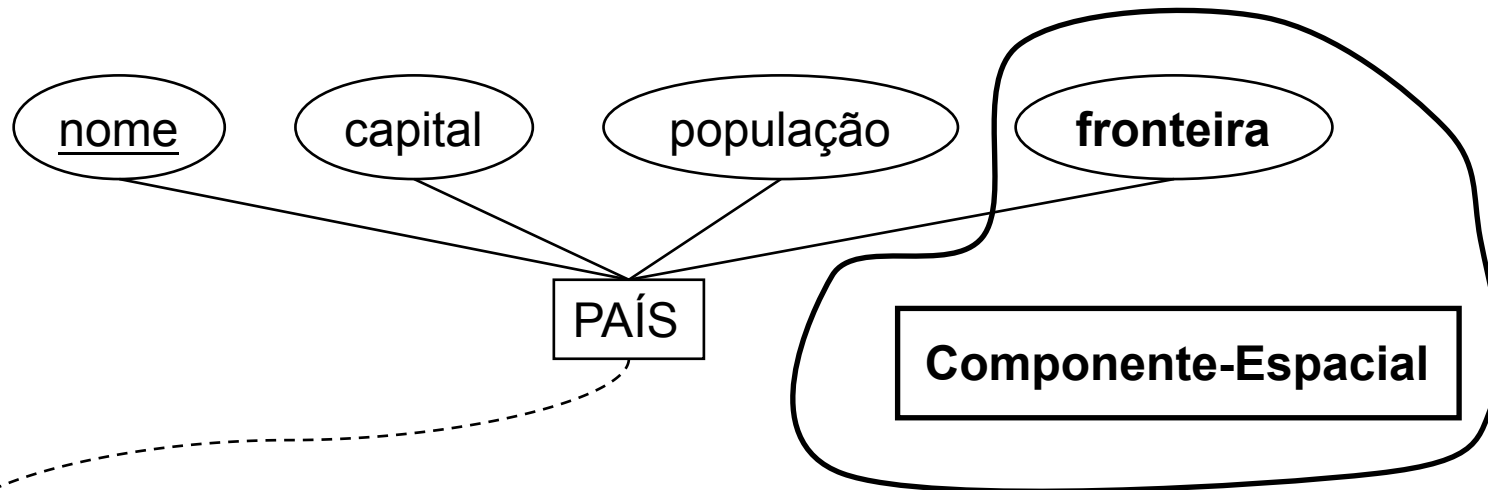
Vantagem desta abordagem: obtém informação espacial recorrendo a SQL

E que desvantagens se podem identificar?

Desvantagens – informação espacial com MRel e SQL

- Quebra do princípio da independência dos dados
 - formular a interrogação exige conhecer estrutura dos objectos espaciais
- ... alterar estrutura implica
 - profunda reorganização da base de dados
 - refazer todas as interrogações espaciais
- Impossível, ou muito difícil, exprimir computações geométricas
 - testes de adjacência ou de inclusão, cálculo de área ou de distância
- Baixo desempenho da abordagem
 - elevado número de junções de tabelas

Exemplo – País com Modelo Objecto-Relacional (MORel)



```
PAÍS( nomeP, capital, população, fronteira:polygon )  
cc = { nomeP }
```

Pretende-se saber:
Qual a fronteira de Portugal?

```
SELECT    fronteira  
  
FROM      PAÍS P  
  
WHERE     P.nomeP = 'Portugal'
```

Modelo Objecto-Relacional (MOREl) – estender o SQL

- O essencial do Modelo Objecto-Relacional (MOREl)
 - é o de suportar novos tipos de dados (pré-definidos ou do utilizador)
 - ... com consequente necessidade de estender o suporte SQL
- Em geral o MOREl deve oferecer suporte para especificação de
 - novos tipos de dados
 - novas funções
 - novos operadores
- ... SQL é estendido para suportar aquelas especificação
 - mas alguns SGBDs permitem usar linguagens de programação para especificar novos operadores ou novas funções
 - ... e.g., Python ou C (no PostgreSQL)

