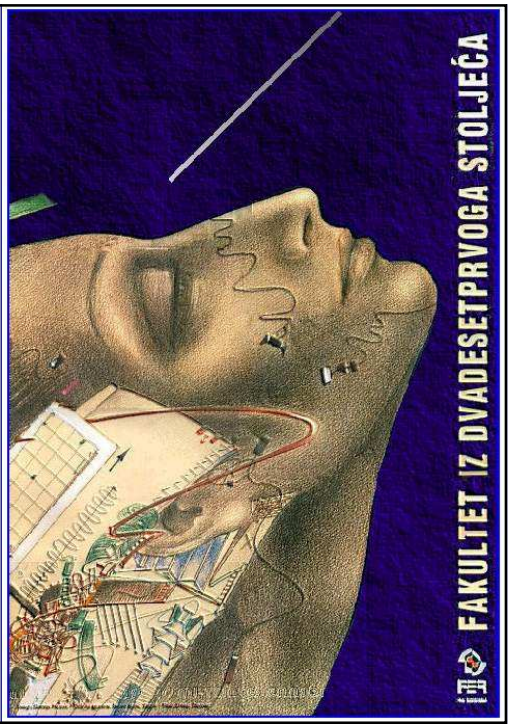
Napredni modeli i baze podataka

Predavanja

8. Geoprostorne baze podataka

Studeni 2008.

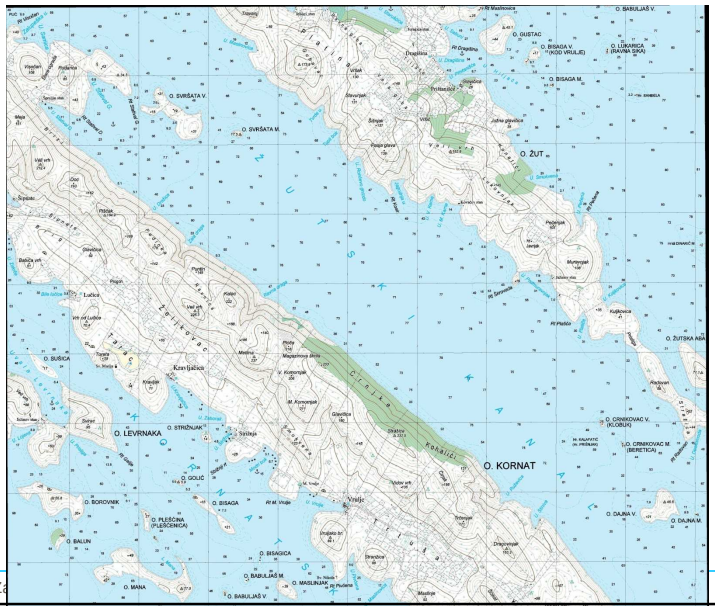


Sadržaj

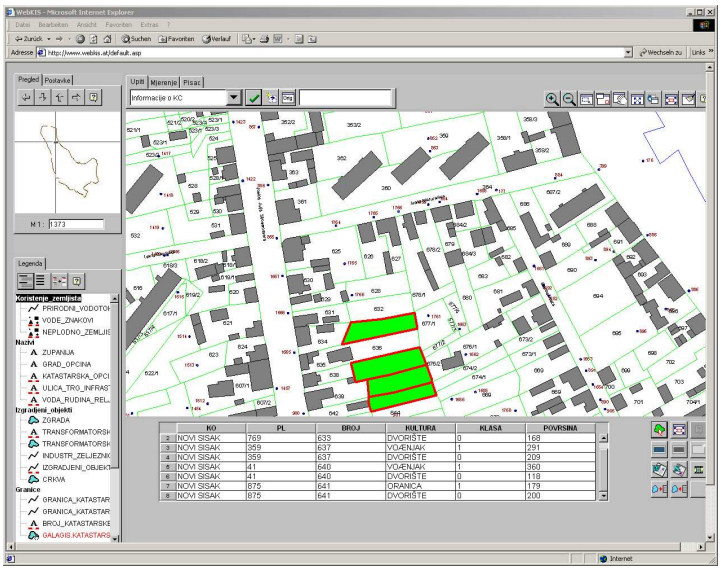
"Nestandardne" DBMS

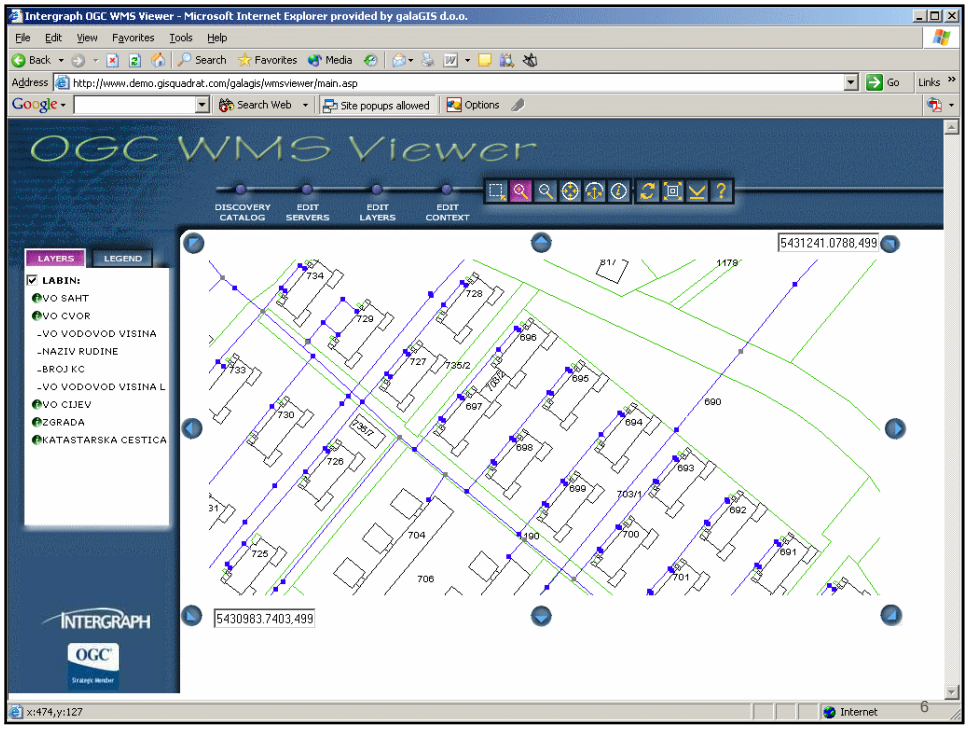
- " Nestandardne" DBMS aplikacijske domene
  - GIS
  - CAD
  - Mobilne baze podataka
  - VLSI Systems
  - Medicina/Biologija (Genetic/Genome Database - GDB)
  - ...
- **"Nestandardni" podaci**
  - slikovni
  - rasterski
  - geometrijski
  - ...
- **Inicijatori/generatori OR i OO DBMS**

Geoinformacijski sustav – GIS



Geoinformacijski sustav – (Web)GIS

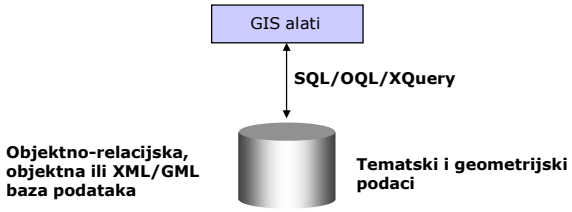




Geoinformacijski sustav - GIS

- Formalna definicija: Informacijski sustav za upravljanje, analizu, vizualiziranje i distribuiranje informacija o objektima i pojavama, čiji referentni sustav je definiran na površini Zemlje

Integrirana GIS arhitektura



- Geoprostorne baze podataka
  - Jedinstvena tehnologija za upravljanje tematskim* i geoprostornim** podacima
  - Objektno-relacijski, objektni ili XML/GML SUBP
  - Generičke GIS funkcije integrirane u SUBP

- * Standardni/konvencionalni podaci
- ** Prostorni/geometrijski podaci

Relacijski model

Parcela (opcina:string, broj:string, oblik:poligon) 1 NF?

