POLITECNICO DI MILANO

CISIS Centro Interregionale per i Sistemi informatici, geografici e statistici
Comitato permanente per i Sistemi Informativi Geografici

Guida ai Modelli Implementativi di Tipo Flat

1 febbraio 2012

SpatialDBgroup
SpatialDBgroup@polimi.it
http://SpatialDBgroup.polimi.it

Autore	Politecnico di Milano – Spatial DB Group	Giuseppe Pelagatti (coordinatore), Alberto Belussi, Jody Marca, Mauro Negri
Coordinamento delle attività	CISIS – CPSG Comitato di Progetto	Maurizio De Gennaro (Regione del Veneto – coordinatore tecnico), Massimo Attias (CISIS-referente area geografica e progetti) Stefano Olivucci (Regione Emilia-Romagna), Raffaella Gelleti, Marco Lunardis, Massimo Zia (Regione Friuli Venezia Giulia), Massimiliano Basso, Alessandra Chiarandini (INSIEL-Regione FVG), Simone Patella (Regione Lazio), Gianbartolomeo Siletto (Regione Piemonte), Mauro Vasone (CSI Piemonte), Marco Guiducci, Andrea Peri (Regione Toscana), Gianfranco Amadio, Domenico Bertoldi, Gianluca Riscaio, Sandra Togni (Regione Umbria), David Freppaz (Regione Valle d'Aosta), Virgilio Cima, Umberto Trivelloni (Regione del Veneto), Leonardo Donnaloia, Claudio Mazzi, Pierpaolo Milan (CISIS)
	CISIS –CPSG Struttura di supporto interna	Massimo Attias, (coordinatore struttura), Leonardo Donnaloia, Claudio Mazzi, Pierpaolo Milan, Antonio Rotundo (CISIS)

INDICE

PREMESSA

I Modelli Implementativi di tipo “flat” trasformano uno schema GeoUML in una struttura “piatta”, basata sostanzialmente su una rappresentazione tabellare dei dati senza nidificazioni (cioè senza possibilità di rappresentare come elemento di una tabella un’altra tabella).

Allo stato attuale esistono 5 MI di tipo flat, classificabili nel modo seguente:

- Esistono 3 MI orientati ai Database Georelazionali (SQL); a loro volta questi si suddividono in
  - MI SQL multigeometria, nei quali una tabella relazionale può contenere più di un attributo geometrico; di questi esiste una versione in tecnologia ORACLE
  - MI SQL monogeometria, nei quali una tabella relazionale può contenere solamente un attributo geometrico – questa limitazione è stata introdotta per supportare gli strumenti GIS che non sono in grado di operare sul modello multigeometria - di questi esistono 2 versioni in base alla tecnologia (ORACLE o POSTGIS)
- Esistono 2 MI di trasferimento basati su tecnologia Shape, che si distinguono in base alla rappresentazione della geometria:
  - SHAPE_FLAT rappresenta la geometria oggetto per oggetto, come tutti gli altri MI classificati come MI a oggetti
  - SHAPE_TOPO utilizza una strutturazione topologica delle geometrie

Tutti questi MI si basano su un numero limitato di principi fondamentali. La comprensione di questi principi fondamentali, di natura strutturale, semplifica la comprensione dei singoli MI, che sono resi complicati da una serie di dettagli relativi alla generazione dei nomi, alle regole di tipo, ecc.. proprie delle varie tecnologie, per i quali si rimanda ai documenti relativi ad ogni singolo modello.

Il documento è diviso in due parti:

- Parte 1: Regole fondamentali di Mapping: illustrate con esemplificazione basata sui Modelli Implementativi SHAPE_FLAT e SQL_MONO_PostGIS
- Parte 2: Varianti adottate da altri MI
  - MI SQL multigeometria Oracle
  - MI topologico SHAPE_TOPO

Il documento non tratta esplicitamente il MI monogeometria Oracle: la sua struttura è identica a quella del MI monogeometria PostGIS e i suoi tipi geometrici sono quelli descritti nel MI SQL multigeometria Oracle.

Popolamento

Un aspetto di cui è necessario tenere conto nella generazione del mapping è il popolamento delle classi; in tutti i MI si applica la seguente Regola riguardante il popolamento:

> *per ogni costrutto istanziato nella specifica di contenuto e dichiarato come popolato ad almeno un livello di scala, si generano le corrispondenti strutture fisiche secondo quanto indicato nei capitoli successivi. I costrutti non popolati non vengono considerati nel mapping e per essi non si genera quindi alcuna struttura fisica.*

Parte I
Regole fondamentali di mapping Flat
(illustrate con riferimento ai MI SHAPE_FLAT e SQL_MONOGEOMETRIA_PostGIS)

1. INTRODUZIONE

Terminologia.
In questa parte del documento per esprimere regole di mapping che valgono sia sui MI basati su Shapefile che su SQL, ci si riferisce con il termine ***Shape/Table*** ad una struttura fisica che sarà uno Shapefile nel MI ShapeFlat, una Table nel MI SQL_monogeometria_PostGIS; il termine ***attributi*** mantiene il suo significato nei due casi. Nel caso in cui non ci sia geometria da rappresentare, la struttura fisica si indica con il termine ***FileDBF/Table***.

Esempi
L'esemplificazione fa riferimento ai contenuti del National Core, dal quale il Catalogue ha generato la documentazione fisica.
Gli esempi riportati sono tutti estratti dalla documentazione prodotta dal Catalogue, che è differenziata in base alle tecnologie SQL e Shapefile:

- per il MI SQL vengono riportati estratti dello ***Script di Generazione*** del Database (Create Table)
- per il MI ShapeFlat vengono riportati estratti del ***Report di Mapping***

Per rendere gli esempi nelle due tecnologie il più simili possibile, le opzioni comuni delle DPS utilizzate per generare i mapping sono state configurate in maniera uguale:

- implementazione sottoaree in *3D*
- implementazione tratti e sottoaree *connessa*

Inoltre, le opzioni presenti solo nel MI SQL sono state rese omogenee alla soluzione adottata negli MI Shapefile:

- nessun prefisso nella generazione nomi delle tabelle SQL (opzione MI SQL)
- generazione nomi tabelle dal codice alfanumerico

In questo modo i nomi generati per le Table SQL risultano uguali ai corrispondenti nomi generati per gli Shapefile.
NOTA: In questo documento non vengono discusse le opzioni delle DPS e le modalità operative per definirle; per questo argomento si rimanda al documento "Guida all'uso del GeoUML Catalogue".

Integrità referenziale
Un aspetto fondamentale di una struttura tabellare ben fatta è costituito dall'integrità referenziale (o foreign key constraint) cioè dal vincolo che i valori di un attributo di uno Shape/Table siano contenuti nei valori di un attributo di un altro Shape/Table.
L'integrità referenziale può essere esplicitamente dichiarata in SQL in modo da essere controllata automaticamente dal DBMS relazionale. Tuttavia, nei MI di tipi SQL_Monogeometria è possibile disabilitare la generazione di tali dichiarazioni, in modo da essere compatibili con alcuni sistemi GIS molto diffusi. Si noti che in ogni caso il validatore controllerà i vincoli di integrità referenziale.
Negli Shapefile non esiste nessun meccanismo di dichiarazione e verifica dell'integrità referenziale.
Dato che l'integrità referenziale è fondamentale per garantire la correttezza dei dati, in questo documento le proprietà di integrità referenziale sono indicate anche se non generano alcuna dichiarazione nello schema fisico; la notazione utilizzata per rappresentare il vincolo che i valori di un attributo A1 di uno Shape/Table SH1 siano contenuti nei valori di un attributo A2 di un altro Shape/Table SH2 è la seguente:

SH1.A1 **IN** SH2.A2

2. MAPPING DI CLASSI MONOGEOMETRIA CON ATTRIBUTI MONOVALORE

2.1 Struttura fondamentale

La struttura degli Shape/Table corrispondenti a una classe GeoUML dotata di una sola componente spaziale e di soli attributi monovalore è molto semplice:

- un unico Shape/Table è sufficiente
- il nome dello Shape/Table è uguale al ***codice alfanumerico*** della classe
- un attributo descrittivo nello Shape/Table per ogni attributo descrittivo della classe popolato
- un attributo geometrico per la componente spaziale della classe; nel caso degli Shapefile l'attributo geometrico è costituito dalla geometria dello Shapefile stesso
- aggiunta di un attributo chiamato ***ClassID*** che svolge il compito di identificatore unico (nello Schema Concettuale è implicito)
- per gli attributi di tipo Datatype del GeoUML i singoli attributi del Datatype sono trasformati in attributi dello Shape/Table
- gli attributi di attributi geometrici sono riassorbiti nella classe
- i nomi degli attributi coincidono con il codice alfanumerico degli attributi.

In sostanza, in questo caso gli elementi di una struttura Shape/Table corrispondono in maniera molto semplice agli elementi della Classe GeoUML.
Sono necessarie alcune precisazioni:

- il tipo dell'attributo geometrico è derivato dal tipo GeoUML della componente spaziale della classe secondo le regole di Tabella 2.1, nella quale per i modelli SQL è riportato il tipo del Simple Feature Model, cioè lo standard di riferimento per i diversi sistemi SQL (più un predicato di precisazione, ove applicabile)

Tipo GeoUML	Tipo SFM	Tipo dello shapefile
GU_Point2D	Point (x,y)	**Point**
GU_Point3D	Point (x,y,z)	**PointZ**
GU_CPCurve2D	LineString(x,y)	**Polyline – 1 part**
GU_CPCurve3D	LineString(x,y,z)	**PolylineZ – 1 part**
GU_CPSimpleCurve2D	LineString(x,y): IsSimple()=true	**Polyline – 1 part**
GU_CPSimpleCurve3D	LineString(x,y,z): IsSimple()=true	**PolylineZ – 1 part**
GU_CPRing2D	LineString(x,y): IsClosed()=true	**Polyline – 1 part**
GU_CPRing3D	LineString(x,y,z): IsClosed()=true	**PolylineZ – 1 part**
GU_CPSurface2D	Polygon(x,y)	**Polygon**
GU_CXPoint2D	MultiPoint (x,y)	**MultiPoint**
GU_CXPoint3D	MultiPoint (x,y,z)	**MultiPointZ**
GU_CXCurve2D	MultiLineString(x,y)	**Polyline**
GU_CXCurve3D	MultiLineString(x,y,z)	**PolylineZ**
GU_CXRing2D	MultiLineString(x,y): isClosed=true	**Polyline**
GU_CXRing3D	MultiLineString(x,y,z): isClosed=true	**PolylineZ**
GU_CNCurve2D	MultiLineString(x,y): connected	**Polyline**
GU_CNCurve3D	MultiLineString(x,y,z): connected	**PolylineZ**
GU_CXSurface2D	MultiPolygon(x,y)	**Polygon**
GU_CPSurfaceB3D	Polygon(x,y,z)	**PolygonZ**
GU_CXSurfaceB3D	MultiPolygon(x,y,z)	**PolygonZ**
GU_Aggregate2D	GeometryCollection(x,y)	***vedi nota sotto***
GU_Aggregate3D	GeometryCollection(x,y,z)	***vedi nota sotto***
*Nota: **g**li Aggregati GeoUML non hanno un tipo equivalente tra quelli degli Shapefile, pertanto un attributo di questo tipo richiede uno Shapefile diverso per ogni componente di base di un aggregato: Multipoint, Polyline e Polygon per gli aggregati 2D e MultipointZ e PolylineZ per gli aggregati 3D.*		

Tabella 2.1

- il tipo degli attributi descrittivi è derivato dal tipo GeoUML dei corrispondenti tipi degli attributi descrittivi della classe secondo le regole di Tabella 2.2
- il tipo dell'attributo ClassID è "Stringa(70)"
- per gli attributi enumerati e gerarchici si applicano alcune regole aggiuntive elencate sotto.

Tipo GeoUML	Tipo Shape_Flat	Tipo SQL/PostGIS
Integer	**Tipo N - massimo 9 cifre (0 decimali)**	`numeric(15,0)`
Real	**Tipo N - massimo 10 cifre (3 decimali)**	`double precision`
String(x)	**Tipo C - lunghezza x (valore massimo di x è 254)**	**varchar(x)**
NumericString(x)	**Tipo C - lunghezza x (il valore massimo di x è 254)**	**varchar(x)**
Date	**Tipo D**	**date**
DateTime	**Tipo C – lunghezza 19 (formato *gg/mm/aaaa hh:mm:ss*)**	**timestamp**
Time	**Tipo C – lunghezza 8 (formato *hh:mm:ss*)**	**time**
Boolean	**Tipo N – massimo 2 cifre senza decimali (0 falso e 1 vero)**	**boolean**

Tabella 2.2

ESEMPIO 2.1

Consideriamo la definizione nello Schema Concettuale (SC) di una classe semplice, la classe "**Albero isolato**".

- il codice alfanumerico della classe è **ALBERO**.
- la classe ha una componente spaziale **Posizione,** con codice alfanumerico **ALBERO_POS** di tipo **GU_Point3D**
- l'unico attributo descrittivo è **ALBERO_TY** (tipo)

Si riportano di seguito gli estratti della documentazione prodotta dal Catalogue nei due MI.

