I sistemi per la gestione di basi di dati geografiche

Regole di corrispondenza tra GeoUML e Simple Features Specification for SQL

Novembre 2005

Alberto Belussi

Progettazione logico/fisica di una base di dati geografica

La progettazione logico/fisica di una base di dati geografica definisce le strutture di memorizzazione della componente spaziale e alfanumerica dell'informazione in essa contenuta.

La progettazione logico/fisica segue la progettazione concettuale.

Essa richiede quindi che lo schema concettuale dei dati sia già stato redatto. Esso è infatti il punto di partenza della progettazione logico/fisica.

L'altro essenziale prerequisito è costituito dalla scelta di un sistema per gestione di basi di dati geografiche (GEO-DBMS). In particolare, deve essere noto il modello dei dati del sistema.

Criteri generali della progettazione logico/fisica

- Creare una struttura normalizzata per ridurre la ridondanza dei dati e quindi le anomalie che si possono verificare in fase di aggiornamento dei dati.
- Gestire i casi di geometria condivisa in modo esplicito nella struttura dei dati (strati topologici) quando possibile.
- Rappresentare il più possibile nella struttura fisica dei dati i vincoli di integrità spaziali significativi espressi nello schema concettuale.
- Creare una struttura che sia fruibile per l'uso applicativo; questo criterio risulta ovviamente in contrasto con l'esigenza di una struttura normalizzata introdotta dal primo criterio.

Mapping da GeoUML a Simple Feature Specification for SQL

Per quanto riguarda la trasformazione delle classi normali senza attributi geometrici e delle associazioni tra classi valgono le regole già presentate per le basi di dati tradizionali.

Poiché il modello logico/fisico è geo-relazionale, è sempre possibile rappresentare relazioni (tabelle) in tale modello e quindi sarà possibile rappresentare in esso anche le relazioni (tabelle) che costituiscono la traduzione di classi normali e associazioni di uno schema dei dati scritto in GeoUML.

Mapping da GeoUML a Simple Feature Specification for SQL

Per quanto riguarda invece i costrutti del modello GeoUML che contengono riferimenti alla geometria, vengono di seguito discusse alcune proposte di traduzione nel modello logico/fisico.

D: *Come si rappresenta la geometria nel modello dei dati di un sistema geo-relazionale?*

R: *Attraverso l'introduzione nello schema logico di* ***attributi geometrici*** *nelle relazioni e di* ***strati topologici****.*

D: *Quanti* ***strati topologici*** *devono essere creati in una base di dati territoriale?*
D: *Quali* ***attributi geometrici*** *vanno messi sullo stesso strato?*

Non esiste un unica risposta a queste domande!

Osservazioni sull'uso degli strati in uno schema logico/fisco

I dati geometrici contenuti nello stesso strato sono memorizzati in un unica struttura topologica nel GEO-DBMS, ciò implica:

- la gestione esplicita della geometria condivisa fra i valori geometrici contenuti nello strato;
- la possibilità di implementare i vincoli geometrici nella struttura dati dello strato;
- una maggiore facilità di aggiornamento dei dati contenuti nello strato, in particolare dove esistono vincoli di integrità spaziale da rispettare;
- la necessità di considerare l'intero strato per ogni accesso ai dati in esso contenuti.
- la possibilità di interrogare con maggiore efficienza i dati contenuti nello strato, in quanto le operazioni più potenti dei sistemi agiscono di solito intra-strato.

Suddivisione in strati degli attributi geometrici di uno schema GeoUML

Tenendo presente le osservazioni riportate in precedenza si possono considerare i tre seguenti criteri:

- generare uno strato topologico nello schema logico (*strato logico*) **per ogni strato topologico dichiarato a nello schema concettuale** (*strato concettuale*).
- porre sullo stesso *strato logico* tutti gli attributi geometrici legati direttamente o indirettamente a livello concettuale (attraverso vincoli strutturali) allo *strato concettuale* corrispondente.
- generare uno strato topologico nello schema logico (*strato logico aggiuntivo*) **per rappresentare gli attributi geometrici di più classi legate da vincoli strutturali su attributi geometrici (inclusi tratti e sottoaree)**.
- porre sullo stesso *strato logico aggiuntivo* tutti gli attributi geometrici legati direttamente o indirettamente a livello concettuale attraverso vincoli strutturali che hanno dato vita allo strato logico aggiuntivo
- porre preferibilmente sullo stesso strato i valori degli attributi geometrici che vengono **utilizzati insieme nelle elaborazioni realizzate con maggior frequenza** dalle applicazioni.