Zgrada (opcina:string, adresa:string, tip:string, oblik:poligon) 1 NF?

Ulica (opcina:string, naziv:string, oblik:poligon) 1 NF?

Vod_NN (id:integer, oblik:polilinija) 1 NF?

?!

Relacijski model

Ulica (opcina:string, naziv:string)

Zgrada (opcina:string, adresa:string, tip:string)

Parcela (opcina:string, broj:string)

Tocka (id:integer, x:real, y:real)

Granica (id:integer,pocTocka:integer, krajTocka:integer, lObjekt:string, dObjekt:string)

Relacijski model

točka

id	x	y
5297	6909.04	7911.07
5341	6920.35	7918.69
5417	6936.39	7888.98
5362	6926.51	7883.38
5361	6926.43	7922.82
5407	6933.19	7906.49
5432	6941.76	7892.03
5417	6936.39	7888.98
5401	6932.49	7926.65
5433	6941.80	7906.00
5447	6946.63	7894.81
5434	6941.76	7892.03
5453	6933.19	7906.49

zgrada

općina	adresa	broj	tip
Centar	I. Gundulića	8	stambena
Centar	I. Gundulića	10	poslovna
Centar	I. Gundulića	9	poslovna

ulica

općina	parcela	naziv
Centar	533	I. Gundulića
Trnje	1276	Miramarska

parcela

općina	broj
Centar	535/2
Centar	535/3
Centar	535/9
Centar	540/1
Centar	535/1
Centar	534/1
Centar	469/1
Centar	469/3
Trnje	1276
Trnje	1283/2

granica

počtočka	krajtočka	lobjekt	dobjekt
5297	5341	533	535/2
5341	5417	535/3	535/2
5417	5362	538/1	535/2
5362	5297	535/1	535/2
5341	5361	533	535/3
5361	5407	535/9	535/3
5432	5417	538/1	535/3
5417	5341	535/2	535/3
5361	5401	533	535/9
5401	5433	540/1	535/9

Relacijski model

Upit: Površina parcele broj ’535/2’ ?

SQL: ?! (operacija/funkcija *površina* nije definirana)

Upit: Koje parcele „presijeca“ vod niskog napona 101?

SQL: ?! (operacija/funkcija *presijeca* nije definirana)

Relacijski model

Upit: Lista graničnih točaka parcele ’535/2’

SQL:
```
SELECT DISTINCT id, x, y
FROM   granica, tocka
WHERE
  (lObjekt = '535/2' AND (pocTocka = broj OR krajTocka = broj))
OR
  (dObjekt = '535/2' AND (pocTocka = broj OR krajTocka = broj));
```

id	x	y
5297	6909.04	7911.07
5341	6920.35	7918.69
5362	6926.51	7983.38
5417	6936.39	7988.98

Relacijski model

Modeliranje geoprostornih objekata u principu je moguće, ali:

- Dekomponiranje logički koherentnih objekata u skup relacija
- Narušavanje 1:1 korespondencije između objekata realnog svijeta i njihove prezentacije u relacijskom modelu
- Korisnik promatra i **očekuje** operacije na razini aplikacijskog objekta, a ne na razini skupa relacija
- Nepostojanje relevantnih geometrijskih operacija/geoprostornih funkcija
- Rekonstrukcija objekata rezultira kompleksnim upitima i korištenjem skupih operacija spajanja (***join***)
- Povećanje geometrijske kompleksnosti objekata neminovno vodi ka povećanju kompleksnosti modela podataka
- Kompleksni modeli zahtijevaju kompleksne i skupe (računalni resursi i vrijeme obrade) upite, što u pravilu prouzrokuje drastičan pad performansi GI sustava

Relacijski model

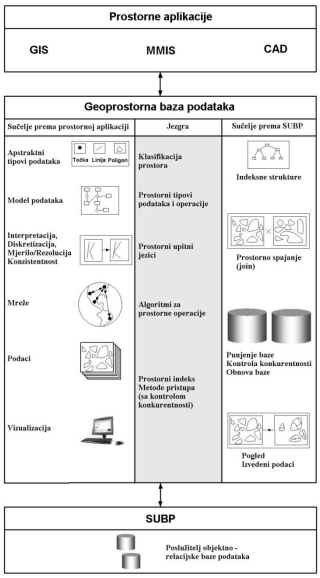
Relacijski model ne odgovara zahtjevima modernih GI sustava

- Relacijska baza podataka (SQL) daje odgovor na svaki upit koji je moguće izraziti relacijskom algebrom, stoga je relacijski model univerzalno prihvaćeni model u standardnim/konvencionalnim aplikacijama
- Nasuprot tome, ne postoji opće prihvaćeni (univerzalni) matematički model geoprostornih baza podataka, što predstavlja izrazito ozbiljnu poteškoću u formalnom definiranju geoprostornih modela podataka i upitnih jezika
- Značajna semantička praznina između GIS-a i relacijskih baza podataka, koja prouzrokuje kompleksnost, probleme i neugodnosti za korisnika

Sustav za upravljanje geoprostornim bazama podataka (SUGBP)

- SUGBP - softverski modul
  - Implementiran proširenjem OR, OO ili polustrukturiranog (XML/GML) modela
  - Posjeduje skup geoprostornih ATP (GeoATP), kao i upitni jezik u kojemu je moguće koristiti te GeoATP
  - Prostorno indeksiranje, djelotvorne algoritme za operacije definirane nad GeoATP, kao i specifična pravila za optimiranje upita

SUGBP



GeoATP

- GIS upiti temelje se na relacijama između geoprostornih objekata:
  - Prikaži sve gradove udaljene od državne ceste br. 8 najviše 1 km
  - Prikaži sve parcele veće od 10000 m^2, udaljene od mora najviše 500m
  - Prikaži sve otoke u Šibensko-kninskoj županiji s površinom > 10000 m^2
  - Prikaži sve gradove koji leže na rijeci Savi a imaju više od 50000 stanovnika

GeoATP

Implementacija (geo)prostornih upita

- ~~Eksplicitno pohranjivanje ***svih*** geometrijskih relacija među objektima~~
  - ~~Za ***n*** objekata mora se pohraniti ***n***^{2} vrijednosti za svaki mogući tip relacije~~
  - ~~Kompleksne procedure za održavanje baze u konzistentnom stanju~~
- Definiranje relacije tijekom izvršenja upita, na temelju njihove geometrije i/ili položaja u prostoru
- Neophodno poznavanje i razumijevanje ***koje*** relacije su moguće i ***kako*** se mogu formalno definirati

GeoATP

- Temeljna obilježja geoprostornih objekata
  - **Geometrijska**
    - *Metrička*
      - Oblik (apstrakcija geometrijske strukture: ***točka***, ***linija***, ***poligon***)
      - Položaj (u odnosu na referentni koordinatni sustav)
      - Veličina (0D, 1D, 2D, 3D)
    - *Topološka*
      - Relacije među objektima (***susjedstvo***, ***povezanost***, itd.)
  - **Tematska**
    - Atributi čije su domene prosti tipovi podataka (`INTEGER,CHAR,...`)