ShapeFlat (estratti dal Report di Mapping)

File: **ALBERO**

Classe 060403 - Albero isolato - ALBERO

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ALBERO_POS	GU_POINT3D	POINTZ	Attributo geometrico 060403101 - Posizione - ALBERO_POS della classe 060403 - Albero isolato - ALBERO
ALBERO_TY	Stringa (80)	Stringa (80)	Attributo enumerato 06040301 - tipo - ALBERO_TY
ClassID	Stringa (70)	Stringa (70)	Identificativo univoco
ScRil	Stringa (80)	Stringa (80)	Riferimento al codice di livello di scala

PostGIS (estratti dallo Script di Generazione)

```
CREATE TABLE: albero
-- Rappresenta la classe: Albero isolato - 060403
--

CREATE TABLE albero (
        classid  varchar(70) NOT NULL,
        scril  varchar(80) NOT NULL,
        albero_ty  varchar(80) NOT NULL
);
SELECT addgeometrycolumn('', 'albero', 'albero_pos', '32632', 'POINT', 3);
```

2.2 Domini enumerati e gerarchici

Per ogni dominio (enumerato ed enumerato gerarchico) del GeoUML utilizzato da un attributo enumerato della classe deve esistere un FileDBF/Table che contiene i possibili valori dell'enumerato; il nome di tale file è generato secondo la regola seguente:

- i nomi dei FileDBF/Table di dominio iniziano tutti con D_ per i domini normali, con H_ per quelli gerarchici
- sono seguiti dal codice alfanumerico del dominio GeoUML per i domini dichiarati separatamente
- dal nome della classe e dell'attributo enumerato GeoUML per i domini embedded.

La struttura di un FileDBF/Table di dominio è mostrata in tabella 2.3.

Nome attributo	Tipo	Descrizione
CODE	String(80)*	**Valore (codice) presente negli attributi che usano questo dominio**
NAME	String(160)	**Transcodifica del codice – Valore del dominio**
DEFINITION	String(254)	**Descrizione del valore di dominio**
ALPHACODE	String(80)	**Ulteriore codice alfanumerico**

TABELLA 2.3

Infine, vale il seguente vincolo di integrità referenziale: ogni attributo enumerato (o enumerato gerarchico) A di un FileDBF/Table S deve soddisfare rispetto all'attributo CODE del corrispondente file D che descrive il suo dominio:

S.A IN D.CODE

ESEMPIO 2.1 (continua)

Nell'esempio 2.1 il dominio dell'attributo ALBERO_TY è di tipo enumerato embedded ed è stato tradotto con un attributo di tipo string(80). I valori presenti in tale attributo devono essere un sottoinsieme di quelli del dominio, pertanto deve esistere un FileDBG/Table relativo al dominio che contiene tali valori.

ShapeFlat (estratti dal Report di Mapping)

Nella sezione relativa ai File DBF di transcodifica del documento di mapping troviamo la riga seguente:

ALBERO	ALBERO_TY	D_ALBERO_TIPO_NI

che indica l'esistenza di un FileDBF contenente i valori del dominio di tale enumerato; il nome di tale file è, coerentemente con le regole illustrate per i domini embedded, D_ALBERO_TIPO_NI, dove:

- D è il prefisso di tutti i domini enumerati (non gerarchici)

- ALBERO è il nome della classe
- TIPO è il nome dell'attributo (non il suo codice alfanumerico)
- NI è un suffisso che indica che nel dominio è inclusa la interpretazione dei valori nulli, perché l'attributo è opzionale (l'argomento "interpretazione dei valori nulli" è trattato più avanti)

Il file DBF D_ALBERO_TIPO_NI generato automaticamente dal Catalogue contiene, in base alle definizioni presenti nello Schema Concettuale, i seguenti 2 record:

CODE	NAME	DEFINITION	ALFACODE
01	monumentale		
95	altro	Valore assunto dall'istanza ma non previsto dalla specifica	

PostGIS (estratti dallo Script di Generazione)

```
-- CREATE TABLE: d_albero_tipo
-- Rappresenta la tabella di transcodifica del dominio: Tipo -
0604030100
--
CREATE TABLE d_albero_tipo (
     code  varchar(80) NOT NULL,
     name  varchar(160),
     definition  varchar(1200),
     alphacode  varchar(80)
);

--
-- INSERT INTO TABLE: d_albero_tipo
-- Valori di trascodifica per il dominio: Tipo - 0604030100
--
INSERT INTO d_albero_tipo (name,code,alphacode,definition) VALUES
('altro','95',null,'Valore assunto dall'istanza ma non previsto dalla
specifica.');
INSERT INTO d_albero_tipo (name,code,alphacode,definition) VALUES
('monumentale','01',null,null);
```

2.2 Interpretazione dei valori nulli

Secondo il modello GeoUML gli attributi opzionali possono contenere un valore nullo. Poiché tale valore non è disponibile nei file DBF, nella definizione di una DPS, che adotti il modello implementativo ShapeFlat, viene stabilito quali siano i valori che nei diversi domini di base rappresentano il valore nullo. Qualora la specifica preveda l'interpretazione del valore nullo, nella DPS dovranno essere specificati i valori sostitutivi per ciascuna possibile interpretazione del valore nullo. Infine, come mostrato nell'esempio 2.1 i valori scelti per rappresentare le diverse interpretazioni del valore nullo nei domini enumerati vengono riportati nel file DBF che contiene tutti i valori ammessi per il dominio (valore '95' nell'esempio).

Nel MI SQL il valore nullo esiste e quindi viene usato per rappresentare l'assenza di un valore in un attributo opzionale. Per quanto riguarda l'interpretazione del valore nullo, se richiesta, le regole del modello MI SQL prevedono che si generi per ogni classe che contiene attributi opzionali una tabella aggiuntiva con le seguenti caratteristiche:

- il nome della Table coincide con il nome della classe con il suffisso _NI
- la tabella contiene un attributo ClassREF e tanti attributi di tipo varchar, uno per ogni attributo opzionale della classe.

Se un attributo opzionale di una riga della corrispondente tabella di classe contiene un valore nullo, esisterà una riga in questa tabella che riporta nel corrispondente attributo il valore che indica l'interpretazione del valore nullo. Ciò vale anche per gli attributi opzionali multivalore e per gli attributi datatype con componenti opzionali.
Infine, anche gli attributi geometrici possono essere opzionali: in MI ShapeFlat si utilizza la variante "nullshape" degli shapefile, mentre in MI SQL si utilizza il valore nullo. Non esiste interpretazione del valore nullo per gli attributi geometrici.

PostGIS (estratti dallo Script di Generazione)

```
--
-- CREATE TABLE: albero_ni
--

CREATE TABLE albero_ni (
     classref  varchar(70) NOT NULL,
     albero_ty_ni  varchar(80)
);
```

Tabelle simili si generano in presenta di attributi a tratti, a sottoaree o a eventi con cardinalità minima 0.

3. MAPPING DEGLI ATTRIBUTI MULTIVALORE E DEI DATATYPE

3.1 Attributi Multivalore

Per ogni attributo di una classe con cardinalità massima maggiore di uno (attributo **multivalore)** viene creato un FileDBF/Table apposito.
Il nome del FileDBF/Table è ottenuto concatenando il codice alfanumerico della classe con il codice alfanumerico dell'attributo.
Le colonne del FileDBF/Table sono:

- riferimento all'identificatore della classe al quale appartiene l'attributo, denominato **ClassREF** ; tale riferimento ha un vincolo di integrità referenziale verso l'identificatore ClassID contenuto nello Shape/Table della classe
- il nome dell'attributo multivalore stesso

La coppia di attributi è univoca all'interno del FileDBF/Table.

ESEMPIO 3.1
La classe **Galleria**, con codice alfanumerico **GALLER**, ha un attributo **uso**, codice alfanumerico **GALLER_USO**, multivalore.

ShapeFlat (estratti dal Report di Mapping)

File: **GALLER_GALLER_USO**

Attributo multivalore 02030302 - uso - GALLER_USO della classe 020303 - GALLER - Galleria

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020303 - GALLER - Galleria
GALLER_USO	Stringa (80)	Stringa (80)	Attributo enumerato 02030302 - uso - GALLER_USO

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:
GALLER_GALLER_USO.ClassREF IN GALLER.ClassID

PostGIS (estratti dallo Script di Generazione)

```
--
-- CREATE TABLE: galler_galler_uso
-- Rappresenta la tabella di appoggio per l'attributo multivalore: uso
- 02030302
--

CREATE TABLE galler_galler_uso (
     classref  varchar(70) NOT NULL,
     galler_uso  varchar(80) NOT NULL
);
```

3.2 Datatype Multivalore

Un attributo monovalore di tipo Datatype costituito da N attributi semplici viene sostituito nello Shapefile dalla rappresentazione diretta degli attributi semplici, come indicato nella regola base di traduzione di una classe.
In presenza di un attributo multivalore di tipo Datatype questa regola si combina con quella del paragrafo precedente e quindi viene creato un FileDBF/Table apposito.
Il nome del FileDBF/Table è ottenuto concatenando il codice alfanumerico della classe con il codice alfanumerico dell'attributo con il suffisso _T.
Le colonne del FileDBF/Table sono:

- riferimento all'identificatore della classe al quale appartiene l'attributo, denominato ClassREF ; tale riferimento ha un vincolo di integrità referenziale verso l'identificatore ClassID contenuto nello Shape/Table della classe
- il nome di tutti gli attributi che costituiscono il Datatype

ESEMPIO 3.2

Nella classe COMUNE l'attributo COMUNE_NOM (nome del comune) appartiene al dominio MULTILING (multilinguismo), che è un Datatype costituito da una coppia di attributi <LINGUA, NOME>. Tale attributo è multivalore, permettendo in tal modo di associare a un comune diverse coppie <lingua, nome>. Di seguito vengono mostrati gli estratti dei mapping generati.

ShapeFlat (estratti dal Report di Mapping)

File: **COMUNE_COMUNE_NOM_T**

DataType 80 - Multilinguismo - MULTILING dell'attributo 09010102 - nome comune - COMUNE_NOM della classe 090101 - COMUNE - Comune

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 090101 - COMUNE - Comune
LINGUA	Stringa (80)	Stringa (80)	Attributo enumerato di DataType 02 - lingua - LINGUA del DataType80 - Multilinguismo - MULTILING
NOME	Stringa (100)	Stringa (100)	Attributo semplice di DataType 01 - nome - NOME del DataType80 - Multilinguismo - MULTILING

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:
COMUNE_COMUNE_NOM.ClassREF IN COMUNE.ClassID

PostGIS (estratti dallo Script di Generazione)

```
-- CREATE TABLE: comune_comune_nom_t
-- Rappresenta il datatype: Multilinguismo - 80
--
CREATE TABLE comune_comune_nom_t (
      classref  varchar(70) NOT NULL,
      comune_n_lingua  varchar(80),
      comune_nom_nome  varchar(100)
);
```

4. MAPPING DELLE ASSOCIAZIONI

4.1 Associazioni 1xN

Le Associazioni di tipo 1xN vengono implementate con attributi di ruolo nello Shapefile della classe C che si collega a N istanze dell'altra classe.
Viene quindi aggiunto un attributo nello Shapefile della classe C con nome uguale al codice alfanumerico del ruolo di C nell'associazione; tale attributo conterrà per ogni oggetto della classe C l'identificatore ClassID dell'oggetto referenziato nell'associazione e dovrà soddisfare il corrispondente vincolo di integrità referenziale.

ESEMPIO 4.1

Nella classe Comune esiste un ruolo ***pvdicm*** che associa al Comune la Provincia di appartenenza. Tale ruolo compare nei due mapping generati.

ShapeFlat (estratti dal Report di Mapping)

File: **COMUNE**

Classe 090101 - Comune - COMUNE

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
COMUNE_IST	Stringa numerica (x) (16)	Stringa numerica (0)	Attributo semplice 09010101 - codice istat comune - COMUNE_IST
ClassID	Stringa (70)	Stringa (70)	Identificativo univoco
PVDICM	Stringa (70)	Stringa (70)	Ruolo 09010172 - Pvdicm - PVDICM

PVDICM deve soddisfare il vincolo di integrità referenziale seguente:
COMUNE.PVDICM IN PROVINCIA.ClassID

PostGIS (estratti dallo Script di Generazione)

```
--
-- CREATE TABLE: comune
-- Rappresenta la classe: Comune - 090101
--

CREATE TABLE comune (
      classid  varchar(70) NOT NULL,
      comune_ist  varchar(16) NOT NULL,
      pvdicm  varchar(70) NOT NULL
);
```

4.2 Associazione NxN

Un'associazione NxN con o senza attributi richiede un FileDBF/Table aggiuntivo composto di due attributi contenenti gli identificatori degli oggetti delle classi coinvolti nell'associazione. La classe 1 è considerata quella che appare come prima in ordine alfabetico.
Il nome del FileDBF/Table è A_NomeClasse1_nomeRuoloClasse1_NomeClasse2.
Il FileDBF/Table contiene due colonne denominate: NomeRuoloClasse1 e NomeRuoloClasse2. Esse contengono gli identificatori degli oggetti delle classi coinvolti nell'associazione.
Un'associazione NxN con attributi aggiunge ai due identificatori gli attributi dell'associazione il cui nome si genera a partire dai relativi nomi. In presenza di attributi multivalore o di tipo Datatype ci si comporta come nel caso delle classi sostituendo ovviamente il codice alfanumerico della classe con quello dell'associazione preceduto dal prefisso **A_**.