Le regole di corrispondenza tra GeoUML e il modello SFS

Classe C_1 contenente gli attributi alfanumerici:

$A_1, ..., A_n$

Mapping degli attributi alfanumerici normali

Per la classe C_1 viene creata una tabella C_1 che contiene tanti attributi quanti sono gli attributi alfanumerici della classe C_1. La tabella C_1 ha un attributo aggiuntivo detto *FID* (feature ID) di tipo integer e con nome idC_1 . Tale attributo diventa chiave primaria della tabella C_1.

$$\mathbf{C_1(\underline{idC_1}, A_1, ..., A_n)}$$

I tipi degli attributi vengono scelti in base alla seguente corrispondenza:

String: VARCHAR(x)

Integer: INTEGER

Real: REAL o DOUBLE

Le regole di corrispondenza tra GeoUML e il modello SFS

Classe C_1 contenente gli attributi alfanumerici:

A_1, ..., A_n dove $A_{i,1}$, …, $A_{i,k}$ sono chiave primaria

Mapping delle chiavi primarie

Alla tabella che rappresenta la C_1 viene aggiunto il seguente vincolo:

C_1($\underline{idC_1}$, A_1, ..., A_n)

vincolo C_1_PK ($A_{i,1}$, …, $A_{i,k}$)

se la chiave include ruoli allora anche la chiave esportata che rappresenta il legame con l'altra classe va indicata nel vincolo.

Le regole di corrispondenza tra GeoUML e il modello SFS

Mapping degli attributi enumerati

Per ogni *attributo enumerato* A_{enum} con valori $(v_1, ..., v_n)$, aggiungere alla tabella che rappresenta la classe *C* che contiene A_{enum} il seguente vincolo:

vincolo C_A_{enum} (A_{enum} IN (v_1, …, v_n))

(sintassi **vincolo** <nome> (<nome attributo> **IN** (<elenco valori>)))

Per ogni *attributo enumerato* A_{enum} con *dominio enumerato D* contenente i valori $(v_1, ..., v_n)$, si genera un dominio indicando il tipo e il vincolo:

dominio D_C_A_{enum} tipo varchar(x)

vincolo (value IN (v_1, …, v_n))

(sintassi **dominio** <nome> **tipo** <tipoSQL> **vincolo** (**value IN** (<elenco valori>)))

e si assegna il dominio D_C_A_{enum} come tipo dell'attributo A_{enum} nella tabella *C*.

Le regole di corrispondenza tra GeoUML e il modello SFS

Mapping degli attributi enumerati gerarchici

Per ogni *attributo enumerato gerarchico* A_{enumG} della classe *C* con dominio specificato ad esempio nel seguente modo:

(v_1 **A_1:(**$v_{1,1}$ **B_1:(**$v_{1,1,1}$, $v_{1,1,2}$**)**, $v_{1,2}$, $v_{1,3}$**)**, v_2 **$A_{2,1}$:(**$v_{2,1}$, $v_{2,2}$**)** **$A_{2,2}$:(**$v_{2,3}$**,** $v_{2,4}$**)**, v_3, v_4**)**

aggiungere una tabella con la seguente struttura:

C_A_{enumG} ($\underline{idA_{enumG}}$, value, A_1, B_1, $A_{2,1}$, $A_{2,2}$)

con i vincoli seguenti:

vincolo C_value (value IN (v_1, v_2 , v_3 , v_4))

vincolo C_ A_1 ((value= v_1 AND A_1 IN ($v_{1,1}$, $v_{1,2}$, $v_{1,3}$)) OR
(value<> v_1 AND A_1 is NULL))

…

(sintassi **vincolo** <nome> (<espressione clausola WHERE di SQL>))

e aggiungere alla tabella ***C*** l'attributo **idA_{enumG}** chiave esportata della tabella precedente.