GeoATP

- Apstraktni tipovi podataka, jer:
  - Naš fokus je na samim geoprostornim tipovima podataka, a ne i načinima njihove implementacije
  - Interesiraju nas ***apstraktni matematički objekti***, a ne konkretni programi (aplikacije)
  - Ostavlja se potpuna neovisnost u načinima njihove implementacije

GeoATP

- Zadaci u procesu modeliranja na konceptualnoj razini
  - Dizajn i formalna definicija unutarnje strukture i semantike
  - Formalni opis konceptualnog pogleda vidljiv na korisničkoj razini
  - Dizajn mehanizama za integraciju u SUBP

Kriteriji

- Rigorozna definicija
  - Formalna, jasna i jedinstvena definicija, kako bi se izbjegle eventualne dvosmislenosti i za korisnika i za onoga tko implementira te ATP
- Neovisnost o modelu podataka
  - ATP opisuje skup objekata kroz kolekciju operacija, pri čemu je način implementacije (strukture podataka i algoritmi) sakriven, odnosno nevidljiv za korisnika (učahurenje; ***encapsulation***)
- Djelotvornost
  - Korištenje računalne geometrije (Engl. ***computational geometry***)

GeoATP

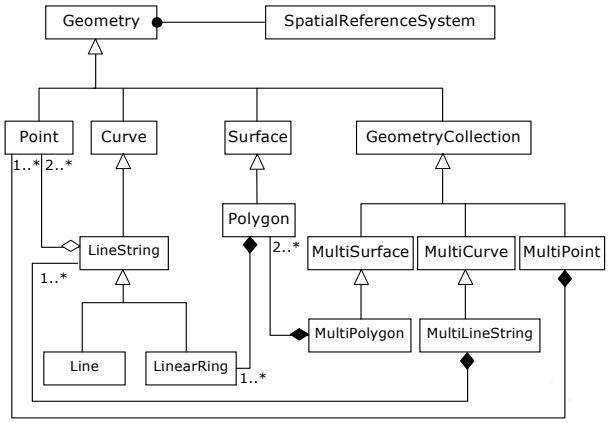
- Operacije
  - Funkcije čiji su argumenti geoprostorni ATP, a rezultat je ili geoprostorni ATP ili prosti tip (skalarna vrijednost)
  - Pretpostavljamo 2D prostor (3D je generalizacija 2D)
  - Zahtjevi
    - **Mali skup operacija**
      - Manji broj operacija smanjuje kompleksnost upitnog jezika
    - **Izražajnost**
      - Operacije trebaju omogućiti korisniku postavljanje širokog opsega upita
    - **Konzistentnost**
      - Skup operacija mora biti formalno specificiran, tako da se osigura kompletnost i međusobna isključivost operacija
    - **Hijerarhijsko strukturiranje**
      - Upitni jezici moraju osigurati hijerarhijsko strukturiranje skupa operatora, u cilju postavljanja upita na različitim razinama granulacije, u kojima geometrijski detalji variraju od manje detaljnih do više detaljnih
    - **Jezička i spoznajna temeljitost**
      - Nazivi operacija (operatori) moraju biti sukladni opće prihvaćenom jezičkom korištenju geoprostornih pojmova i spoznajnim osnovama za geoprostorne koncepte

GeoATP

- Različiti prostorni koncepti vode ka različitim kategorijama modela podataka, tipova podataka, operacija, struktura podataka i algoritama
- Prostorni koncepti - apstrakcije realnosti
  - ***Objektno-orijentirani***
    - Granična reprezentacija
      - Prostor se promatra kao konačna kolekcija skupa točaka bez dimenzije, koje sačinjavaju kontinuum. Svaka točka je definirana svojim koordinatama, koje u 2D prostoru odgovaraju elementima $\mathcal{R}x\mathcal{R}$.
      - Beskonačni skupovi točaka nad ovim prostorom mogu biti konačno specificirani kao entiteti sa graničnom reprezentacijom različitim temeljnim tipovima podataka, koji su obično ***točke***, ***linije*** i ***poligoni***
    - Euklidovska geometrija
    - Grafovi
  - ***Prostorno-orijentirani***
    - Ćelijski-utemeljena podjela prostora
      - **Raster**

GeoATP – Granična reprezentacija

- UML model geometrijskih objektnih klasa*



*OGC Simple Features Specification for SQL (http://www.opengeospatial.org)

GeoATP

- Topološke relacije (*touch, in, disjoint* …)
- Geometrijske operacije
  - Skupovne (unija, presjek …)
  - Aritmetičke (površina poligona, duljina krivulje …)
  - Druge
- Operacije nad grafovima (traženje najkraćeg puta)

GeoATP – Topološke relacije

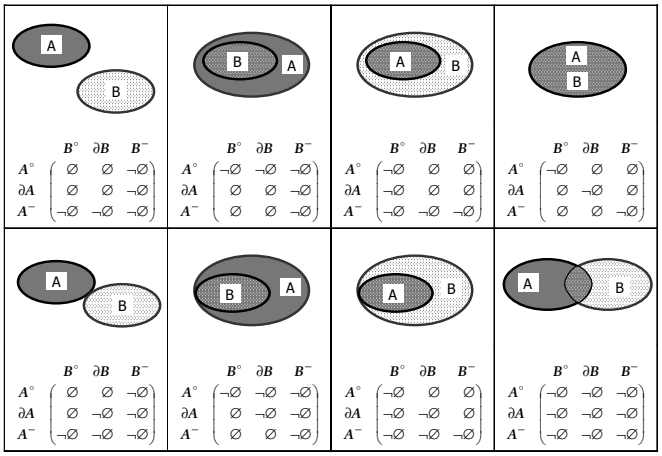
Model 9-presjeka (9-IM)

- Binarna topološka relacija ***R*** između dva GeoATP ***A*** i ***B***, određuje se usporedbom unutrašnjosti (A°), granice (∂A) i vanjštine (A^-) ***A***, i unutrašnjosti (B°), granice (∂B) i vanjštine (B^-) ***B***
- Tih šest komponenata ATP moguće je kombinirati tako, da obrazuju 9 temeljnih opisa topoloških relacija između dva GeoATP
- Uređeni skup tih 9 presjeka može se koncizno reprezentirati matricom:

$$R(A,B) = \begin{pmatrix} A^0 \cap B^0 & A^0 \cap \partial B & A^0 \cap B^- \\ \partial A \cap B^0 & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^0 & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

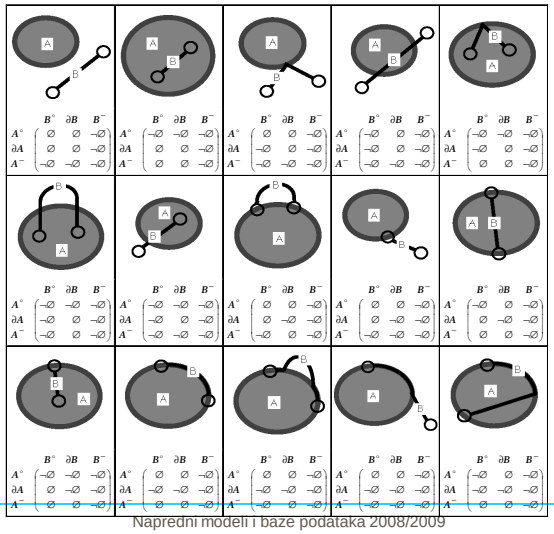
GeoATP – Topološke relacije

Topološke relacije između dva poligona



GeoATP – Topološke relacije

Topološke relacije između poligona i linije (15/19)



GeoATP – Topološke relacije

Dimenzijski prošireni model 9-presjeka (DE-9IM) utemeljen na objektnom računu (*Object-Calculus*)