5. MAPPING DI CLASSI CON PIU' DI UNA COMPONENTE SPAZIALE

Se una classe possiede più di una componente spaziale (anche dopo aver scartato le componenti non popolate), le componenti spaziali vengono mappate su Shape/Table separati, mentre tutti gli attributi descrittivi vengono mappati su un FileDBF/Table privo di componente spaziale, seguendo le regole viste al capitolo 2 per le classi monogeometria.
Per gli Shape/Table dedicati a rappresentare le componenti spaziali valgono le seguenti regole:

- Il nome degli Shape/Table associati ad ogni componente spaziale è composto dalla concatenazione del codice alfanumerico della classe e del codice alfanumerico dell'attributo geometrico rappresentato nello specifico Shape/Table
- Il tipo di ogni Shape/Table è determinato in base alla tabella 2.1, come per le classi monogeometria
- In ogni Shape/Table è presente un attributo chiamato **ClassREF** di tipo varchar(70) che contiene l'identificatore dell'oggetto della classe al quale appartiene la geometria, con un vincolo di integrità referenziale verso tale attributo

Il nome dello FileDBF /Table dedicato a rappresentare i soli attributi descrittivi corrisponde al codice alfanumerico della classe stessa assegnato nella specifica concettuale

ESEMPIO 5.1
Consideriamo la classe DIGA, che possiede ben 4 componenti spaziali:

- Coronamento
- Sostegno esterno
- Sostegno interno
- Superficie di riferimento

L'applicazione delle regole viste nei corrispondenti mapping porta alla creazione di 4 Shape/Table per le componenti spaziali e un FileDBF/Table per gli attributi descrittivi della classe.

ShapeFlat (estratti dal Report di Mapping)

File: **DIGA** **Classe 020501 - Diga - DIGA**			
NOME	**TIPO GEOUML**	**TIPO**	**DESCRIZIONE**
ClassID	Stringa (70)	Stringa (70)	Identificativo univoco
DIGA_CLASS	Stringa (80)	Stringa (80)	Attributo enumerato 02050102 - classificazione ufficiale - DIGA_CLASS
altri attributi…			

File:	DIGA_DIGA_CR
	Attributo geometrico 020501104 - Coronamento - DIGA_CR della classe 020501 - DIGA - Diga

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020501 - DIGA - Diga
DIGA_CR	GU_CPSURFACEB3D	POLYGONZ	Attributo geometrico 020501104 - Coronamento - DIGA_CR della classe 020501 - Diga - DIGA

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

DIGA_DIGA_CR.ClassREF IN DIGA.ClassID

File:	DIGA_DIGA_SE
	Attributo geometrico 020501102 - Sostegno_esterno - DIGA_SE della classe 020501 - DIGA - Diga

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020501 - DIGA - Diga
DIGA_SE	GU_CPSURFACEB3D	POLYGONZ	Attributo geometrico 020501102 - Sostegno_esterno - DIGA_SE della classe 020501 - Diga - DIGA

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

DIGA_DIGA_SE.ClassREF IN DIGA.ClassID

File:	DIGA_DIGA_SI
	Attributo geometrico 020501103 - Sostegno_interno - DIGA_SI della classe 020501 - DIGA - Diga

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020501 - DIGA - Diga
DIGA_SI	GU_CPSURFACEB3D	POLYGONZ	Attributo geometrico 020501103 - Sostegno_interno - DIGA_SI della classe 020501 - Diga - DIGA

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

DIGA_DIGA_SI.ClassREF IN DIGA.ClassID

File: DIGA_DIGA_SUP Attributo geometrico 020501101 - Sup_riferimento - DIGA_SUP della classe 020501 - DIGA - Diga			
NOME	**TIPO GEOUML**	**TIPO**	**DESCRIZIONE**
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020501 - DIGA - Diga
DIGA_SUP	GU_CXSURFACEB3D	POLYGONZ	Attributo geometrico 020501101 - Sup_riferimento - DIGA_SUP della classe 020501 - Diga - DIGA

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

DIGA_DIGA_SUP.ClassREF IN DIGA.ClassID

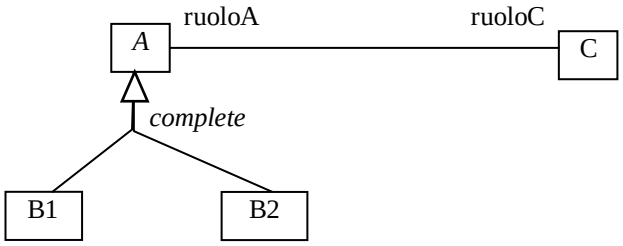
PostGIS (estratti dallo Script di Generazione)

```
-- CREATE TABLE: diga
-- Rappresenta la classe: Diga - 020501
--
CREATE TABLE diga (
     classid  varchar(70) NOT NULL,
     scril  varchar(80) NOT NULL,
     diga_class  varchar(80) NOT NULL,
     …altri attributi …
);
--
-- CREATE TABLE: diga_diga_cr
-- Rappresenta il geoattribute: Coronamento - 020501104
--
CREATE TABLE diga_diga_cr (
     classref  varchar(70) NOT NULL
);
select addgeometrycolumn ('', 'diga_diga_cr', 'diga_cr', '32632',
'POLYGON', 3);
--
-- CREATE TABLE: diga_diga_sup
-- Rappresenta il geoattribute: Sup_riferimento - 020501101
--
CREATE TABLE diga_diga_sup (
     classref  varchar(70) NOT NULL
);
select addgeometrycolumn ('', 'diga_diga_sup', 'diga_sup', '32632',
'MULTIPOLYGON', 3);
--
-- CREATE TABLE: diga_diga_si
-- Rappresenta il geoattribute: Sostegno_interno - 020501103
--
CREATE TABLE diga_diga_si (
     classref  varchar(70) NOT NULL
);
select  addgeometrycolumn  ('',  'diga_diga_si',  'diga_si',  '32632',
'POLYGON', 3);
--
-- CREATE TABLE: diga_diga_se
-- Rappresenta il geoattribute: Sostegno_esterno - 020501102
--
CREATE TABLE diga_diga_se (
     classref  varchar(70) NOT NULL
);
select  addgeometrycolumn  ('',  'diga_diga_se',  'diga_se',  '32632',
'POLYGON', 3);
```

6. MAPPING DELLE GERARCHIE

6.1 Gerarchie complete

Data una gerarchia di classi di tipo *complete* come quella della figura seguente si applica la regola di mapping di seguito descritta.



Ogni classe astratta e ogni superclasse che si collega ai figli con gerarchia *complete* non viene implementata (implementazione nei figli, perché tutte le istanze di A si ottengono dall'unione delle istanze di B1 con le istanze di B2). Quindi, con riferimento alla figura, la classe A non è implementata. Le altre classi sono implementate seguendo le regole di mapping della classe normale tenendo conto del fatto che esse devono includere, oltre ai propri attributi specifici, anche tutti gli attributi e i ruoli ereditati dagli antenati nella gerarchia; nell'esempio di figura le classi B1 e B2 ereditano gli attributi di A e in particolare anche il ruoloA che collega alla classe C.
Anche gli attributi ereditati multivalore e gli attributi ereditati di tipo DataType vengono implementati come fossero della classe, quindi con una Shapefile dedicato (se necessario).

ESEMPIO 6.1 (gerarchia *complete*)
Il Corpo Edificato (CR_EDF) è una classe astratta con due sottoclassi: edificio (EDIFC) e edificio minore (EDI_MIN). Il corpo edificato possiede 2 componenti spaziali, l'ingombro al suolo (IS) e la massima estensione (ME), che vengono ereditate da entrambe le sottoclassi. Nel seguito sono mostrati gli estratti relativi a la mapping delle 2 componenti spaziali ereditate

ShapeFlat (estratti dal Report di Mapping)

File: **EDIFC_CR_EDF_IS**

Attributo geometrico 020181101 - Ingombro al suolo - CR_EDF_IS della classe 020102 - EDIFC - Edificio

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
CR_EDF_IS	GU_CXSURFACEB3D	POLYGONZ	Attributo geometrico 020181101 - Ingombro al suolo - CR_EDF_IS della classe 020181 - Corpo edificato - CR_EDF
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020102 - EDIFC - Edificio

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

EDIFC_CR_EDF_IS.ClassREF IN EDIFC.ClassID

File: **EDIFC_CR_EDF_ME**

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
CR_EDF_ME	GU_CPSURFACE2D	POLYGON	Attributo geometrico 020181102 - Max_estensione - CR_EDF_ME della classe 020181 - Corpo edificato - CR_EDF
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020102 - EDIFC - Edificio

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

EDIFC_CR_EDF_ME.ClassREF IN EDIFC.ClassID

File: **EDI_MIN_CR_EDF_IS**

Attributo geometrico 020181101 - Ingombro al suolo - CR_EDF_IS della classe 020106 - EDI_MIN - Edificio minore

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
CR_EDF_IS	GU_CXSURFACEB3D	POLYGONZ	Attributo geometrico 020181101 - Ingombro al suolo - CR_EDF_IS della classe 020181 - Corpo edificato - CR_EDF
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020106 - EDI_MIN - Edificio minore

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

EDI_MIN_CR_EDF_IS.ClassREF IN EDI_MIN.ClassID

File: EDI_MIN_CR_EDF_ME
Attributo geometrico 020181102 - Max_estensione - CR_EDF_ME della classe 020106 - EDI_MIN - Edificio minore

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
CR_EDF_ME	GU_CPSURFACE2D	POLYGON	Attributo geometrico 020181102 - Max_estensione - CR_EDF_ME della classe 020181 - Corpo edificato - CR_EDF
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020106 - EDI_MIN -

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

EDI_MIN_CR_EDF_ME.ClassREF IN EDI_MIN.ClassID

PostGIS (estratti dallo Script di Generazione)

```
-- CREATE TABLE: edifc_cr_edf_is
-- Rappresenta il geoattribute: Ingombro al suolo - 020181101
--
CREATE TABLE edifc_cr_edf_is (
        classref  varchar(70) NOT NULL
);
select addgeometrycolumn ('', 'edifc_cr_edf_is', 'cr_edf_is', '32632', 'MULTIPOLYGON', 3);
--
-- CREATE TABLE: edifc_cr_edf_me
-- Rappresenta il geoattribute: Max_estensione - 020181102
--
CREATE TABLE edifc_cr_edf_me (
        classref  varchar(70) NOT NULL
);
select addgeometrycolumn ('', 'edifc_cr_edf_me', 'cr_edf_me', '32632', 'POLYGON', 2);
--
-- CREATE TABLE: edi_min_cr_edf_is
-- Rappresenta il geoattribute: Ingombro al suolo - 020181101
--
CREATE TABLE edi_min_cr_edf_is (
        classref  varchar(70) NOT NULL
);
select addgeometrycolumn ('', 'edi_min_cr_edf_is', 'cr_edf_is', '32632', 'MULTIPOLYGON', 3);
--
-- CREATE TABLE: edi_min_cr_edf_me
-- Rappresenta il geoattribute: Max_estensione - 020181102
--
CREATE TABLE edi_min_cr_edf_me (
        classref  varchar(70) NOT NULL
);
select addgeometrycolumn ('', 'edi_min_cr_edf_me', 'cr_edf_me', '32632', 'POLYGON', 2);
```

7. MAPPING DEGLI ATTRIBUTI A TRATTI E A SOTTOAREE

Questi attributi richiedono uno Shape/Table aggiuntivo per la memorizzazione della geometria dei tratti o delle sottoaree e dei relativi valori descrittivi.
Le regole si basano sul concetto di tratti e sottoaree minime, ossia si memorizzano le singole porzioni della geometria di una componente spaziale che condividono gli stessi valori di tutti gli attributi a tratti o sottoaree definiti sulla stessa componente spaziale.

7.1 Attributi a tratti

I tratti minimi definiti su uno stesso attributo geometrico sono memorizzati in uno Shape/Table di tipo Polyline/Linestring(x,y) (se l'attributo geometrico è di tipo curve 2D) o PolylineZ/Linestring(x,y,z) (se l'attributo geometrico è di tipo curve 3D) col vincolo "1 part" per ogni record nel caso di Shapefile. Il nome dello Shape/Table è dato dalla concatenazione <codice alfanumerico classe>_<codice alfanumerico attributo geometrico>**_SG**.
Lo Shape/Table contiene i seguenti attributi:

- **SegmentID**: identificatore univoco della combinazione di valori degli attributi a tratti,
- **ClassREF**: contiene l'identificatore ClassID dell'oggetto della classe a cui fanno riferimento gli attributi a tratti, ed è quindi soggetto a un vincolo di integrità referenziale verso tale attributo
- tutti gli attributi a tratti definiti sull'attributo geometrico; il nome di questi attributi è composto dal codice alfanumerico associato ai corrispondenti attributi a tratti.

L'unione dei tratti definiti sulla geometria di una componente spaziale deve equivalere alla geometria della componente spaziale stessa, pertanto nel caso in cui gli attributi a tratti siano tutti opzionali si devono generare anche i tratti con valore NULL in tutti gli attributi.

ESEMPIO 7.1
Consideriamo gli attributi a tratti definiti sul tracciato di un elemento di viabilità mista secondaria.