Le regole di corrispondenza tra GeoUML e il modello SFS

Classe C_1 contenente un attributo geometrico G_1.

Mapping degli attributi geometrici

Si aggiunge un attributo geometrico G_1 alla relazione C_1 con dominio geometrico scelto secondo la corrispondenza mostrata nel lucido successivo

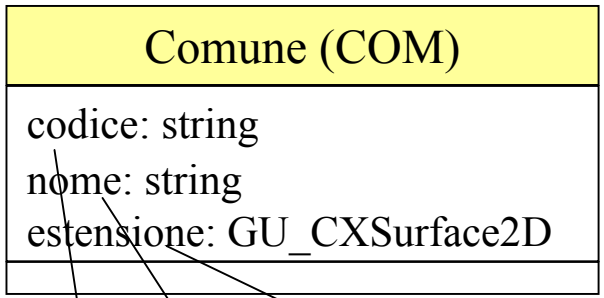
Corrispondenza tra tipi geometrici GeoUML e tipi geometrici SFS

Tipo GeoUML	Tipo SFS
GU_Point*	Point
GU_CPCurve*	LineString
GU_CPRing*	LineString + vincolo
GU_CPSurface2D	Polygon
GU_CXPoint*	Multipoint + vincolo
GU_CXCurve*	Multilinestring + vincolo
GU_CNCurve*	Multilinestring + vincolo
GU_CXRing*	Multilinestring + vincolo
GU_CXSurface2D	Multipolygon

Tipo GeoUML	Tipo SFS
GU_Complex*	GeometryCollection + vincolo
GU_Aggregate*	GeometryCollection
GU_MPoint*	Multipoint
GU_MCurve*	GeometryCollection + vincolo
GU_MRing*	GeometryCollection + vincolo
GU_MSurface2D	GeometryCollection + vincolo

N.B.: si noti che tutti i tipi 3D degenerano allo spazio 2D.

Esempio di traduzione



FID

Comune (idComune, codice, nome, estensione: Multipolygon)

Le regole di corrispondenza tra GeoUML e il modello SFS

Associazione A tra una Classe C_1 e Classe C_2.

Mapping delle associazioni

Se l'associazione è uno a molti o uno a uno si esporta il feature id della classe coinvolta con cardinalità molti nella tabella della classe coinvolta con cardinalità uno usando il nome del ruolo come nome dell'attributo.

$\mathbf{C_1(\underline{idC_1}, A_1, ..., A_n, \boxed{ruoloC_2})} \longrightarrow \mathbf{C_2(\underline{idC_2}, B_1, ..., B_m)}$

Se l'associazione è molti a molti si genera una tabella esplicita per rappresentare l'associazione:

$\mathbf{C_1(\underline{idC_1}, A_1, ..., A_n, idC_2)}$ → $\mathbf{C_2(\underline{idC_2}, B_1, ..., B_m)}$

$\mathbf{A(\boxed{\underline{ruoloC_1}}, \boxed{\underline{ruoloC_2}}, D_1, \ldots, D_k)}$

dove $D_1, \ldots, D_k$ sono gli eventuali attributi dell'associazione.

Le regole di corrispondenza tra GeoUML e il modello SFS

Classe tratto (sottoarea) T_1 contenente gli attributi alfanumerici:

$B_1, ..., B_n$

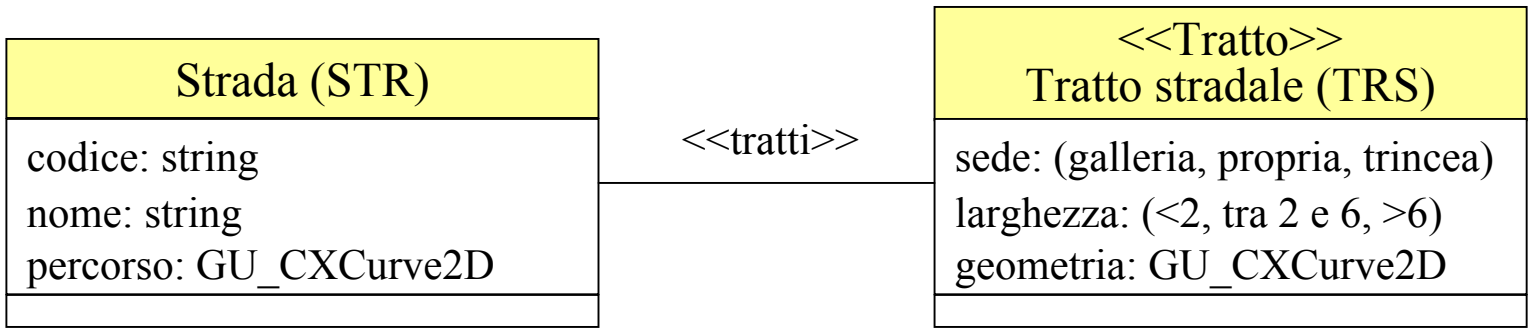
Mapping delle classi tratto (sottoarea)

Per la classe tratto (sottoarea) *T* viene creata una relazione (tabella) *T* che contiene tanti attributi quanti sono gli attributi alfanumerici della classe *T* e un attributo geometrico detto *geometria* di tipo Multilinestring (Multipolygon).

La relazione (tabella) T_1 ha un attributo aggiuntivo detto *SID* (segment or subregion ID) di tipo integer e con nome *sidT*. Tale attributo insieme al feature id della classe "padre del tratto" C_i (detto *id* C_i) diventa chiave primaria della tabella *T*.

T(sidT, idC_i, B_1, ..., B_n, geometria: Multilinestring)

Esempio di traduzione



Strada (idSTR, codice, nome, percorso: Multilinestring)

TrattoStradale (sidTRS, idSTR, sede, larghezza,
geometria: Multilinestring)
vincolo TRS_sede (sede **IN** (‘galleria’, ‘propria’, ‘trincea’))
vincolo TRS_larghezza (larghezza **IN** (‘<2’, ‘tra 2 e 6’, ‘>6’))

Le regole di corrispondenza tra GeoUML e il modello SFS

Mapping degli strati topologici

Per ogni strato topologico presente nello schema GeoUML si genera dove possibile uno strato topologico a nello schema logico, facendo confluire sullo strato tutti gli attributi geometrici direttamente o indirettamente vincolati allo strato.