- Objektni račun je jednostavan formalni jezik, u kojemu notacija <λ_1, *r*, λ_2> znači da su objekti λ_1 i λ_2 u relaciji *r*
  - Pored operatora unutrašnjost (°) granica (∂) i vanjština (¯), uvodimo operator dimenzija (*dim*), koji je funkcija:

$$\dim(S) = \begin{cases} -\text{ ako je } S=\varnothing \\ 0 \text{ ako } S \text{ sadrži barem točku, ali ne i linije i površine} \\ 1 \text{ ako } S \text{ sadrži barem liniju, ali ne površinu} \\ 2 \text{ ako } S \text{ sadrži barem površinu} \end{cases}$$

*S – opći skup točaka

GeoATP – Topološke relacije

Dimenzijski prošireni model 9-presjeka (DE-9IM) dobije se jednostavnim proširenjem svakog 9IM presjeka njegovom dimenzijom:

$$DE9I = \begin{pmatrix} \dim(\partial\lambda_1 \cap \partial\lambda_2) & \dim(\partial\lambda_1 \cap \lambda_2^0) & \dim(\partial\lambda_1 \cap \lambda_2^-) \\ \dim(\lambda_1^0 \cap \partial\lambda_2) & \dim(\lambda_1^0 \cap \lambda_2^0) & \dim(\lambda_1^0 \cap \lambda_2^-) \\ \dim(\lambda_1^- \cap \partial\lambda_2) & \dim(\lambda_1^- \cap \lambda_2^0) & \dim(\lambda_1^- \cap \lambda_2^-) \end{pmatrix}$$

GeoATP – Topološke relacije

Relacija *touch* važi za grupe poligon/poligon, linija/linija, linija/poligon, točka/poligon i točka/linija:

$$\langle \lambda_1, touch, \lambda_2 \rangle \Leftrightarrow (\lambda_1^0 \cap \lambda_2^0 = \varnothing) \wedge (\lambda_1 \cap \lambda_2 \neq \varnothing)$$

Relacija *cross* važi za grupe linija/linija i linija/poligon:

$$\langle \lambda_1, cross, \lambda_2 \rangle \Leftrightarrow (\dim(\lambda_1^0 \cap \lambda_2^0) = \max(\dim(\lambda_1^0), \dim(\lambda_2^0)) - 1) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_1) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_2)$$

Relacija *in* važi za sve grupe:

$$\langle \lambda_1, in, \lambda_2 \rangle \Leftrightarrow (\lambda_1 \cap \lambda_2 = \lambda_1) \wedge (\lambda_1^0 \cap \lambda_2^0 \neq \varnothing)$$

GeoATP – Topološke relacije

Relacija *overlap* važi za homogene grupe poligon/poligon i linija/linija:

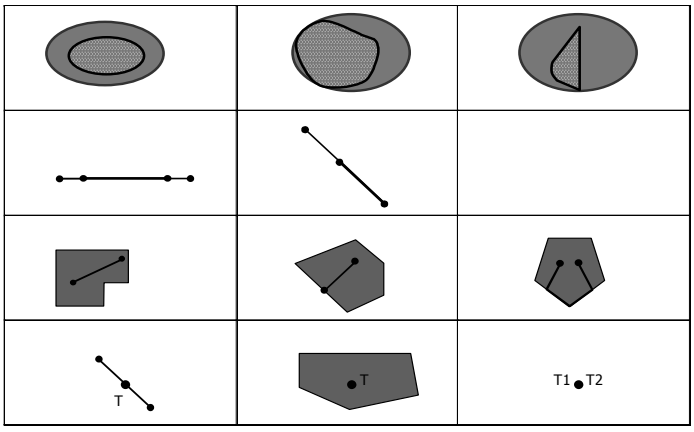
$$\langle \lambda_1, overlap, \lambda_2 \rangle \Leftrightarrow \left(\dim(\lambda_1^0) = \dim(\lambda_2^0) = \dim(\lambda_1^0 \cap \lambda_2^0)\right) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_1) \wedge (\lambda_1 \cap \lambda_2 \neq \lambda_2)$$

Relacija *disjoint* važi za sve grupe:

$$\langle \lambda_1, \text{disjoint}, \lambda_2 \rangle \Leftrightarrow \lambda_1 \cap \lambda_2 = \varnothing$$

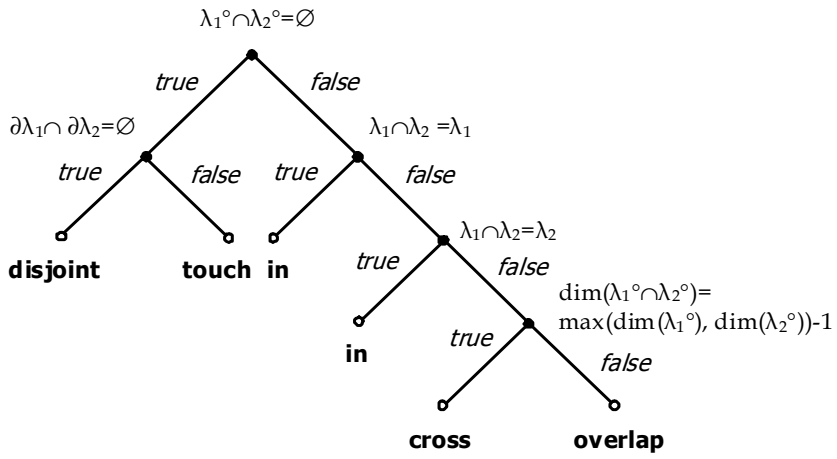
GeoATP – Topološke relacije

Topološka relacija *in* :



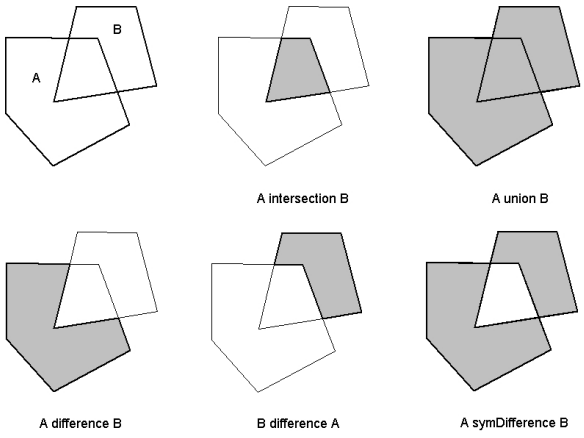
GeoATP – Topološke relacije

Uzajamna isključivost i potpuna pokrivenost



GeoATP – Geometrijske operacije

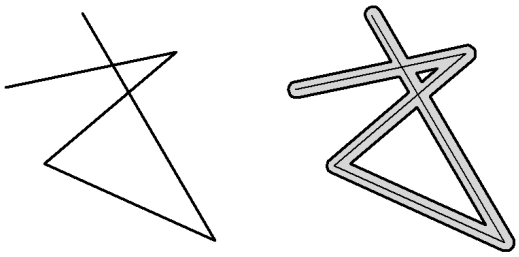
Skupovne geometrijske operacije



- $A \cup B = \{x \in \Re^2 \mid x \in A \vee x \in B\}$
- $A \cap B = \{x \in \Re^2 \mid x \in A \wedge x \in B\}$
- $A - B = \{x \in \Re^2 \mid x \in A \wedge x \notin B\}$
- $A \oplus B = \{x \in \Re^2 \mid (x \in A \wedge x \notin B) \cup (x \in B \wedge x \notin A)\}$

GeoATP – Geometrijske operacije

Bafer (*buffer*)



(a) (b)

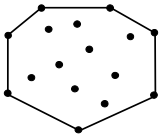
$$buffer(\delta) = \quad \delta > 0 : \{x \in \Re^2 \mid d(x,g) \leq \delta\}$$

$$\delta < 0 : \{x \in \Re^2 \mid x \in g \wedge d(x, \partial g) > \delta\}$$

GeoATP – Geometrijske operacije

Konveksna ljuska (*convexHull*)

- Konveksna ljuska skupa točaka ***S***, označena sa ***convexHull(S)*** je najmanji poligon P, za koji je svaka točka iz skupa ***S*** ili na granici ili u unutrašnjosti.