ShapeFlat (estratti dal Report di Mapping)

File:	**EL_VMS** **Classe 010116 - Elemento viabilita' mista secondaria - EL_VMS**		
NOME	**TIPO GEOUML**	**TIPO**	**DESCRIZIONE**
ClassID	Stringa (70)	Stringa (70)	Identificativo univoco
EL_VMS_TRA	GU_CPCURVE3D	ARCZ	Attributo geometrico 010116101 - Tracciato - EL_VMS_TRA della classe 010116 - Elemento viabilita' mista secondaria - EL_VMS
EL_VMS_TY	Stringa (80)	Stringa (80)	Attributo enumerato gerarchico 01011601 - tipo - EL_VMS_TY

File: **EL_VMS_EL_VMS_TRA_SG**

Tratti minimi dell'attributo geometrico 010116101 - Tracciato - EL_VMS_TRA della classe 010116 - EL_VMS - Elemento viabilita' mista secondaria

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 010116 - EL_VMS - Elemento viabilita' mista secondaria
EL_VMS_LIV	Stringa (80)	Stringa (80)	Attributo enumerato Tratto 01011603 - Livello - EL_VMS_LIV dell'attributo geometrico 010116101 - Tracciato - EL_VMS_TRA della classe 010116 - Elemento viabilita' mista secondaria - EL_VMS
EL_VMS_SED	Stringa (80)	Stringa (80)	Attributo enumerato Tratto 01011602 - Sede - EL_VMS_SED dell'attributo geometrico 010116101 - Tracciato - EL_VMS_TRA della classe 010116 - Elemento viabilita' mista secondaria - EL_VMS
SegmentID	Stringa (70)	Stringa (70)	Identificativo univoco
geometry	GU_CPCURVE3D	ARCZ	Geometria

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

EL_VMS_EL_VMS_TRA_SG.ClassREF IN EL_VMS.ClassID

PostGIS (estratti dallo Script di Generazione)

```
--
-- CREATE TABLE: el_vms
-- Rappresenta la classe: Elemento viabilita' mista secondaria - 010116
--
CREATE TABLE el_vms (
     classid  varchar(70) NOT NULL,
     el_vms_ty  varchar(80) NOT NULL
);
select addgeometrycolumn ('', 'el_vms', 'el_vms_tra', '32632',
'LINESTRING', 3);
--
-- CREATE TABLE: el_vms_el_vms_tra_sg
-- Rappresenta gli associati alla classe: Elemento viabilita' mista
secondaria - 010116
--
CREATE TABLE el_vms_el_vms_tra_sg (
     segmentid  varchar(70) NOT NULL,
     classref  varchar(70) NOT NULL,
     el_vms_liv  varchar(80) NOT NULL,
     el_vms_sed  varchar(80) NOT NULL
);
select addgeometrycolumn ('', 'el_vms_el_vms_tra_sg', 'geometry',
'32632', 'LINESTRING', 3);
```

7.2 Attributi a sottoaree

Le sottoaree minime definite su uno stesso attributo geometrico sono memorizzate in uno Shape/Table di tipo Polygon/Polygon(x,y) col vincolo "1 part" per ogni record nel caso degli Shapefile. Il nome dello Shape/Table è dato dalla concatenazione <codice alfanumerico classe>_<codice alfanumerico attributo geometrico>**_SR**.

Il File DBF dello shape contiene i seguenti attributi:

- **SubRegID**: identificatore univoco della combinazione di valori degli attributi a sottoaree,
- **ClassREF**: contiene l'identificatore ClassID dell'oggetto della classe al quale fanno riferimento gli attributi a sottoaree,
- tutti gli attributi a sottoaree definiti sull'attributo geometrico; il nome di questi attributi è composto dal codice alfanumerico associato ai corrispondenti attributi a sottoaree.

Analogamente ai tratti, l'unione delle sottoaree definite sulla geometria di una componente spaziale deve equivalere alla geometria della componente spaziale stessa, pertanto nel caso in cui gli attributi a sottoarea siano tutti opzionali si devono generare anche le sottoaree con valore NULL in tutti gli attributi.

Per gli attributi a sottoarea delle superfici B3D le regole di mapping sono equivalenti a quelle degli attributi a sottoaree di superfici 2D con la seguente possibile alternativa (da scegliere nella definizione della DPS): le sottoaree minime definite su una superficie B3D sono memorizzate in uno shape di tipo PolygonZ/Polygon(x,y,z). Si noti che in questo caso le sottoaree non sono descritte da reali superfici 3D, ma dagli anelli che ne costituiscono la frontiera in 3D analogamente all'implementazione della stessa superficie B3D.

ESEMPIO 7.2

Consideriamo gli attributi a sottoaree definiti sulla superficie di riferimento di una diga.

ShapeFlat (estratti dal Report di Mapping)

File:	**DIGA_DIGA_SUP_SR** **Sottoaree minime dell'attributo geometrico 020501101 - Sup_riferimento - DIGA_SUP della classe 020501 - DIGA – Diga**		
NOME	**TIPO GEOUML**	**TIPO**	**DESCRIZIONE**
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020501 - DIGA - Diga
DIGA_EX	Stringa (80)	Stringa (80)	Attributo enumerato Sottoarea 02050122 - Tipo estrusione - DIGA_EX dell'attributo geometrico 020501101 - Sup_riferimento - DIGA_SUP della classe 020501 - Diga – DIGA
DIGA_ZONA	Stringa (80)	Stringa (80)	Attributo enumerato Sottoarea 02050104 - Zona - DIGA_ZONA dell'attributo geometrico 020501101 - Sup_riferimento - DIGA_SUP della classe 020501 - Diga – DIGA
SubRegID	Stringa (70)	Stringa (70)	Identificativo univoco
geometry	GU_CPSURFACEB3D	POLYGONZ	Geometria

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

DIGA_DIGA_SUP_SR.ClassREF IN DIGA.ClassID

PostGIS (estratti dallo Script di Generazione)

```
--
-- CREATE TABLE: diga_diga_sup_sr
-- Rappresenta gli sottoaree associati alla classe: Diga - 020501
--

CREATE TABLE diga_diga_sup_sr (
     subregid  varchar(70) NOT NULL,
     classref  varchar(70) NOT NULL,
     diga_ex  varchar(80) NOT NULL,
     diga_zona  varchar(80) NOT NULL
);
select addgeometrycolumn ('', 'diga_diga_sup_sr', 'geometry', '32632',
'POLYGON', 3);
```

7.3 Attributi a tratti sul contorno

Il mapping è equivalente a quello degli attributi a tratti definiti su una componente spaziale lineare 2D. Se la superficie è di tipo Surface B3D il mapping è equivalente a quello degli attributi a tratti definiti su una componente spaziale lineare 3D.

ESEMPIO 7.3

Consideriamo gli attributi a tratti definiti sul contorno della componente spaziale estensione di un comune.

ShapeFlat (estratti dal Report di Mapping)

File: **COMUNE_COMUNE_EXT_SG**

Tratti minimi dell'attributo geometrico 090101102 - Estensione - COMUNE_EXT della classe 090101 - COMUNE - Comune

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
COMUNE_TLI	Stringa (50)	Stringa (50)	Attributo semplice Tratto 09010105 - Tipo confine - COMUNE_TLI dell'attributo geometrico 090101102 - Estensione - COMUNE_EXT della classe 090101 - Comune - COMUNE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 090101 - COMUNE - Comune
SegmentID	Stringa (70)	Stringa (70)	Identificativo univoco
geometry	GU_CPCURVE2D	ARC	Geometria

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

COMUNE_COMUNE_EXT_SG.ClassREF COMUNE.ClassID

PostGIS (estratti dallo Script di Generazione)

```
--
-- CREATE TABLE: comune_comune_ext_sg
-- Rappresenta gli associati alla classe: Comune - 090101
--

CREATE TABLE comune_comune_ext_sg (
      segmentid  varchar(70) NOT NULL,
      classref  varchar(70) NOT NULL,
      comune_tli  varchar(50) NOT NULL
);
select addgeometrycolumn ('', 'comune_comune_ext_sg', 'geometry',
'32632', 'LINESTRING', 2);
```

8. MAPPING DELLE SUPERFICI COLLASSATE

Se una classe ha un attributo geometrico areale collassabile, viene trattata come una classe multigeometria con tante componenti spaziali quanti sono i tipi di geometrie necessari. La componente spaziale principale, poligonale, ha il codice alfanumerico definito nella classe; quelle aggiuntive, hanno il codice alfanumerico definito nella classe seguito da un suffisso basato sul tipo (_L, _P).

ESEMPIO 8.1
Consideriamo la superficie di riferimento di una diga, definita collassabile a linea.

ShapeFlat (estratti dal Report di Mapping)

File: **DIGA_DIGA_SUP_L**

Geometria collassata a linea 020501101 - Sup_riferimento - DIGA_SUP della classe 020501 - DIGA - Diga

NOME	TIPO GEOUML	TIPO	DESCRIZIONE
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 020501 - DIGA - Diga
DIGA_SUP_L	GU_CXCURVE3D	ARCZ	Geometria collassamento a linea dell'attributo geometrico 020501101 - Sup_riferimento - DIGA_SUP della classe 020501 - Diga - DIGA

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

DIGA_DIGA_SUP_L.ClassREF IN DIGA.ClassID

PostGIS (estratti dallo Script di Generazione)

```
--
-- CREATE TABLE: diga_diga_sup_l
-- Rappresenta il collassamento linea del geoattribute: Sup_riferimento
- 020501101
--

CREATE TABLE diga_diga_sup_l (
     classref  varchar(70) NOT NULL
);
select addgeometrycolumn ('', 'diga_diga_sup_l', 'diga_sup_l', '32632',
'MULTILINESTRING', 3);
```

Parte II
Varianti adottate da altri Modelli Implementativi

9. MI SQL MULTIGEOMETRIA ORACLE

Rivediamo ora tutte le regole presentate nella prima parte del documento considerando di generare il mapping verso un modello implementativo SQL multigeometria. La principale differenza di tale modello rispetto a quello monogeometria riguarda il fatto che in presenza di più attributi geometrici su una classe non si generano strutture fisiche distinte (una per ogni attributo geometrico della classe) ma nella stessa struttura fisica che contiene gli oggetti di una classe si aggiungono tutti gli attributi geometrici dichiarati sulla classe. Si ottiene quindi una struttura con più attributi geometrici (tabella multigeometria).
Esistono alcune altre differenza di dettaglio che sono dovute a motivazioni storiche, perché questo MI è stato il primo ad essere realizzato.

9.1 MAPPING DI CLASSI MULTIGEOMETRIA CON ATTRIBUTI MONOVALORE

9.1.1 Struttura fondamentale

La struttura della Table corrispondente a una classe GeoUML dotata di una o più componenti spaziali e di soli attributi monovalore è molto semplice:

- un'unica Table è sufficiente
- il nome della Table è uguale al ***codice alfanumerico*** della classe
- un attributo descrittivo nella Table per ogni attributo descrittivo della classe popolato
- un attributo geometrico per ogni componente spaziale della classe;
- aggiunta di un attributo chiamato ***UUID*** che svolge il compito di identificatore unico (nello Schema Concettuale è implicito)
- per gli attributi di tipo Datatype del GeoUML i singoli attributi del Datatype sono trasformati in attributi della Table
- gli attributi di attributi geometrici sono riassorbiti nella classe

In sostanza, in questo caso gli elementi di una struttura Table corrispondono in maniera molto semplice agli elementi della Classe GeoUML.
Sono necessarie alcune precisazioni:

- il tipo degli attributi geometrici: per Oracle il tipo da precisare per gli attributi geometrici è sempre `MDSYS.SDO_Geometry`; in tabella 9.1 riportiamo le proprietà che devono soddisfare le geometrie contenute negli attributi di tipo `MDSYS.SDO_Geometry` tenendo conto del tipo dichiarato al livello concettuale e secondo la sintassi prevista da Oracle. Per forzare il tipo si aggiunge in Oracle un indice sull'attributo che contiene la geometria, imponendo all'indice di lavorare solo su un certo tipo di geometrie
- il tipo degli attributi descrittivi è derivato dal tipo GeoUML dei corrispondenti tipi degli attributi descrittivi della classe secondo le regole di Tabella 2.2.
- il tipo dell'attributo UUID è `VARCHAR2(70)`
- per gli attributi enumerati e gerarchici si applicano alcune regole aggiuntive viste per il mapping SQL monogeometria.