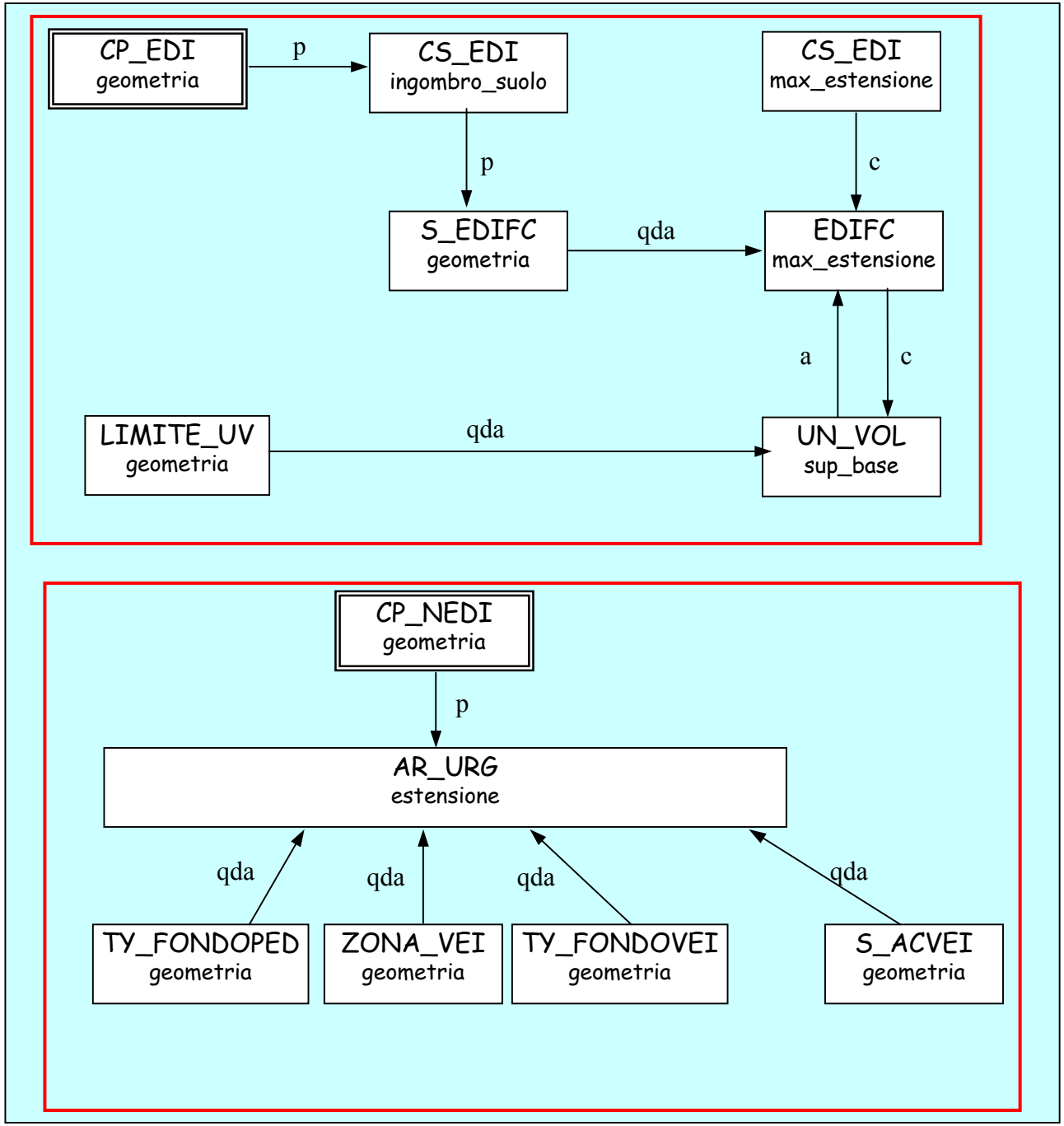
Le regole di corrispondenza tra GeoUML e il modello SFS

Generazione degli strati topologici

Nel caso in cui sia possibile generare strati topologici a livello logico, allora dato uno schema GeoUML è possibile applicare il seguente procedimento per la generazione degli strati:

- generare il grafo dei vincoli strutturali (**grafo strutturale**): ogni attributo geometrico, classe tratto/sottoarea o strato è uno nodo e ogni vincolo strutturale è un arco non orientato)
- generare i **sottografi connessi** del grafo strutturale
- **per ogni sottografo connesso generare uno strato topologico a livello logico** e associare allo strato tutti gli attributi geometrici presenti nel sottografo, comprese le geometrie dei tratti e delle sottoaree ed esclusa la geometria dello strato topologico concettuale se è l'unico strato del sottografo.

Esempio


CP_EDI
geometria
p
CS_EDI
ingombro_suolo
CS_EDI
max_estensione
p
c
S_EDIFC
geometria
qda
EDIFC
max_estensione
a
c
LIMITE_UV
geometria
qda
UN_VOL
sup_base
CP_NEDI
geometria
p
AR_URG
estensione
qda
qda
qda
qda
TY_FONDOPED
geometria
ZONA_VEI
geometria
TY_FONDOVEI
geometria
S_ACVEI
geometria

Mapping degli strati (ESEMPIO)

Strato CP_EDI

CS_EDI.ingombro_suolo → CP_EDI
CS_EDI.max_estensione → CP_EDI
S_EDIF.geometria → CP_EDI
EDIFC.max_estensione → CP_EDI
UN_VOL.sup_base → CP_EDI
LIMITE_UV.geometria → CP_EDI

Strato CP_NEDI

AR_URG.ingombro_suolo → CP_NEDI
TY_FONDOPED.geometria → CP_NEDI
TY_FONDOVEI.geometria → CP_NEDI
ZONA_VEI.geometria → CP_NEDI
S_ACVEI.geometria → CP_NEDI