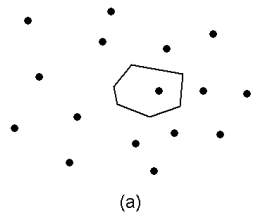
(a) (b) (c)

GeoATP – Geometrijske operacije

Voronoijev dijagram (*voronoi*)

Geoprostorni upit korisnika:

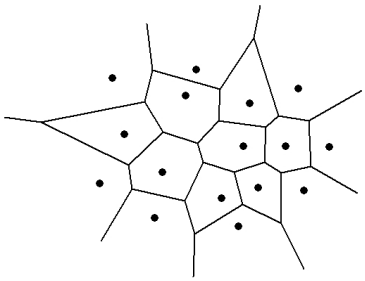
- *Odredi zone u Hrvatskoj koje gravitiraju gradovima, čiji broj stanovnika je veći od 50000.*



(a)

Geometrijska operacija nad GeoATP:

- *Za svaku točku ravnine iz skupa S odrediti koja od N točaka iz podskupa T ($t_i \in T \subset S$) je najbliža.*



(b)

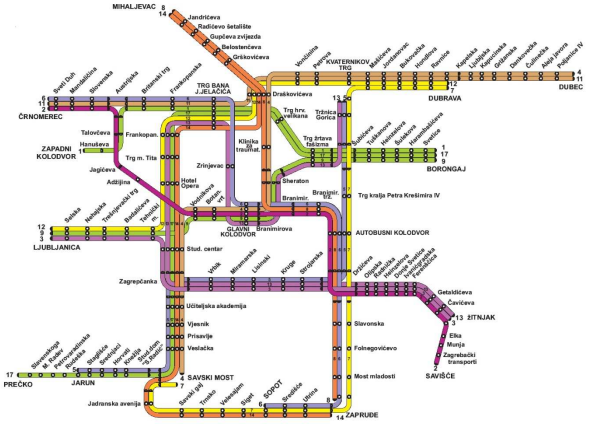
voronoi(t) $\Leftrightarrow \{t' \in \ | \forall q \in T: d(t',t) \ \ d(t',q)\}$

GeoATP – Graf operacije

Najkraći put (*shortestPath*)

Geoprostorni upit korisnika:

- *Najkraći put između stanica* Zapadni kolodvor *i* Velesajam.



Objektni model*

- Fleksibilnost za modeliranje *strukture* kompleksnih objekata i *operacija* nad tim objektima
  - Povećano korištenje OO programskih jezika u razvoju GI aplikacija
- Rezultat integracije objektno-orijentirane paradigme razvijene u programskim jezicima/softverskom inženjerstvu i baza podataka
  - Objekt definiraju dvije komponente:
    - Stanje – tekuća vrijednost
    - Ponašanje – skup operacija nad objektom

*Objektno-orijentirani model

Objektni model

- Razlike između OO PJ i ODB:
  - U objektno-orijentiranim PJ objekti egzistiraju samo tijekom izvršenja programa (prolazni/privremeni objekti)
  - U objektnim bazama podataka objekti egzistiraju nakon izvršenja programa (stalni/perzistentni objekti), uz mogućnost njihovog istovremenog korištenja u različitim programima/aplikacijama

Objektni model

- Polimorfizam (preopterećenje operatora)
  - Više implementacija jedne operacije mogu imati isto ime
- Nasljeđivanje
  - Mogućnost objektne klase/tipa da nasljeđuje atribute i operacije neke druge (super)klase / (super) tipa
  - Jednostavno (*simple*) i višestruko (*multiple*)
- Operacije nad objektima
  - Skup proizvoljno kompleksnih operacija nad objektima, koje odgovaraju potrebama korisnika u kompleksnim aplikacijskim okolišima

Objektni model

- Jedan od razloga za uspjeh relacijskih baza podataka jeste postojanje SQL standarda
- Jedan od razloga za sporo prihvaćanje objektnog modela bilo je nepostojanje standarda za objektni model
- ODMG (Object Database Management Group) standard
  - ODMG 3.0
    - ODMG Object Model
    - Object Definition Language (ODL)
    - Object Query Language (OQL)
    - C++/Smaltalk/Java vezivanje (*binding*)

http://www.odmg.org

ODMG Object Model

- Specificira vrste semantika koje mogu biti eksplicitno definirane u objektnim bazama podataka
- *De facto*: skup sučelja (*interface*)
  - Sučelje (**interface**) opisuje atribute, operacije i relacije sa drugim tipovima objekata
  - Nije moguće kreirati primjerke objekata jednog sučelja
  - Objektne klase definirane od strane korisnika nasljeđuju apstraktne operacije sučelja
  - Ključna riječ klasa (*class*) rezervirana je za deklaraciju korisnički/aplikacijski-specifičnih klasa

ODMG Object Model

- Svi objekti, pa tako i oni definirani od strane korisnika, nasljeđuju sučelje Objekt:
- ```
  interface Object {
          enum    Lock_Type(read, write, upgrade);
          void    lock(in Lock_Type mode)
                  raises (LockNotGranted);
          boolean try_lock(in Lock_Type mode);
          boolean same_as(in Object anObject);
          Object  copy();
          void    delete();
      };
  ```
  ```

ODMG Object Model

- Objekti se kreiraju pozivanjem operacije na sučelju tvornice (*factory interface*):

```
interface ObjectFactory {
      Object  new();
  };
```


ODMG Object Model - `Collection`

```
interface Collection : Object {
       exception      InvalidCollectionType{};
       exception      ElementNotFound{Object element;};
       unsigned long  cardinality();
       boolean is_empty();
       …
       boolean contains_element(in Object element);
       void    insert_element(in Object element);
       void    remove_element(in Object element)
                      raises(ElementNotFound);
       Iterator       create_iterator(in boolean stable);
       …
};
```

ODMG Object Model - Set

```
interface SetFactory : ObjectFactory {
   Set   new_of_size(in long size);
};

interface Set : Collection {
   attribute     set<t> value;
   Set           create_union(in Set other_set);
   Set           create_intersection(in Set other_set);
   Set           create_difference(in Set other_set);
   boolean       is_subset_of(in Set other_set);
   …
   boolean       is_superset_of (in Set other_set);
   …
};
```


ODL

- Standardizirani jezik za specificiranje strukture/sheme baze podataka korištenjem objektno-orijentirane paradigme
  - Proširenje IDL (*Interface Description Language* - CORBA* komponenta)
  - Kreiranje specifikacije objekata (klasa i sučelja) sukladno semantičkim konstrukcijama ODMG objektnog modela, neovisno o programskim jezicima
  - Korištenjem jednog od standardnih vezivanja, korisnik specificira način preslikavanja ODL konstrukcija u specifični programski jezik

*Common Object Request Broker Architecture – distribuirano, objektno-orijentirano računarstvo

ODL – deklaracija klase

- Deklaracija klase opisuje:
  - Atribute
  - Relacije s drugim objektima
  - Metode – operacije (funkcije) koje se mogu koristiti nad objektima klase
    - ODL specificira potpise (*signature)* metoda, a ne njihove implementacije
    - Implementacije metoda realiziraju se jednim od jezika (C++, Smalltalk, Java) definiranim ODMG standardom

ODL – module OpenGIS