GeoUML geometry types	Gtype	Oracle Etype	Etype Interpretation	Vincoli
GU_Point2D	2001	1	1	
GU_CXPoint2D	2005	1	n>1	
GU_CPCurve2D	2002	2	1	
GU_CPSimpleCurve2D	2002	2	1	isSimple
GU_CPRing2D	2002	2	1	isClosed
GU_CXCurve2D	2006	2	1	
GU_CNCurve2D	2006	2	1	isConnected
GU_CXRing2D	2006	2	1	isClosed
GU_CPSurface2D	2003	1003/2003	1	
GU_CXSurface2D	2007	1003/2003	1	
GU_Aggregate2D	2004	1	1/n	
		2,3	1	
GU_Point3D	3001	1	1	
GU_CXPoint3D	3005	1	n	
GU_CPCurve3D	3002	2	1	
GU_CPSimpleCurve3D	3002	2	1	isSimple
GU_CPRing3D	3002	2	1	isClosed
GU_CXCurve3D	3006	2	1	
GU_CNCurve3D	3006	2	1	connect
GU_CXRing3D	3006	2	1	isClosed
GU_CPSurfaceB3D	3003	1003/2003	1	
GU_CXSurfaceB3D	3006	1003/2003	1	
GU_Aggregate3D	3004	1	1/n	
		2,3	1	

Tabella 9.1

Se consideriamo l'esempio 2.1 e supponiamo di generare il mapping verso Oracle otteniamo il seguente script di generazione SQL:

```
--
-- CREATE TABLE: ALBERO
-- Rappresenta la classe: Albero isolato - 060403
--

CREATE TABLE ALBERO (
       UUID  VARCHAR2(70) NOT NULL,
       SCRIL  VARCHAR2(80) NOT NULL,
       ALBERO_TY  VARCHAR2(80),
       ALBERO_POS MDSYS.SDO_Geometry NOT NULL
);
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ( 'ALBERO', 'ALBERO_POS',
       MDSYS.SDO_DIM_ARRAY(
       MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1000000.0 , 0.0),
       MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1000000.0 , 0.0),
       MDSYS.SDO_DIM_ELEMENT('Z', 0.0 , 10000.0 , 0.0)),
 NULL
);
```

In aggiunta si precisa: (i) la chiave primaria della tabella di classe, (ii) un indice per ogni attributo della tabella e (iii) si genera un indice spaziale per ogni attributo geometrico aggiunto alla tabella.

```
--
-- TABLE CONSTRAINT: ALBERO
-- PRIMARY KEY per la tabella della classe : Albero isolato - 060403
--
ALTER TABLE ALBERO ADD CONSTRAINT ALBERO_PK PRIMARY KEY (UUID);
--
-- TABLE INDEX: ALBERO
-- INDEX per la tabella della classe : Albero isolato - 060403
--
CREATE INDEX ALBERO_IN1 ON ALBERO(UUID);
CREATE INDEX ALBERO_IN2 ON ALBERO(SCRIL);
CREATE INDEX ALBERO_IN3 ON ALBERO(ALBERO_TY);
--
-- SPATIAL INDEX: ALBERO
-- SPATIAL INDEX per la tabella della classe : Albero isolato - 060403
--
CREATE INDEX ALBERO_SI1 ON ALBERO(ALBERO_POS) INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS('LAYER_GTYPE=POINT');
```

9.1.2 Domini enumerati e gerarchici

Per i domini enumerati ed enumerati gerarchici la regola di mapping presentata in sezione 2.2 non subisce variazioni. Quindi per l'esempio 2.1 lo strumento GeoUML Catalogue produce il seguente script SQL:

```
--
-- CREATE TABLE: D_ALBERO_TIPO
-- Rappresenta la tabella di trascodifica del dominio: Tipo - 0604030100
--
CREATE TABLE D_ALBERO_TIPO (
       CODE  VARCHAR2(80) NOT NULL,
       NAME  VARCHAR2(160),
       DEFINITION  VARCHAR2(1200),
       ALPHACODE  VARCHAR2(80)
);
--
-- DOMAIN CONSTRAINT: D_ALBERO_TIPO
-- PRIMARY KEY per la tabella di trascodifica del dominio : Tipo - 0604030100
--
ALTER TABLE D_ALBERO_TIPO ADD CONSTRAINT D_ALBERO_TIPO_PK PRIMARY KEY (CODE);
--
-- TABLE CONSTRAINT: ALBERO
-- FOREIGN KEY per la tabella della classe : Albero isolato - 060403
--
ALTER TABLE ALBERO ADD CONSTRAINT ALBERO_FK1 FOREIGN KEY (ALBERO_TY) REFERENCES
D_ALBERO_TIPO (CODE) ON DELETE CASCADE;
--
-- INSERT INTO TABLE: D_ALBERO_TIPO
-- Valori di trascodifica per il dominio: Tipo - 0604030100
--
INSERT INTO D_ALBERO_TIPO (NAME,CODE,ALPHACODE,DEFINITION) VALUES
('altro','95',null,
         'Valore assunto dall'istanza ma non previsto dalla specifica.');
INSERT INTO D_ALBERO_TIPO (NAME,CODE,ALPHACODE,DEFINITION) VALUES
('monumentale','01',null,null);
```

9.2. MAPPING DEGLI ATTRIBUTI MULTIVALORE E DEI DATATYPE

9.2.1 Attributi Multivalore

Per gli attributi multivalore la regola di mapping presentata in sezione 3.1 non subisce variazioni sostanziali, l'unica differenza sta nel nome dell'attributo che rappresenta la chiave esportata della tabella di classe, che in questo caso prende semplicemente il nome di UUID. Per l'esempio 3.1 il GeoUML Catalogue genera, per il caso Oracle, il seguente script SQL:

```
--
-- CREATE TABLE: GALLER_GALLER_USO
-- Rappresenta la tabella di appoggio per l'attributo multivalore: uso -
02030302
--
CREATE TABLE GALLER_GALLER_USO (
       UUID  VARCHAR2(70) NOT NULL,
       GALLER_USO  VARCHAR2(80) NOT NULL
);
-- TABLE CONSTRAINT: GALLER_GALLER_USO
-- FOREIGN KEY per la tabella dell'attributo multivalore : uso - 02030302
--
ALTER TABLE GALLER_GALLER_USO ADD CONSTRAINT GALLER_GALLER_USO_FK1 FOREIGN KEY
(UUID) REFERENCES  GALLER (UUID) ON DELETE CASCADE;
ALTER TABLE GALLER_GALLER_USO ADD CONSTRAINT GALLER_GALLER_USO_FK2 FOREIGN KEY
(GALLER_USO) REFERENCES  D_GALLER_USO (CODE) ON DELETE CASCADE;
ALTER TABLE GALLER_GALLER_USO ADD CONSTRAINT GALLER_GALLER_USO_PK PRIMARY KEY
(UUID ,GALLER_USO );
--
-- TABLE INDEX: GALLER_GALLER_USO
-- INDEX per la tabella dell'attributo multivalore : uso - 02030302
--
CREATE INDEX GALLER_GALLER_USO_IN1 ON GALLER_GALLER_USO(UUID);
CREATE INDEX GALLER_GALLER_USO_IN2 ON GALLER_GALLER_USO(GALLER_USO);
```

9.2.2 Datatype Multivalore

Per i datatype multivalore la regola di mapping presentata in sezione 3.1 non subisce variazioni. Per l'esempio 3.2 il GeoUML Catalogue genera, per il caso Oracle, il seguente script SQL:

```
--
-- CREATE TABLE: COMUNE_COMUNE_NOM_T
-- Rappresenta il datatype: Multilinguismo - 80
--
CREATE TABLE COMUNE_COMUNE_NOM_T (
       UUID  VARCHAR2(70),
       COMUNE_NOM_LINGUA  VARCHAR2(80),
       COMUNE_NOM_NOME  VARCHAR2(100)
);
--
-- TABLE CONSTRAINT: COMUNE_COMUNE_NOM_T
-- PRIMARY KEY e FOREIGN KEY per la tabella dell'attributo: Multilinguismo - 80
--
ALTER TABLE COMUNE_COMUNE_NOM_T ADD CONSTRAINT COMUNE_COMUNE_NOM_T_FK1 FOREIGN
KEY (UUID) REFERENCES  COMUNE (UUID) ON DELETE CASCADE;
ALTER TABLE COMUNE_COMUNE_NOM_T ADD CONSTRAINT COMUNE_COMUNE_NOM_T_PK PRIMARY
KEY (CLASSREF ,COMUNE_NOM_LINGUA ,COMUNE_NOM_NOME );
--
-- TABLE INDEX: COMUNE_COMUNE_NOM_T
-- INDEX per la tabella dell'attributo: Multilinguismo - 80
--
```

```
CREATE INDEX COMUNE_COMUNE_NOM_T_IN1 ON COMUNE_COMUNE_NOM_T(CLASSREF);
```

9.3 MAPPING DELLE ASSOCIAZIONI

9.3.1 Associazioni 1xN

Per le associazioni 1xN la regola di mapping presentata in sezione 4.1 non subisce variazioni, tranne che per il nome dell'attributo che rappresenta il ruolo, che si ottiene concatenando il prefisso "UUID_" al nome del ruolo. Per l'esempio 4.1 il GeoUML Catalogue genera, per il caso Oracle, il seguente script SQL:

```
--
-- CREATE TABLE: COMUNE
-- Rappresenta la classe: Comune - 090101
--
CREATE TABLE COMUNE (
       UUID  VARCHAR2(70) NOT NULL,
       COMUNE_IST  VARCHAR2(16) NOT NULL,
       UUID_PVDICM  VARCHAR2(70) NOT NULL,
       COMUNE_EXT MDSYS.SDO_Geometry NOT NULL,
       COMUNE_SED MDSYS.SDO_Geometry NOT NULL
);
--
-- TABLE CONSTRAINT: COMUNE
-- FOREIGN KEY per la tabella della classe : Comune - 090101
--
ALTER TABLE COMUNE ADD CONSTRAINT COMUNE_FK1 FOREIGN KEY (UUID_PVDICM)
                                  REFERENCES  PROVIN (UUID) ON DELETE CASCADE;
```

9.3.2 Associazione NxN

Per l'associazione NxN si segue la regola di mapping 4.2.

9.4 MAPPING DELLE GERARCHIE

9.4.1 Gerarchie complete

Anche in questo caso vale la regola di mapping 5.1. Vediamo come l'esempio 5.1 viene mappato su Oracle mostrando lo script SQL che il GeoUML Catalogue genera.

```
--
-- CREATE TABLE: EDIFC
-- Rappresenta la classe: Edificio - 020102
--
CREATE TABLE EDIFC (
       UUID  VARCHAR2(70) NOT NULL,
       SCRIL  VARCHAR2(80),
       EDIFC_MON  CHAR(1)  NOT NULL,
       EDIFC_SOT  VARCHAR2(80) NOT NULL,
       EDIFC_STAT  VARCHAR2(80) NOT NULL,
       EDIFC_TY  VARCHAR2(80) NOT NULL,
       CR_EDF_IS MDSYS.SDO_Geometry NOT NULL,
       CR_EDF_ME MDSYS.SDO_Geometry NOT NULL
);
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('EDIFC', 'CR_EDF_IS',
 MDSYS.SDO_DIM_ARRAY( MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
       MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001),
       MDSYS.SDO_DIM_ELEMENT('Z', 0.0 , 1.0E7 , 0.001)), NULL
);
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('EDIFC', 'CR_EDF_ME',
 MDSYS.SDO_DIM_ARRAY( MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
       MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001)), NULL
);
-- TABLE CONSTRAINT: EDIFC
-- PRIMARY KEY per la tabella della classe : Edificio - 020102
--
ALTER TABLE EDIFC ADD CONSTRAINT EDIFC_PK PRIMARY KEY (UUID);
--
-- SPATIAL INDEX: EDIFC
-- SPATIAL INDEX per la tabella della classe : Edificio - 020102
--
CREATE INDEX EDIFC_SI1 ON EDIFC(CR_EDF_IS) INDEXTYPE IS
       MDSYS.SPATIAL_INDEX PARAMETERS('LAYER_GTYPE=MULTIPOLYGON');
CREATE INDEX EDIFC_SI2 ON EDIFC(CR_EDF_ME) INDEXTYPE IS
       MDSYS.SPATIAL_INDEX PARAMETERS('LAYER_GTYPE=POLYGON');
--
-- CREATE TABLE: EDI_MIN
-- Rappresenta la classe: Edificio minore - 020106
--
CREATE TABLE EDI_MIN (
       UUID  VARCHAR2(70) NOT NULL,
       SCRIL  VARCHAR2(80),
       EDI_MIN_PR  CHAR(1)  NOT NULL,
       EDI_MIN_ST  VARCHAR2(80) NOT NULL,
       EDI_MIN_TY  VARCHAR2(80) NOT NULL,
       CR_EDF_IS MDSYS.SDO_Geometry NOT NULL,
       CR_EDF_ME MDSYS.SDO_Geometry NOT NULL
);
```

```
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('EDI_MIN', 'CR_EDF_IS',
 MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
       MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001),
       MDSYS.SDO_DIM_ELEMENT('Z', 0.0 , 1.0E7 , 0.001)), NULL
);
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('EDI_MIN', 'CR_EDF_ME',
 MDSYS.SDO_DIM_ARRAY(MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
       MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001)), NULL
);

--
-- TABLE CONSTRAINT: EDI_MIN
-- PRIMARY KEY per la tabella della classe : Edificio minore - 020106
--
ALTER TABLE EDI_MIN ADD CONSTRAINT EDI_MIN_PK PRIMARY KEY (UUID);

--
-- SPATIAL INDEX: EDI_MIN
-- SPATIAL INDEX per la tabella della classe : Edificio minore - 020106
--
CREATE INDEX EDI_MIN_SI1 ON EDI_MIN(CR_EDF_IS) INDEXTYPE IS
       MDSYS.SPATIAL_INDEX PARAMETERS('LAYER_GTYPE=MULTIPOLYGON');
CREATE INDEX EDI_MIN_SI2 ON EDI_MIN(CR_EDF_ME) INDEXTYPE IS
       MDSYS.SPATIAL_INDEX PARAMETERS('LAYER_GTYPE=POLYGON');
```

9.5 MAPPING DEGLI ATTRIBUTI A TRATTI E A SOTTOAREE

Questi attributi richiedono una Table aggiuntiva per la memorizzazione della geometria dei tratti o delle sottoaree e dei relativi valori descrittivi.