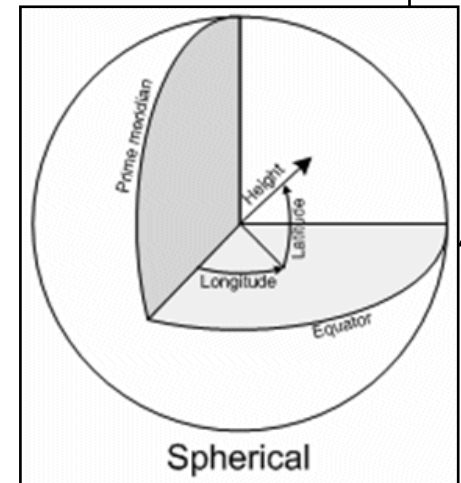
Criar Novos Tipos de Dados (do utilizador)

```
CREATE TYPE t_PONTO AS ( latitude numeric(4,2), longitude numeric(4,2) );  
CREATE TYPE t_CONTINENTE AS ENUM (  
    'Europa', 'América', 'África', 'Ásia', 'Oceania', 'Antártica');
```

```
CREATE TABLE CIDADE  
( nome varchar(30),  
  continente t_CONTINENTE,  
  geo t_PONTO );
```

```
INSERT INTO CIDADE VALUES  
( 'Lisboa', 'Europa',  
  ROW( 38.42, -9.11 ) );
```

```
INSERT INTO CIDADE VALUES  
( 'Madrid', 'Europa',  
  ROW( 40.23, -3.40 ) );
```



Convenção:
Norte ≡ +; Sul ≡ -
Este ≡ +; Oeste ≡ -

no PostgreSQL

Interrogar usando os novos tipos de dados

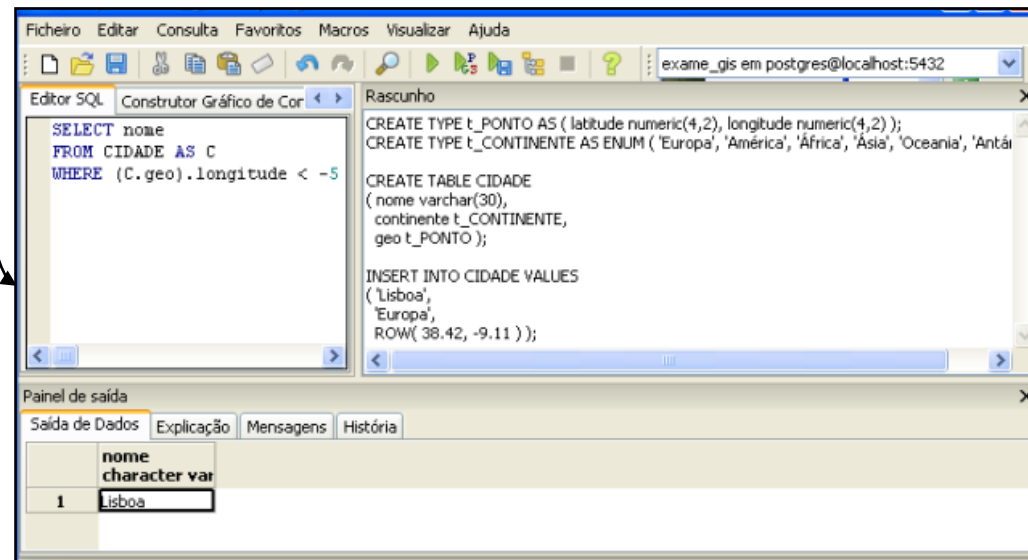
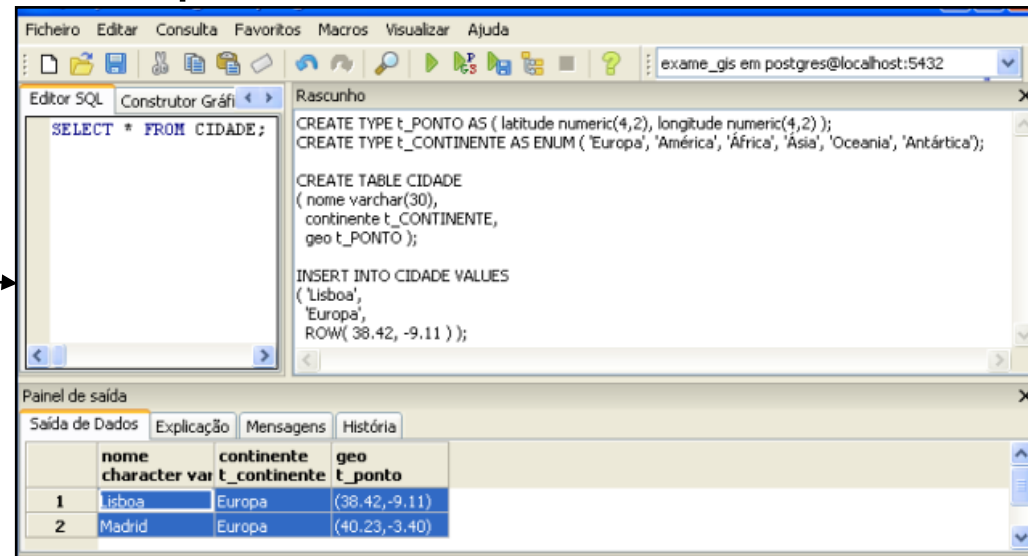
```
SELECT * FROM CIDADE
```

```
SELECT nome  
FROM CIDADE AS C  
WHERE (C.geo).longitude < -5
```