```
module OpenGIS {
   struct WKSPoint {double x, double y};
   typedef list<WKSPoint> WKSLineString;
   interface Geometry {
      readonly attribute SpatialReferenceSystem spatial_reference_system;
      // constructive operators
      Geometry boundary();
      Geometry buffer (in double distance);
      Geometry convex_hull();
      // relational operators
      boolean equals (in Geometry other);
      boolean touches (in Geometry other);
      boolean contains (in Geometry other);
      boolean within (in Geometry other);
      boolean disjoint (in Geometry other);
      boolean crosses (in Geometry other);
      boolean overlaps (in Geometry other);
      boolean intersects (in Geometry other);
      // set operators
      Geometry intersection (in Geometry other);
      Geometry union(in Geometry other);
      Geometry difference (in Geometry other);
   };
```

ODL – module OpenGIS

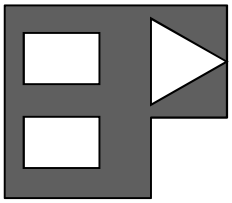
```
interface Surface : Geometry {
   attribute double area;
   attribute Point centroid;
   boolean is_planar();
};
interface Curve : Geometry {
   readonly attribute Point start_point;
   readonly attribute Point end_point;
   …
};
interface Ring : Curve {
};
interface LineString : Curve {
   readonly attribute long num_points;
   Point get_point_by_index (in long index) raises (InvalidIndex);
   …
};
```


ODL – module OpenGIS

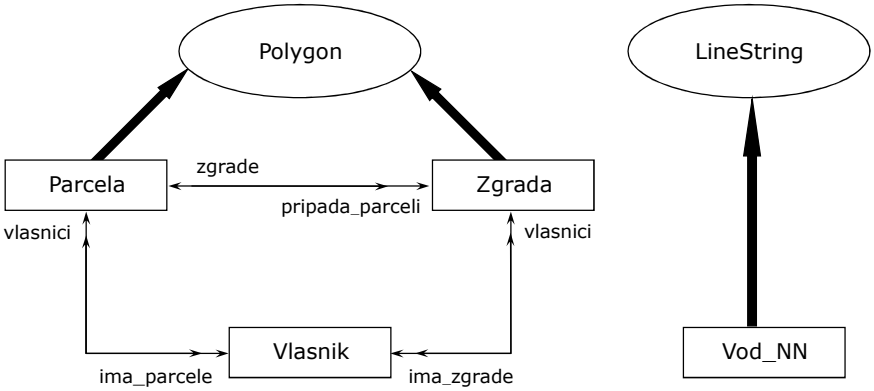
```
interface LineString : Curve {
   exception          InvalidIndex {};
   exception          MinimumPoints {};
   readonly attribute longnum_points;
   Point                          get_point_by_index (in long index) raises
(InvalidIndex);
   void               set_point_by_index (in Point new_point, in long index)
                  raises (InvalidIndex);
   void                    insert_point_by_index (in Point new_point, in long
index)
                  raises (InvalidIndex);
   void               append_point (in Point new_point);
   void               delete_point_by_index (in long index)
                  raises (InvalidIndex, MinimumPoints);
};
interface LineStringFactory : GeometryFactory {
      exception … {};
      LineString create_from_LineString(in LineString existing);
   };
   interface LinearRing : Ring, LineString {
   };
```

ODL – module OpenGIS

```
    interface Polygon : Surface {
       attribute LinearRing boundary;
       attribute MultiCurve interior_rings;
    };
    interface PolygonFactory : GeometryFactory {
       exception InvalidWKSLinearString {};
       Polygon create_from_Polygon(in Polygon existing)
               raises (InvalidWKSLinearString);
    };
    …
}; // End of OpenGIS Module
```



ODL – myGIS primjer



ODL – myGIS primjer

```
module myGIS {
typedef array<char,13>     JMBG;
struct Adresa {
string    grad;
string    ulica;
short     kucni_broj;
long      postanski_broj;
};
class Zgrada;       // forward deklaracija
class Vlasnik;      // forward deklaracija
```


ODL – myGIS primjer

```
class Parcela : OpenGIS ::Poligon
(extent parcele)
{
   attribute string        opcina;
   attribute short         broj;
   attribute short         podbroj;
   attribute string        naziv;
   attribute long          broj_plana;
   attribute double        area;
   attribute OpenGIS::Point       centroid;
   readonly attribute OpenGIS::Ring       exterior_ring;
   readonly attribute OpenGIS::MultiCurve interior_rings;
   relationship set<Zgrada> zgrade inverseZgrada::pripada_parceli;
   relationship set<Vlasnik> vlasnici     inverse
Vlasnik::ima_parcele;
};
```

ODL – myGIS primjer

```
enum TipZgrade{Stambena, Poslovna, Obrazovna, Administrativna};
class Zgrada : OpenGIS::Poligon
(extent zgrade)
{
   attribute Adresa         adresa;
   attribute TipZgrade              tip;
   attribute double         area;
   attribute OpenGIS::Point         centroid;
   readonly attribute OpenGIS::Ring        exterior_ring;
   readonly attribute OpenGIS::MultiCurve  interior_rings;
   relationship Parcela pripada_parceli    inverse Parcela::zgrade;
   relationship set<Vlasnik> vlasnici      inverse
Vlasnik::ima_zgrade;
};
```


ODL – myGIS primjer

```
class Vlasnik
(extent vlasnici)
{
   attribute string         ime;
   attribute string         prezime;
   Adresa           adresa;
   JMBG             jmbg;
   relationship set<Parcela> ima_parcele   inverse Parcela::vlasnici;
   relationship set<Zgrada> ima_zgrade     inverse Zgrada::vlasnici;
};
 class Vod_NN : OpenGIS::LineString
   (extent vodovi_nn) // skup perzistentnih objekata klase Vod_NN
   {
      atribute integer id;
   }
};// Kraj modula myGIS
```

OQL

- Utemeljen na ODMG Objektnom Modelu
- *De facto* SQL-92 SELECT proširen OO konceptima
- Posjeduje *operatore visoke razine* za rad sa skupovima objekata
- Funkcionalni jezik - omogućuje proizvoljno komponiranje operatora
- OQL se može koristiti u programskim jezicima definiranim ODMG standardom
- OQL može koristiti metode/operacije implementirane u tim jezicima

OQL - myGIS

Upit: Površina parcele broj ’535/2’ ?

OQL:
```
SELECT p.povrsina()
FROM   parcele p
WHERE p.broj = “535/2”
```

Upit: Koje parcele „presijeca“ vod niskog napona 101?

OQL:
```
SELECT  p.broj
FROM   nn IN vodovi_nn, p IN parcele
WHERE nn.crosses(p)
```

OQL - myGIS

Upit: Granice svih parcela u općini 'Maksimir' ?

OQL:
```
SELECT  p.boundary()
FROM    p in parcele
WHERE p.opcina = 'Maksimir';
```

Upit: Granice parcela u općini 'Centar' koje su susjedne parceli '535/2'?

OQL:
```
SELECT  p.boundary()
 FROM   parc IN parcele, p IN parcele
 WHERE parc.opcina = 'Centar'
  AND     parc.broj = 535 and parc.podbroj = 2
 AND     parc.touches(p);
```


ODL – myGIS primjer