9.5.1 Attributi a tratti

I tratti minimi definiti su uno stesso attributo geometrico sono memorizzati in una Table di tipo CURVE (con la definizione di due/tre range di coordinate a seconda che l'attributo geometrico si di tipo 2D o 3D). Il nome della Table è dato dalla concatenazione <codice alfanumerico classe>_<codice alfanumerico attributo geometrico>**_SG**.
La Table contiene i seguenti attributi:

- **UUID**: identificatore univoco della combinazione di valori degli attributi a tratti,
- **UUID_**<codice alfanumerico classe>: contiene l'identificatore UUID dell'oggetto della classe a cui fanno riferimento gli attributi a tratti, ed è quindi soggetto a un vincolo di integrità referenziale verso tale attributo,
- tutti gli attributi a tratti definiti sull'attributo geometrico; il nome di questi attributi è composto dal codice alfanumerico associato ai corrispondenti attributi a tratti.

Vediamo come l'esempio 7.1 viene mappato su Oracle mostrando lo script SQL che il GeoUML Catalogue genera.

```
--
-- CREATE TABLE: EL_VMS
-- Rappresenta la classe: Elemento viabilità mista secondaria - 010116
--
CREATE TABLE EL_VMS (
       UUID  VARCHAR2(70) NOT NULL,
       EL_VMS_TY  VARCHAR2(80) NOT NULL,
       EL_VMS_TRA MDSYS.SDO_Geometry NOT NULL
);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('EL_VMS','EL_VMS_TRA',MDSYS.SDO_DIM_ARRAY(
            MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Z', 0.0 , 1.0E7 , 0.001)
 ), NULL );

--
-- CREATE TABLE: EL_VMS_EL_VMS_TRA_SG
-- Rappresenta i tratti di Elemento viabilità mista secondaria - 010116
--
CREATE TABLE EL_VMS_EL_VMS_TRA_SG (
       UUID  VARCHAR2(70) NOT NULL,
       UUID_EL_VMS  VARCHAR2(70) NOT NULL,
       EL_VMS_LIV  VARCHAR2(80) NOT NULL,
       EL_VMS_SED  VARCHAR2(80) NOT NULL,
       GEOMETRY MDSYS.SDO_Geometry NOT NULL
);
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('EL_VMS_EL_VMS_TRA_SG','GEOMETRY',MDSYS.SDO_DIM_ARRAY(
            MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Z', 0.0 , 1.0E7 , 0.001)
 ), NULL );
```

9.5.2 Attributi a sottoaree

Le sottoaree minime definite su uno stesso attributo geometrico sono memorizzate in una Table di tipo POLYGON (con la definizione di due/tre range di coordinate a seconda che l'attributo geometrico si di tipo 2D o 3D). Il nome della Table è dato dalla concatenazione <codice alfanumerico classe>_<codice alfanumerico attributo geometrico>**_SR**.
La Table contiene i seguenti attributi:

- **UUID**: identificatore univoco della combinazione di valori degli attributi a sottoaree,
- **UUID_**<codice alfanumerico classe>: contiene l'identificatore UUID dell'oggetto della classe a cui fanno riferimento gli attributi a sottoaree, ed è quindi soggetto a un vincolo di integrità referenziale verso tale attributo,
- tutti gli attributi a sottoaree definiti sull'attributo geometrico; il nome di questi attributi è composto dal codice alfanumerico associato ai corrispondenti attributi a sottoaree.

Valgono poi tutte le osservazioni e i vincoli descritti nella sezione 7.2. Vediamo come l'esempio 7.2 viene mappato su Oracle mostrando lo script SQL che il GeoUML Catalogue genera.

```
--
-- CREATE TABLE: DIGA
-- Rappresenta la classe: Diga - 020501
--
CREATE TABLE DIGA (
       UUID  VARCHAR2(70) NOT NULL,
       SCRIL  VARCHAR2(80),
       DIGA_CLASS  VARCHAR2(80) NOT NULL,
       DIGA_CR_DIGA_CR_EX  VARCHAR2(80) NOT NULL,
       DIGA_CR_DIGA_CR_QE  DOUBLE PRECISION NOT NULL,
       DIGA_CT  VARCHAR2(80) NOT NULL,
       DIGA_SE_DIGA_SE_EX  VARCHAR2(80) NOT NULL,
       DIGA_SE_DIGA_SE_QE  DOUBLE PRECISION NOT NULL,
       DIGA_SI_DIGA_SI_EX  VARCHAR2(80) NOT NULL,
       DIGA_SI_DIGA_SI_QE  DOUBLE PRECISION NOT NULL,
       DIGA_TY  VARCHAR2(80) NOT NULL,
       DIGA_CR MDSYS.SDO_Geometry,
       DIGA_SE MDSYS.SDO_Geometry NOT NULL,
       DIGA_SI MDSYS.SDO_Geometry,
       DIGA_SUP MDSYS.SDO_Geometry,
       DIGA_SUP_C MDSYS.SDO_Geometry
);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('DIGA','DIGA_CR', MDSYS.SDO_DIM_ARRAY(
            MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Z', 0.0 , 1.0E7 , 0.001)
 ), NULL);

INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('DIGA', 'DIGA_SE', …


--
-- CREATE TABLE: DIGA_DIGA_SUP_SR
-- Rappresenta le sottoaree associate alla classe: Diga - 020501
--
CREATE TABLE DIGA_DIGA_SUP_SR (
       UUID  VARCHAR2(70) NOT NULL,
       UUID_DIGA  VARCHAR2(70) NOT NULL,
       DIGA_EX  VARCHAR2(80) NOT NULL,
       DIGA_ZONA  VARCHAR2(80) NOT NULL,
       GEOMETRY MDSYS.SDO_Geometry NOT NULL
);
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID) VALUES ('DIGA_DIGA_SUP_SR','GEOMETRY',MDSYS.SDO_DIM_ARRAY(
            MDSYS.SDO_DIM_ELEMENT('X', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Y', 0.0 , 1.0E7 , 0.001),
            MDSYS.SDO_DIM_ELEMENT('Z', 0.0 , 1.0E7 , 0.001)
 ), NULL);
```

9.6 Geometrie collassate

Per le geometrie collassate il MI SQL multi geometria prevede di generare più attributi geometrici nella tabella della classe, uno per ogni tipo di collassa mento dichiarato nello schema. Ad esempio, la classe Diga mostrata nell'esempio precedente contiene una componente geometrica "Sup_riferimento" è collassabile a linea e pertanto nella tabella che implementa la classe troviamo due attributi "DIGA_SUP" e "DIGA_SUP_C" come di seguito riportato.

```
--
-- CREATE TABLE: DIGA
-- Rappresenta la classe: Diga - 020501
--
CREATE TABLE DIGA (
       UUID  VARCHAR2(70) NOT NULL,
       SCRIL  VARCHAR2(80),
       DIGA_CLASS  VARCHAR2(80) NOT NULL,
       DIGA_CR_DIGA_CR_EX  VARCHAR2(80) NOT NULL,
       DIGA_CR_DIGA_CR_QE  DOUBLE PRECISION NOT NULL,
       DIGA_CT  VARCHAR2(80) NOT NULL,
       DIGA_SE_DIGA_SE_EX  VARCHAR2(80) NOT NULL,
       DIGA_SE_DIGA_SE_QE  DOUBLE PRECISION NOT NULL,
       DIGA_SI_DIGA_SI_EX  VARCHAR2(80) NOT NULL,
       DIGA_SI_DIGA_SI_QE  DOUBLE PRECISION NOT NULL,
       DIGA_TY  VARCHAR2(80) NOT NULL,
       DIGA_CR MDSYS.SDO_Geometry,
       DIGA_SE MDSYS.SDO_Geometry NOT NULL,
       DIGA_SI MDSYS.SDO_Geometry,
       DIGA_SUP MDSYS.SDO_Geometry,
       DIGA_SUP_C MDSYS.SDO_Geometry
);
```

10. MI TOPOLOGICO SHAPE_TOPO

Rivediamo ora tutte le regole presentate nella prima parte del documento considerando di generare il mapping verso un modello implementativo topologico.
La principale differenza di tale modello rispetto a quello SHAPE_FLAT riguarda il fatto che le componenti geometriche delle classi non vengono implementate negli Shape delle classi, ma le loro istanze sono rappresentate in strutture, dette Insiemi Topologici, che raggruppano geometrie provenienti da diverse componenti geometriche. Ciò consente di precisare vincoli di integrità spaziale che devono essere validi complessivamente sull'insieme topologico.
Pertanto questo modello implementativo richiede per poter essere applicato ad una specifica concettuale la definizione di un certo numero di insiemi topologici in cui far confluire tutte le geometrie delle classi; in particolare, per ogni componente geometrica si deve indicare in quale insieme topologico si intendono rappresentare le sue istanze. Per ogni insieme topologico si generano alcuni Shape, mentre gli oggetti delle classi saranno rappresentati in file DBF.

10.1 GLI INSIEMI TOPOLOGICI

Un insieme topologico è un insieme di primitive geometriche 2D o 3D che rispettano alcune proprietà topologiche di seguito descritte le quali fondamentalmente garantiscono che ogni primitiva è semplice e non si sovrappone alle altre (al massimo è adiacente).
Un insieme topologico IT viene implementato da 4 **Shape Topologici** contenenti la rappresentazione geometrica delle **primitive topologiche** (geometrie di base dalle quali partire per comporre le geometrie delle componenti spaziali degli oggetti). Le primitive sono suddivise in **edge** (primitive lineari) e **node** (primitive puntiformi).

I 4 Shape Topologici di un insieme topologico *IT* sono:

- *IT*_**E_3D**: è uno Shape di tipo PolylineZ – 1 part (non accetta record con curve multipart) chiamato Shape degli edge nello spazio 3D.
- *IT*_**E_2D**: è uno Shape di tipo Polyline – 1 part (non accetta record con curve multipart) chiamato Shape di edge nello spazio 2D.
- *IT*_**N_3D**: è uno Shape di tipo PointZ, chiamato Shape di nodi nello spazio 3D.
- *IT*_**N_2D**: è uno Shape di tipo Point, chiamato Shape di nodi nello spazio 2D.

Le primitive contenute negli Shape sopra descritti devono soddisfare le seguenti proprietà topologiche e metriche allo scopo di garantire che tutte le intersezioni tra primitive siano pre-calcolate e che le relazioni tra primitive non vengano alterate a fronte di operazioni di trasferimento o a causa di elaborazioni eseguite su due o più geometrie.

1) **Vincoli interni ad ognuno dei 4 Shape**
   a) Ogni edge di *IT*_**E_2D** e di *IT*_**E_3D** deve essere semplice (in ogni edge non sono ammesse né autointersezioni, né autotangenze)
   b) Ogni edge di *IT*_**E_3D** non può avere segmenti verticali, ossia edge nei quali due vertici consecutivi (o un estremo e il suo vertice più vicino) differiscano solo nel valore della coordinata Z.
   c) Per ogni coppia di edge di *IT*_**E_2D** (*IT*_**E_3D**) la loro intersezione è vuota oppure deve coincidere con uno degli estremi degli edge coinvolti.
   d) Due nodi di *IT*_**N_2D** (*IT*_**N_3D**) non possono coincidere, vale a dire non ci sono nodi duplicati.

   Osservazione: si noti che dal vincolo c) deriva che non possono esistere due edge uguali o parzialmente sovrapposti (condivisione) in *IT*_**E_2D** (*IT*_**E_3D**).

2) **Vincoli di consistenza Edge/Nodi**
   Ogni nodo di *IT*_**N_2D** (*IT*_**N_3D**) è isolato (intersezione nulla con tutti gli edge di *IT*_**E_2D** (*IT*_**E_3D**)) oppure coincide con un estremo di un edge di *IT*_**E_2D** (*IT*_**E_3D**).
   Osservazione: non è accettata la coincidenza di un nodo con un vertice interno di un edge.

3) **Vincoli di consistenza tra spazio 2D e spazio 3D**
   Questi vincoli impongono che un edge (node) definito nello spazio 3D abbia un corrispondente elemento nello spazio 2D, ma viceversa ammettono edge (node) definiti nello spazio 2D che non abbiano un corrispondente elemento nello spazio 3D.