aceder a campo em estrutura;
ou apenas (geo).latitude
se a tabela não for renomeada

... e no final!

```
DROP TABLE CIDADE;  
DROP TYPE t_PONTO;  
DROP TYPE t_CONTINENTE;
```



Criar Novos Tipos de Dados (aspectos mais avançados)

```
CREATE TYPE name AS
    ( attribute_name data_type [, ... ] )

CREATE TYPE name AS ENUM
    ( [ 'label' [, ... ] ] )

CREATE TYPE name (
    INPUT = input_function,
    OUTPUT = output_function
    [ , RECEIVE = receive_function ]
    [ , SEND = send_function ]
    [ , TYPMOD_IN = type_modifier_input_function ]
    [ , TYPMOD_OUT = type_modifier_output_function ]
    [ , ANALYZE = analyze_function ]
    [ , INTERNALLENGTH = { internallength | VARIABLE } ]
    [ , PASSEDBYVALUE ]
    [ , ALIGNMENT = alignment ]
    [ , STORAGE = storage ]
    [ , LIKE = like_type ]
    [ , CATEGORY = category ]
    [ , PREFERRED = preferred ]
    [ , DEFAULT = default ]
    [ , ELEMENT = element ]
    [ , DELIMITER = delimiter ]
)

CREATE TYPE name
```

aceita uma *string* e devolve a representação, em memória, do tipo

aceita uma representação, em memória, do tipo e devolve uma *string*

detalhe adicional em:

postgresql-9.0-A4.pdf (e.g., secção 35.11)

Criar Novas Funções (do utilizador – em Python)

```
CREATE FUNCTION nomeDaFuncao( lista-de-argumentos )  
RETURNS tipo-do-retorno  
AS $$  
    # corpo da função PL/Python  
$$ LANGUAGE plpythonu;
```

Exemplo em Python

```
CREATE LANGUAGE plpythonu;  
CREATE FUNCTION py_max( a integer, b integer )  
RETURNS integer  
AS $$  
    if a > b: return a  
    return b  
$$ LANGUAGE plpythonu;
```

Muito importante:

instalar versão Python compatível com a do PostgreSQL

e.g., Python 2.6.x para PostgreSQL 8.4
(Python 2.7.x não funciona com o 8.4)

Criar Novas Funções (passagem parâmetros em Python)

```
CREATE FUNCTION py_max( a integer, b integer )  
RETURNS integer  
AS $$  
    if a > b: return a  
    return b  
$$ LANGUAGE plpythonu;
```

```
def __plpython_procedure_py_max_23456():  
    if a > b: return a  
    return b
```

código gerado assumindo que 23456 é o
OID atribuído à função pelo PostgreSQL

os argumentos são definidos como variáveis globais

as regras de *scope* do Python não permitem afectar a variável com uma expressão que envolva a própria variável; para isso é preciso re-declarar a variável como global.

```
CREATE FUNCTION myStrip( x text )  
RETURNS text  
AS $$  
    x = x.strip() // ERRADO  
    return x  
$$ LANGUAGE plpythonu;
```

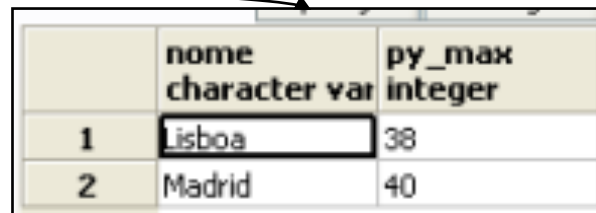
```
CREATE FUNCTION myStrip( x text )  
RETURNS text  
AS $$  
    global x  
    x = x.strip() // CORRECTO  
    return x  
$$ LANGUAGE plpythonu;
```