```
class Autocesta : OpenGIS::LineString
(extent autoceste)
{
   attribute array<char,2>         oznaka;
   readonly attribute long         num_points;
   readonly attribute double       length;
   readonly attribute OpenGIS::Point      start_point;
   readonly attribute OpenGIS::Point      end_point;
};
class Rijeka : OpenGIS::LineString
(extent rijeke)
{
   attribute string                naziv;
   readonly attribute long         num_points;
   readonly attribute double       length;
   readonly attribute OpenGIS::Point      start_point;
   readonly attribute OpenGIS::Point      end_point;
};
```

ODL – myGIS primjer

```
class Grad : OpenGIS::Point
(extent gradovi)
{
   attribute string        naziv;
   attribute long          broj_stanovnika;
   attribute OpenGIS::WKSPoint     coordinates;
};
class Zupanija : OpenGIS::MultiPoligon
(extent zupanije)
{
   attribute string                naziv;
   readonly attribute double       area;
   readonly attribute long         number_of_elements;
};
```


OQL - myGIS

Upit: Geometrija i duljina autoceste A1, u dijelu koji prolazi Šibensko-kninskom županijom?

OQL:
```
SELECT struct(a.intersection(z) as geometrija,
              a.intersection(z).length() as duljina)
FROM   a in autoceste, z in zupanije
WHERE a.oznaka = 'A1' and z.naziv = 'Šibensko-kninska';
```

Upit: Nazivi i koordinate (položaj) gradova Splitsko-dalmatinske županije?

OQL:
```
SELECT    g.naziv, g.coordinates
FROM      g in gradovi, z in zupanije
WHERE     z.naziv = 'Splitsko-dalmatinska'
AND       g.within(z);
```

ODMG Java vezivanje

- OQL = SQL-92 SELECT + OO koncepti
- OML nije definiran ODMG standardom (!)
- ODMG: " *... we don't believe exclusively in a universal DML syntax...*"
- Za razliku od SQL u relacijskim SUBP, ODMG DMLs (OMLs) su specificirani za OO programske jezike (C++, Smalltalk i Java), u cilju osiguranja jedinstvenog, integriranog okoliša za programiranje i manipuliranje podacima

ODMG Java vezivanje

- Korisnik/Programer mora razumjeti i doživljavati vezivanje kao jedinstven jezik za opisivanje/specifikaciju i operacija baza podataka i operacija programskog jezika
  - Jedinstven sustav tipova podataka i za Java jezik i za bazu podataka
  - Individualni primjerci tih zajedničkih tipova podataka mogu biti perzistentni ili prolazni
  - Vezivanje respektira Java sintaksu → Java jezik nije potrebo modificirati
  - Vezivanje respektira Java semantiku automatskog upravljanja memorijom

ODMG specifikacija definira ODL, OML i OQL preslikavanje

- Java ODL, **Java OML** i **Java OQL**

Java OML

```
package com.myGIS.OpenGIS;
import java.io.Serializable;
import java.util.Collection;
import java.util.Iterator;
import com.myGIS.algorithm.*;
import com.myGIS.io.WKTWriter;
import com.myGIS.operation.buffer.BufferOp;
import com.myGIS.operation.distance.DistanceOp;
import com.myGIS.operation.overlay.OverlayOp;
import com.myGIS.operation.relate.RelateOp;
import com.myGIS.operation.valid.IsValidOp;
import com.myGIS.util.Assert;
```


Java OML

```
public abstract class Geometry implements Cloneable,
                                          Comparable, Serializable
{
    protected Envelope envelope;
    private final static Class[] sortedClasses = new Class[] {
      Point.class,
      MultiPoint.class,
      LineString.class,
      LinearRing.class,
      MultiLineString.class,
      Polygon.class,
      MultiPolygon.class,
      GeometryCollection.class
    };

    public Geometry(GeometryFactory factory) {
      this.factory = factory;
      this.SRID = factory.getSRID();
   }
   ...
```

Java OML

```
public Geometry(GeometryFactory factory) {
   this.factory = factory;
   this.SRID = factory.getSRID();
}
public abstract boolean isEmpty();
public abstract Geometry getBoundary();
public boolean disjoint(Geometry g) {
      return relate(g).isDisjoint();
}
public boolean touches(Geometry g) {
      return relate(g).isTouches(getDimension(), g.getDimension());
}
public boolean intersects(Geometry g) {
      return relate(g).isIntersects();
}
public boolean crosses(Geometry g) {
      return relate(g).isCrosses(getDimension(), g.getDimension());
}
```


Java OML

```
public boolean within(Geometry g) {
      return relate(g).isWithin();
}
public boolean contains(Geometry g) {
      return relate(g).isContains();
}
public boolean overlaps(Geometry g) {
      return relate(g).isOverlaps(getDimension(), g.getDimension());
}
public Geometry buffer(double distance) {
      return   fromInternalGeometry (
        BufferOp.bufferOp(toInternalGeometry(this), distance));
}
public Geometry convexHull() {
return fromInternalGeometry (
       (new ConvexHull(toInternalGeometry(this))).getConvexHull());
}
```

Java OML

```
  public Geometry intersection(Geometry other) {
        …
  }
  public Geometry union(Geometry other) {
  checkNotGeometryCollection(this);
        …
  }
  public Geometry difference(Geometry other) {
        …
  }
  public Geometry symDifference(Geometry other) {
        …
  }
} // abstract class Geometry
```


Java OML

```
  // polygon.java
  package com.myGIS.OpenGIS;
  import java.util.Arrays;
  import com.myGIS.algorithm.*;
  public class Polygon extends Geometry {
    protected LinearRing exterior_ring = null;
    protected LinearRing[] interior_rings;

    public Polygon(LinearRing shell, LinearRing[] holes,
                   GeometryFactory factory)  {
      super(factory);
      if (shell == null) {
        shell = getFactory().createLinearRing((CoordinateSequence)null);
      }
      if (holes == null) {
        holes = new LinearRing[]{};
      }
      this.shell = shell;    this.holes = holes;
    }
    …
  }
```

Java OML

```
// parcela.java
package com.myGIS.GIS;
import com.myGIS.OpenGIS.*;
import org.odmg.*;
import java.util.Date;
import com.myGIS.Vlasnik;
import com.myGIS.Zgrada;
class Parcela extends Polygon
{
   String        opcina;
   short         broj;
   short         podbroj;
   String        naziv;
   long          broj_plana;
   DSet          zgrade;
   DSet          vlasnici;
```


Java OML

```
   public Parcela(LinearRing granica, LinearRing[] otvori,
                  GeometryFactory tvornica,
                  String o, short br, short podbr, long plan, double p)
   {
     super (granica, otvori, tvornica);
     opcina = o; broj = br; podbroj = podbr;
     broj_plana = plan; povrsina = p;
     naziv = Integer.toString(br) + "/" + Integer.toString(podbr);
     vlasnici = ODMG.get().newDSet();
     zgrade  = ODMG.get().newDSet();
   }
   public void dodajVlasnika(Vlasnik v) {
         vlasnici.add(v);
   }
   public void dodajZgradu(Zgrada z) {
         zgrade.add(z);
   }
}
```

Java OML

```
// odmg_primjer.java
import org.odmg.ODMGException;
...
import com.myGIS.OpenGIS.*;
import com.myGIS.Parcela;
public class ODMG_Primjer
{
   static void unesiParcele (Database gisdb) throws ODMGException {
      Transaction trans = new Transaction(gisdb);
      trans.begin();
      try  {
         Parcela parc;
         short broj, podbroj;
         long  plan;
         Coordinate[] koordinate;
         LinearRing granica;
         LinearRing[] otvori = new LinearRing[0];
         GeometryFactory tvornica = new GeometryFactory();
```


Java OML