10.1.2 Casi particolari di Insiemi topologici

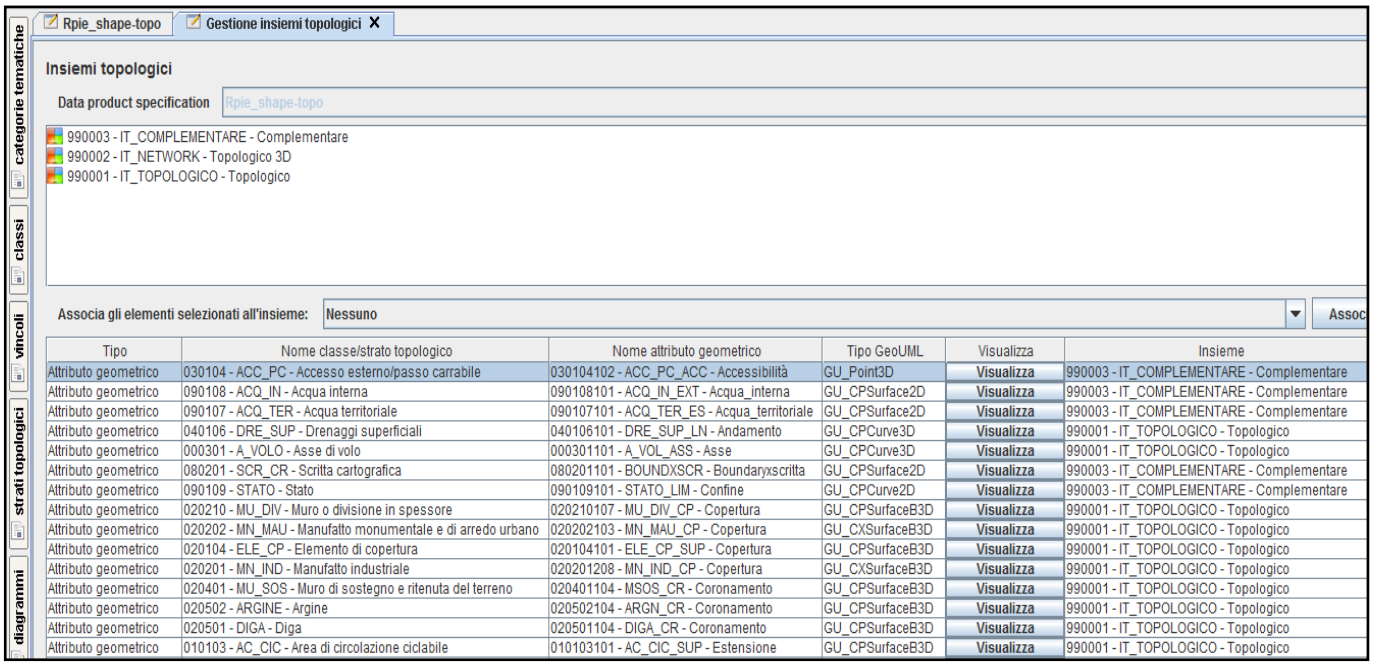
In taluni casi un insieme topologico può essere rappresentato utilizzando un sottoinsieme dei 4 Shape sopra descritti. In tal caso rimangono validi solo i vincoli applicabili agli Shape effettivamente presenti.

Geometrie mappate sull'insieme topologico	Shape necessary
Lineari e puntiformi in 2D	***IT*_E_2D e *IT*_N_2D**
Lineari in 2D	***IT*_E_2D**
Puntiformi in 2D	***IT*_N_2D**
Lineari in 3D	***IT*_E_3D e *IT*_E_2D**
Puntiformi in 3D	***IT*_N_3D e *IT*_N_2D**
Lineari in 3D con topologia solo 3D	***IT*_E_3D**
Puntiformi in 3D con topologia solo 3D	***IT*_N_3D**
Lineari e puntiformi in 3D con topologia solo 3D	***IT*_E_3D e *IT*_N_3D**

Una geometria 3D inserita in un insieme topologico implica, in generale, la generazione anche del corrispondente Shape in 2D ottenuto per proiezione. Si noti che, come previsto nelle ultime tre righe della tabella, è data la possibilità di definire anche un insieme topologico solo 3D che memorizza le geometrie 3D negli shapefile *IT*_**E_3D** e *IT*_**N_3D**, ma che non richiede la corrispondente componente planare in 2D.

Gli insiemi topologici si definiscono nell'ambito una Data Product Specification e possono quindi cambiare da una DPS all'altra anche se condividono la stessa specifica concettuale. Come esempio di insiemi topologici definiti su una specifica, si riporta di seguito uno shapshot preso dallo strumento GeoUML Catalogue che mostra la maschera di definizione di un insieme topologico e del mapping degli attributi geometrici sugli insiemi topologici.

Si noti che esistono tre tipi di Insiemi Topologici: (i) ***Topologico***: contiene geometrie 2D e 3D; (ii) ***Topologico 3D***: contiene solo geometrie 3D; (iii) ***Complementare***: contiene tutte le geometrie di componenti geometriche che non ricadono in uno specifico IT. Si noti che a quest'ultimo insieme topologico non si applicano pertanto i vincoli che caratterizzano l'insieme topologico sopra descritti.



Tipo	Nome classe/strato topologico	Nome attributo geometrico	Tipo GeoUML	Visualizza	Insieme
Attributo geometrico	030104 - ACC_PC - Accesso esterno/passo carrabile	030104102 - ACC_PC_ACC - Accessibilità	GU_Point3D	Visualizza	990003 - IT_COMPLEMENTARE - Complementare
Attributo geometrico	090108 - ACQ_IN - Acqua interna	090108101 - ACQ_IN_EXT - Acqua_interna	GU_CPSurface2D	Visualizza	990003 - IT_COMPLEMENTARE - Complementare
Attributo geometrico	090107 - ACQ_TER - Acqua territoriale	090107101 - ACQ_TER_ES - Acqua_territoriale	GU_CPSurface2D	Visualizza	990003 - IT_COMPLEMENTARE - Complementare
Attributo geometrico	040106 - DRE_SUP - Drenaggi superficiali	040106101 - DRE_SUP_LN - Andamento	GU_CPCurve3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	000301 - A_VOLO - Asse di volo	000301101 - A_VOL_ASS - Asse	GU_CPCurve3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	080201 - SCR_CR - Scritta cartografica	080201101 - BOUNDXSCR - Boundaryxscritta	GU_CPSurface2D	Visualizza	990003 - IT_COMPLEMENTARE - Complementare
Attributo geometrico	090109 - STATO - Stato	090109101 - STATO_LIM - Confine	GU_CPCurve2D	Visualizza	990003 - IT_COMPLEMENTARE - Complementare
Attributo geometrico	020210 - MU_DIV - Muro o divisione in spessore	020210107 - MU_DIV_CP - Copertura	GU_CPSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	020202 - MN_MAU - Manufatto monumentale e di arredo urbano	020202103 - MN_MAU_CP - Copertura	GU_CXSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	020104 - ELE_CP - Elemento di copertura	020104101 - ELE_CP_SUP - Copertura	GU_CPSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	020201 - MN_IND - Manufatto industriale	020201208 - MN_IND_CP - Copertura	GU_CXSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	020401 - MU_SOS - Muro di sostegno e ritenuta del terreno	020401104 - MSOS_CR - Coronamento	GU_CPSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	020502 - ARGINE - Argine	020502104 - ARGN_CR - Coronamento	GU_CPSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	020501 - DIGA - Diga	020501104 - DIGA_CR - Coronamento	GU_CPSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico
Attributo geometrico	010103 - AC_CIC - Area di circolazione ciclabile	010103101 - AC_CIC_SUP - Estensione	GU_CPSurfaceB3D	Visualizza	990001 - IT_TOPOLOGICO - Topologico

Schermata del GeoUML Catalogue: definizione di insiemi topologici e mapping delle componenti geometriche.

Quando sono stati definiti gli insiemi topologici e il mapping delle componenti geometriche delle classi di su essi, è possibile generare gli Shape per rappresentare ogni insieme topologico. Componenti spaziali di classi diverse e di diversa dimensionalità possono confluire in uno stesso insieme topologico e una classe con più componenti spaziali può memorizzarle in insiemi topologici diversi.
Data una componente spaziale (detta anche attributo geometrico) CL.AG di una classe CL esiste un preciso Shape di un insieme topologico (o complementare) destinato a contenerla; esso è determinato nel modo seguente:

1. in base al mapping esplicito dichiarato nella DPS, CL.AG → IT
2. in base alla Tabella 10.1 e al tipo della componente spaziale, CL.AG.type → Shape di IT

Il nome degli shapefile di un insieme topologico è composto da un prefisso ed un suffisso costanti concatenati al *codice* dell'insieme topologico definito nella DPS nel seguente modo:

IT_*codice*_E_3D.shp IT_*codice*_E_2D.shp IT_*codice*_N_3D.shp IT_*codice*_N_2D.shp

La struttura del DBF associato a ciascuno di questi Shape è fissa, non dipende dalla DPS, e contiene i seguenti attributi:

- **PrimID** di tipo *Number 9* (0 decimali) che contiene l'identificatore di ogni primitiva dello Shape; tale identificatore è univoco nello Shape.
- Nessun altro attributo specifico è presente negli shape di un insieme topologico.

Tipo GeoUML della componente spaziale	Shape dell'insieme topologico dove si rappresentano le geometrie della componente
GU_Point2D	*IT*_N_2D
GU_Point3D	*IT*_N_3D e *IT*_N_2D(*)
GU_CPCurve2D	*IT*_E_2D
GU_CPCurve3D	*IT*_E_3D e *IT*_E_2D(*)
GU_CPSimpleCurve2D	*IT*_E_2D
GU_CPSimpleCurve3D	*IT*_E_3D e IT_E_2D(*)
GU_CPRing2D	*IT*_E_2D
GU_CPRing3D	*IT*_E_3D e *IT*_E_2D(*)
GU_CPSurface2D	*IT*_E_2D
GU_CXPoint2D	*IT*_N_2D
GU_CXPoint3D	*IT*_N_3D e *IT*_N_2D(*)
GU_CXCurve2D	*IT*_E_2D
GU_CXCurve3D	*IT*_E_3D e *IT*_E_2D(*)
GU_CXRing2D	*IT*_E_2D
GU_CXRing3D	*IT*_E_3D e *IT*_E_2D(*)
GU_CNCurve2D	*IT*_E_2D
GU_CNCurve3D	*IT*_E_3D e *IT*_E_2D(*)
GU_CXSurface2D	*IT*_E_2D
GU_CPSurfaceB3D	*IT*_E_3D e *IT*_E_2D(*)
GU_CXSurfaceB3D	*IT*_E_3D e *IT*_E_2D(*)

TABELLA 10.1: (*) lo shape 2D è omesso se la topologia è solo 3D.

La corrispondenza tra le istanze della componente spaziale di una classe (CL.GA) e le primitive dello Shape è descritta in un file DBF aggiuntivo chiamato "**file di composizione**". I nomi dei file di composizione seguono la stessa convenzione stabilita per gli Shape: si utilizza un prefisso ed un suffisso concatenato al *codice* dell'insieme topologico, ottenendo:
IT_*codice*_E_3D_COMP, IT_*codice*_E_2D_COMP,
IT_*codice*_N_3D_COMP e **IT_*codice*_N_2D_COMP.**

I file di composizione contengono due attributi:

- **PrimID** di tipo Number 9 (0 decimale): è l'identificatore della primitiva geometrica
- **GeoID** di tipo Number 9 (0 decimale): è l'identificatore di una singola geometria (istanza di una componente spaziale associata all'insieme topologico); tale identificatore è univoco nell'intero insieme topologico e inoltre la coppia <PrimID, GeoID> è univoca all'interno di ogni file di composizione.

Data un'istanza di geometria di una componente spaziale di una classe identificata da un valore GeoID, l'insieme delle coppie {<$PrimID_{x1}$, GeoID>, <$PrimID_{x2}$, GeoID>, ..., <$PrimID_{xk}$, GeoID>} definisce l'insieme di primitive {$PrimID_{x1}$, $PrimID_{x2}$, ..., $PrimID_{xk}$} che compongono tale istanza di attributo geometrico.

10.2 MAPPING DI CLASSI MULTIGEOMETRIA CON ATTRIBUTI MONOVALORE

10.2.1 Struttura fondamentale

La struttura fisica corrispondente a una classe GeoUML dotata di una o più componente spaziale AG_1**, …,** AG_n e di soli attributi monovalore, nel modello MI Shape_Topo è molto simile a quanto si ottiene nel modello MI Shape_Flat, solo che invece di avere uno Shape per la classe abbiamo solo un DBF in quanto le geometrie istanze delle componenti spaziali sono rappresentate negli Shape degli insiemi topologici:

- un unico file DBF è sufficiente;
- il nome del file DBF è uguale al ***codice alfanumerico*** della classe;
- i nomi degli attributi coincidono con il codice alfanumerico degli attributi;
- un attributo descrittivo nel file DBF per ogni attributo descrittivo della classe popolato;
- uno attributo di tipo Number 9 (0 decimale) per ogni componente spaziale popolata della classe; tale attributo conterrà il GeoID della geometria che rappresenta l'istanza della componente geometrica (nel caso in cui l'attributo geometrico sia opzionale, la mancanza della geometria viene rappresentata dal valore -1);
- in aggiunta un attributo chiamato ***ClassID*** che svolge il compito di identificatore unico (nello Schema Concettuale è implicito);
- per gli attributi di tipo Datatype del GeoUML i singoli attributi del Datatype sono trasformati in attributi del file DBF;
- gli attributi di attributi geometrici sono riassorbiti nella classe.