MELHOR: tratar parâmetros como *read-only*

Exemplo completo – com interrogação e coerção (*cast*)

```
DROP LANGUAGE plpythonu CASCADE;  
CREATE LANGUAGE plpythonu;  
CREATE FUNCTION py_max( a integer, b integer )  
RETURNS integer  
AS $$  
    if a > b: return a  
    return b  
$$ LANGUAGE plpythonu;
```

```
SELECT nome, py_max( CAST( (C.geo).latitude AS integer ), CAST( (C.geo).longitude AS integer ) )  
FROM CIDADE AS C  
  
DROP FUNCTION py_max( integer, integer )
```



	nome character var	py_max integer
1	Lisboa	38
2	Madrid	40

Exemplo completo – função com parâmetros *numeric*

```
DROP LANGUAGE plpythonu CASCADE;  
CREATE LANGUAGE plpythonu;  
CREATE FUNCTION py_max( a numeric, b numeric )  
RETURNS numeric  
AS $$  
    if a > b: return a  
    return b  
$$ LANGUAGE plpythonu;
```

```
SELECT nome, py_max( (C.geo).latitude, (C.geo).longitude )  
FROM CIDADE AS C  
  
DROP FUNCTION py_max( numeric, numeric )
```

	nome character var	py_max numeric
1	Lisboa	38.42
2	Madrid	40.23

detalhe adicional em:

postgresql-9.0-A4.pdf (e.g., secção 42.2)

Criar Novas Funções (com argumentos que são tabelas)

```
CREATE TABLE EMPREGADO (  
    nome text PRIMARY KEY,  
    salario numeric,  
    idade integer );
```

```
INSERT INTO EMPREGADO VALUES ( 'Pedro', 1700, 40 );  
INSERT INTO EMPREGADO VALUES ( 'Maria', 1600, 29 );  
INSERT INTO EMPREGADO VALUES ( 'Joana', 3000, 35 );
```

Considera-se “salário alto” o que é superior a 2000.
Também é “salário alto” se superior a 1500 quando idade inferior a 30 anos.

Construir função para detectar salários altos!

```
CREATE FUNCTION salarioAlto ( e EMPREGADO ) ...
```

... funções com argumentos que são tabelas

```
CREATE FUNCTION salarioAlto ( e EMPREGADO )  
RETURNS boolean  
AS $$  
    if e[ "salario" ] > 2000: return True  
    if ( e[ "idade" ] < 30) and ( e[ "salario" ] > 1500 ): return True  
    return False  
$$ LANGUAGE plpythonu;
```

```
SELECT *  
FROM EMPREGADO AS E  
WHERE salarioAlto( E );
```

	nome text	salario numeric	idade integer
1	Maria	1600	29
2	Joana	3000	35

Criar Novos Operadores

```
CREATE TYPE t_PONTO AS ( x numeric(4,2), y numeric(4,2) );

CREATE FUNCTION somaPontos( p1 t_PONTO, p2 t_PONTO )
RETURNS t_PONTO
AS $$
    new_x = p1[ "x" ] + p2[ "x" ]
    new_y = p1[ "y" ] + p2[ "y" ]
    return { "x": new_x, "y": new_y }
$$ LANGUAGE plpythonu;
```

Definição do novo operador +
sobre t_PONTO

```
CREATE OPERATOR + (
    leftarg = t_PONTO,
    rightarg = t_PONTO,
    procedure = somaPontos,
    commutator = +
);
```

O retorno é uma tabela;
pode usar-se um dicionário
(tabela de *hash*) Python


Usar o Novo Operador Criado

```
CREATE TABLE LOCAL(  
    nome varchar(30) PRIMARY KEY,  
    geo t_PONTO );  
  
INSERT INTO LOCAL VALUES ( 'localA', ROW( 0, 0 ) );  
INSERT INTO LOCAL VALUES ( 'localB', ROW( 5, 5 ) );
```

```
SELECT (L1.geo) + (L2.geo) AS c  
FROM LOCAL L1, LOCAL L2;
```

... e no final!

```
DROP OPERATOR + ( t_PONTO, t_PONTO );  
DROP FUNCTION somaPontos ( t_PONTO, t_PONTO );  
DROP TABLE LOCAL;  
DROP TYPE t_PONTO;
```



	c t_ponto
1	(0.0,0.0)
2	(5.0,5.0)
3	(5.0,5.0)
4	(10.0,10.0)