ESEMPIO 10.1

Consideriamo la definizione nello Schema Concettuale (SC) di una classe semplice, la classe "**Albero isolato**" e supponiamo esistano tre insiemi topologici definiti: IT_TOPOLOGICO (COPERTURA), IT_NETWORK (RETI) e IT_COMPLEMENTARE (COMPLEMENTARE). La classe "**Albero isolato**" ha le seguenti caratteristiche:

- il codice alfanumerico della classe è **ALBERO**.
- la classe ha una componente spaziale **Posizione,** con codice alfanumerico **ALBERO_POS** di tipo **GU_Point3D** associata all'insieme topologico IT_TOPOLOGICO.
- l'unico attributo descrittivo è **ALBERO_TY** (tipo)

Si riportano di seguito gli estratti della documentazione prodotta dal Catalogue nei due MI

ShapeTopo (estratti dal Report di Mapping)

Insiemi topologici			
CODICE	**NOME**	**TIPO**	**DESCRIZIONE**
990001	IT_TOPOLOGICO	Topologico	COPERTURA
990002	IT_NETWORK	Topologico 3D	RETI
990003	IT_COMPLEMENTARE	Non topologico	COMPLEMENTARE

Elementi associati all'insieme 990001 - IT_TOPOLOGICO – Topologico			
NOME	**CODICE - CODICE ALFANUMERICO**	**TIPO GEOUML**	**TIPO DI ELEMENTO**
…	…	…	…
Posizione	060403101 - ALBERO_POS	GU_Point3D	Attributo geometrico
	si acquisisce il punto 3D in corrispondenza del piede albero		

File ALBERO Classe 060403 - Albero isolato - ALBERO			
NOME	**TIPO GEOUML**	**TIPO**	**TIPO**
ClassID	Stringa (70)	Stringa (70)	Identificativo univoco
ALBERO_POS	Intero	Intero (9)	GEOID dell'attributo geometrico 060403101 - Posizione - ALBERO_POS
ALBERO_TY	Stringa (80)	Stringa (80)	Attributo enumerato 06040301 - tipo - ALBERO_TY
ScRil	Stringa (80)	Stringa (80)	Riferimento al codice di livello di scala

Si noti che nel report di mapping non vengono descritti nel dettaglio gli Shape e i file DBF a struttura fissa generati per l'insieme topologico. In questo caso, vale a dire per rappresentare la componente geometrica "ALBERO_POS" di ALBERO si genereranno per l'insieme topologico IT_TOPOLOGICO i seguenti Shape e file DBF:

- IT_990001_N_3D.shp;
- IT_990001_N_2D.shp;
- IT_99001_N_2D.dbf (PrimID);
- IT_99001_N_3D.dbf (PrimID);
- IT_99001_N_2D_COMP.dbf (PrimID, GeoID);
- IT_99001_N_3D_COMP.dbf (PrimID, GeoID);

In realtà, poiché le componenti geometriche di altre classi verranno mappate su IT_TOPOLOGICO verranno prodotti anche gli Shape degli edge.

10.3 Mapping degli Attributi a eventi, tratti, sottoarea

La geometria degli eventi, tratti o sottoaree deve essere memorizzata nello stesso insieme topologico della relativa componente spaziale. L'implementazione adottata è quella dei tratti, eventi e sottoaree minime, ossia si identificano le porzioni della geometria di una componente spaziale che condividono gli stessi valori di tutti gli attributi a tratti, eventi o sottoarea definiti sulla componente spaziale.

NOTA: quando sono presenti gli attributi a tratti, a sottoaree o a tratti sul contorno la corrispondente componente spaziale non viene rappresentata. Vale a dire non c'è il corrispondente attributo geometrico nella tabella di classe che conterrebbe il GeoID.

10.3.1 Attributi a eventi

Gli eventi sono memorizzati nello shape dei nodi *IT*_**N_2D** (*IT*_**N_3D**) e poiché sono definiti su una componente lineare o poligonale descritta negli shapefile *IT*_**E_2D** (*IT*_**N_3D**) devono quindi coincidere con gli estremi di un edge o essere isolati (e quindi appartenenti alla parte interna di un poligono).
Ogni insieme di eventi definiti su una componente geometrica di codice *AG* appartenente ad una classe di codice *CL* produrrà un file DBF il cui nome è dato da: *CL_AG*_**E** e che sarà composto dai seguenti campi:

- **EventID**: identificatore univoco della combinazione di eventi,
- **ClassREF**: contiene l'identificatore ClassID dell'oggetto della classe a cui fanno riferimento gli attributi a eventi
- **Geometry**: identificatore della geometria dell'evento
- tutti gli attributi ad eventi definiti sulla componente geometrica AG; il nome di questi attributi coincide con il codice alfanumerico associato al corrispondente attributo a eventi.

Ad esempio, data una classe *C1* con componente geometrica *AG1* di tipo *GU_CPCurve3D* e un attributo ad eventi *A1: Integer aEventi su AG1*, e supposto che *AG* sia mappata sull'insieme topologico *X* il mapping nel modello MI Shape_Topo produce oltre alle seguenti strutture fisiche per la descrizione della classe e della componente spaziale AG1 (supponiamo che *C1*, *AG1*, *A1* siano i codici alfanumerici degli elementi da rappresentare):

- IT_*X*_E_3D.shp
- IT_*X*_E_3D.dbf (PrimID)
- IT_*X*_E_3D_COMP (PrimID, GeoID)
- *C1*(ClassID, *AG1*) dove *AG1* contiene istanze di GeoID

anche le seguenti strutture per gli eventi:

- IT_*X*_N_3D.shp
- IT_*X*_N_3D.dbf (PrimID)
- IT_*X*_N_3D_COMP (PrimID, GeoID)
- *C1_AG1*_E(EventID, ClassREF, Geometry, *A1*)

10.3.2 Attributi a tratti

I tratti sono memorizzati nello stesso Shape *IT*_**E_2D** (*IT*_**E_3D**) nel quale sono memorizzati gli edge della componente spaziale a cui si riferiscono.
Ogni insieme di attributi a tratti definiti su una componente spaziale di codice *AG* appartenente ad una classe di codice *CL* produrrà un file DBF di nome *CL_AG*_**SG.** Un record del file può contenere un singolo tratto minimo che condivide gli stessi valori per tutti gli attributi a tratti o un insieme di tratti minimi; nel primo caso la geometria associata ad ogni riga è una CPCurve*D, mentre nel secondo caso sarà un CXCurve*D. La scelta tra le due opzioni avviene nella definizione della DPS.
Il file DBF contiene i seguenti campi:

- **SegmentID**: identificatore univoco della combinazione di valori degli attributi a tratti,
- **ClassREF**: contiene l'identificatore ClassID dell'oggetto della classe a cui fanno riferimento gli attributi a tratti
- **Geometry**: identificatore della geometria del tratto o dei tratti minimi.
- tutti gli attributi a tratti definiti sull'attributo geometrico; il nome di questi attributi coincide con il codice alfanumerico del corrispondente attributo a tratti.

Si noti che, poiché l'unione della geometria tratti definiti sulla geometria di una componente spaziale produce la geometria della componente spaziale stessa, la componente spaziale derivabile non viene rappresentata come attributo nel file DBF della classe. Si ricorda che ciò implica che nel caso in cui

l'attributo a tratti sia opzionale si debbano generare obbligatoriamente anche i tratti associati al valore NULL.

Consideriamo ora l'esempio 6.1 gli attributi a tratti definiti sul tracciato di un elemento di viabilità mista secondaria e vediamo quale struttura fisica si genera applicando il MI Shape_Topo.

ShapeTopo (estratti dal Report di Mapping)

Elementi associati all'insieme	**990002 - IT_NETWORK - Topologico 3D**		
NOME	**CODICE - CODICE ALFANUMERICO**	**TIPO GEOUML**	**TIPO DI ELEMENTO**
Tracciato	010116101 - EL_VMS_TRA	GU_CPCurve3D	Attributo geometrico
	acquisizione della mezzeria dei percorsi di viabilità mista secondaria come definiti dall'attributo tipo		

File: **EL_VMS**

Classe 010116 - Elemento viabilita' mista secondaria - EL_VMS

NOME	**TIPO GEOUML**	**TIPO**	**DESCRIZIONE**
ClassID	Stringa (70)	Stringa (70)	Identificativo univoco
EL_VMS_TY	Stringa (80)	Stringa (80)	Attributo enumerato gerarchico 01011601 - tipo - EL_VMS_TY

File: **EL_VMS_EL_VMS_TRA_SG**

Tratti minimi dell'attributo geometrico 010116101 - Tracciato - EL_VMS_TRA della classe 010116 - EL_VMS - Elemento viabilita' mista secondaria

NOME	**TIPO GEOUML**	**TIPO**	**DESCRIZIONE**
ClassREF	Stringa (70)	Stringa (70)	Riferimento alla classe 010116 - EL_VMS – Elemento viabilità mista secondaria
SegmentID	Stringa (70)	Stringa (70)	Identificativo univoco
EL_VMS_LIV	Stringa (80)	Stringa (80)	Attributo enumerato Tratto 01011603 - Livello - EL_VMS_LIV dell'attributo geometrico 010116101 - Tracciato - EL_VMS_TRA della classe 010116 - Elemento viabilita' mista secondaria - EL_VMS
EL_VMS_SED	Stringa (80)	Stringa (80)	Attributo enumerato Tratto 01011602 - Sede - EL_VMS_SED dell'attributo geometrico 010116101 - Tracciato - EL_VMS_TRA della classe 010116 - Elemento viabilita' mista secondaria - EL_VMS
geometry	Intero	Intero (9)	GEOID dei tratti minimi dell'attributo geometrico 010116101 - Tracciato - EL_VMS_TRA

ClassREF deve soddisfare il vincolo di integrità referenziale seguente:

EL_VMS_EL_VMS_TRA_SG.ClassREF IN EL_VMS.ClassID

Le strutture fisiche che si generano sono pertanto le seguenti:

- IT_*990002*_E_3D.shp
- IT_*990002*_E_3D.dbf (PrimID)
- IT_*990002*_E_3D_COMP (PrimID, GeoID)

- EL_VMS(ClassID, EL_VMS_TY)
- EL_VMS_EL_VMS_TRA_SG(SegmentID, ClassREF, EL_VMS_LIV, EL_VMS_SED, Geometry)

Si noti che:

- Anche quando gli attributi a tratti sono definiti sul contorno 2D di una superficie 2D, l'attributo geometrico della classe (anche se poligonale) non viene generato ma risulta derivabile dai tratti che rappresentano la frontiera esterna ed eventualmente le frontiere interne della superficie
- Lo stesso vale per il caso in cui gli attributi a tratti siano definiti sul contorno 3D di una superficie B3D
- Se invece gli attributi a tratti sono definiti sul contorno 2D di una superficie B3D la derivabilità delle frontiere della superficie dai tratti viene meno e quindi viene rappresentata esplicitamente la superficie con un attributo nel DBF della classe

10.3.3 Attributi a sottoaree

Si implementano in modo del tutto analogo agli attributi a tratti, il suffisso sul nome del file DBF in questo caso è "_SR" invece di "_SG". Inoltre l'attributo nel DBF che rappresenta l'identificatore della sottoarea prende il nome di "SubRegID". Anche per le sottoaree si applica la derivabilità della componente sulla quale sono definite tranne nel caso in cui si richiedano sottoaree 2D su superfici B3D.

10.4 Componenti spaziali di tipo aggregati generici (GU_Aggregate2D o GU_Aggregate3D)

Per ogni componente spaziale di tipo aggregato le primitive componenti saranno suddivise tra gli shapefile 2D o 3D di un insieme topologico in base alla dimensione dell'aggregato. Viene inoltre definito un ulteriore file .dbf che permette la descrizione dei componenti di un aggregato chiamato codice alfanumerico classe_codice alfanumerico attributo geometrico_AG. Si omette la componente spaziale corrispondente nel file .dbf della classe.

Data la classe C(…, AG1:GU_Aggregate2D,…) si generano i seguenti file:

IT_codice_E_2D (…, PrimID)

IT_codice_E_2D_COMP (PrimID, GeoID)

IT_codice_N_2D (…, PrimID)

IT_codice_N_2D_COMP (PrimID, GeoID)

codiceC(ClassID,…) e codiceC_codiceAG1_AG(ClassREF, GeoID, tipo)

dove:

- **ClassREF:** contiene l'identificatore ClassID dell'oggetto a cui si riferisce l'aggregato
- **GeoID:** identificatore assegnato al singolo componente dell'aggregato
- **TIPO** codice che indica la tipologia dell'aggregato: line, point, poly

Si noti che nel caso in cui l'aggregato contenga più di un poligono semplice è necessario definire ciascun poligono semplice come un componente dell'aggregato, mentre per i punti e le curve il componente può essere sia il punto/curva semplice o un multipunto/multicurva. Inoltre si noti che l'attributo tipo serve per distinguere gli edge che compongono curve da quelli che compongono i poligoni dell'aggregato.

10.5 Mapping in presenza di collassamento

Per ogni attributo sul quale è previsto il collassamento il mapping definisce nel file Dbf della classe un attributo che riporta il GeoID della geometria principale della componente spaziale collassata (poligoni per una componente spaziale di tipo surface) e si aggiungono al massimo i due attributi per contenere le componenti collassate nei quali si riporta il GeoID della geometria collassata corrispondente. Il nome degli attributi generati sarà composto dal codice alfanumerico dell'attributo geometrico seguito dal suffisso _P e _L per la componente collassata a punti e curve.

10.6 Altri costrutti

Tutti gli altri costrutti si implementano nel modello Shape_Topo esattamente come nel modello Shape_Flat.