```
         koordinate = new Coordinate[]{
               new Coordinate(6909.04,7911.07),
               new Coordinate(6920.35,7918.69),
               new Coordinate(6936.39,7888.98),
               new Coordinate(6926.51,7883.38),
               new Coordinate(6909.04,7911.07)
          };
          granica = new GeometryFactory().createLinearRing(koordinate);
          broj = 535; podbroj = 2; plan = 34652;
          parc = new Parcela (granica, otvori, tvornica,
                              "Centar", broj, podbroj, plan, povrsina);
          // korištenjem operacije makePersistent
          gisdb.makePersistent(parc);
          // izvrši promjene u bazi podataka
          trans.commit();
      }
      catch (ODMGRuntimeException exc) {
         trans.abort(); throw exc;
      }
}
```

Java OQL

OQLQuery sučelje omogućuje prosljeđivanje parametara upitu, izvršenje upita i dohvaćanje rezultata:

```
public interface OQLQuery {
    public void create (String query) throws QueryInvalidException;
    public void bind (Object parameter)
                throws QueryParameterCountInvalidException,
                       QueryParameterTypeInvalidException;
    public Object       execute() throws QueryException;
};
```


Java OQL

```
public void susjedneParcele(Database gisdb) throws ODMGException {
    DSet susjedne;
    Transaction trans = new Transaction(gisdb);
    trans.begin();
    try {
      String opcina = new String ("Centar");
      Short br = Short.valueOf("535"); Short podb = Short.valueOf("2");
      OQLQuery upit = new OQLQuery(trans);
      upit.create("select p.opcina, p.br, p.podbr
                   from parc in parcele, p in parcele
                   where parc.opcina = \"$1\" and parc.broj = $2
                   and parc.podbroj = $3 and parc.touches(p)");
      upit.bind(opcina); upit.bind(br); upit.bind(podbr);
      susjedne = (DSet) upit.execute();
    }
    catch (ODMGRuntimeException x) {
      trans.abort();
    }
    catch (QueryException x) {
      trans.abort();
    }
   trans.commit();
}
```

Objektni model

Pro

- Definiran standard
- *De facto,* nema ograničenja u modeliranju geoprostornih (GIS) aplikacija
- Proizvoljno kompleksni objekti i operacije
- Upitni jezik visoke razine OQL (sličan SQLu)
- Standardan način vezivanja sa OO programskim jezicima

Contra

- Relativno kompleksan u odnosu na relacijski model
- Zahtijeva poznavanje i relacijskog modela i OO paradigme
- "Ignoriran" od strane vodećih proizvođača DB softvera
- ODB razvijaju relativno male tvrtke → nesigurnost korisnika
- Upitne performanse

Objektno-relacijski model

- Relacijski model proširen konceptima objektnog modela
  - SQL/Object - dio SQL:2003 standarda
    - Proširenje SQL-92 standarda objektno-orijentiranim konceptima
      - Strukturirani tipovi atributa
      - Identifikatori objekata
      - ATP (Apstraktni tipovi podataka)
      - Učahurenje operacija
      - Nasljeđivanje
  - SQL Multimedia and Application Packages – **Part 3 : Spatial**

Objektno-relacijski model

- Strukturirani tipovi atributa
  - Pored atomičnih tipova podataka, domene atributa relacije mogu biti strukturirani tipovi
    - N-torka (**row); (***tuple*, *struct* u ODL)
    - Niz (**array**)
    - Skup n-torki (relacija) – Ugniježđena relacija (*nested relation*)
    - Drugi strukturirani tip

```
CREATE TYPE Tocka AS OBJECT (
        x   NUMBER,
        y   NUMBER
);
CREATE TABLE Tocke OF Tocka;
CREATE TYPE nizLinija AS ARRAY(1024) OF Tocka;
```


Apstraktni tip podataka

- Odgovara konceptu *klase* u objektnom modelu
- Skup atributa i metoda (operacija)
- Definicija tipa sadrži samo **signature** metoda (operacija)

```
CREATE TYPE Geometry AS (
  dimension             SMALLINT DEFAULT –1,
  coordinateDimension   SMALLINT DEFAULT 2,
)
NOT INSTANTIABLE NOT FINAL
METHOD relate (other Geometry matrix CHARACTER(9))
RETURNS CHARACTER VARYING(5) LANGUAGE SQL,
METHOD touches (other Geometry) RETURNS CHARACTER VARYING(5) LANGUAGE SQL,
METHOD within (other Geometry) RETURNS CHARACTER VARYING(5) LANGUAGE SQL,
METHOD crosses (other Geometry) RETURNS CHARACTER VARYING(5) LANGUAGE SQL,
METHOD convexHull (other Geometry) RETURNS Geometry LANGUAGE SQL,
METHOD difference (other Geometry) RETURNS Geometry LANGUAGE SQL,
...
```

Apstraktni tip podataka

- Implementacija metoda
  - Interne metode (SQL)
  - Eksterne metode (programski jezik: C, C++, Java)

```
CREATE METHOD crosses (other Geometry) RETURNS CHARACTER VARYING(5)
FOR Geometry RETURN
        CASE
            WHEN (SELF.dimension() = 0 AND other.dimension() = 1) THEN
                SELF.relate (other, 'T*T******')
            WHEN (SELF.dimension() = 0 AND other.dimension() = 2) THEN
                SELF.relate (other, 'T*T******')
            WHEN (SELF.dimension() = 1 AND other.dimension() = 1) THEN
                SELF.relate (other, '0********')
            WHEN (SELF.dimension() = 1 AND other.dimension() = 2) THEN
                SELF.relate (other, 'T*T******')
            ELSE
                NULL
        END
```


Objektno-relacijski model

- Nasljeđivanje, eksterne metode

```
CREATE TYPE Polygon UNDER Geometry AS (
    boundary      Curve;
    interiorRings Curve ARRAY[32] DEFAULT ARRAY[];
)
INSTANTIABLE
NOT FINAL

CONSTRUCTOR METHOD Polygon (aBoundary LineString, rings LineString ARRAY[32])
RETURNS Polygon SELF AS RESULT
LANGUAGE SQL,

METHOD area () RETURNS REAL LANGUAGE JAVA,
METHOD centroid() RETURNS Point LANGUAGE JAVA
```

Objektno-relacijski model

- Deklaracija i implementacija eksternih metoda

```
CREATE METHOD area () RETURNS REAL
FOR Polygon
EXTERNAL NAME ' myGIS.Polygon.area() return java.lang.Double'
PARAMETER STYLE JAVA

CREATE METHOD centroid() RETURNS Point
FOR Polygon
EXTERNAL NAME ' myGIS.Polygon.centrod () return myGIS.Point'
PARAMETER STYLE JAVA
```


Objektno-relacijski model

- Identifikatori objekata

```
CREATE TYPE Parcela UNDER Polygon AS (
    opcina CHARACTER VARYING(12),
    broj CHARACTER VARYING(8),
)
CREATE TABLE Parcele OF Parcela (
   REF IS parcela_id SYSTEM GENERATED
)
CREATE TYPE Vod_NN UNDER LineString AS (
    napon NUMBER
)
CREATE TABLE vodovi_nn OF Vod_NN (
   REF IS vod_id SYSTEM GENERATED
)
```

Objektno-relacijski model

Upit: Površina parcele broj ’535/2’ ?

SQL:
```
SELECT p.povrsina()
FROM   parcele p
WHERE p.broj = “535/2”
```

Upit: Granice parcela koje „presijeca“ vod niskog napona 101?

SQL:
```
SELECT  p.boundary()
FROM    Vodovi_NN nn, Parcele p
WHERE  nn–>vod_id = 101
AND       nn.crosses(p) = 'TRUE'
```


Objektno-relacijski model

Pro:

- Definiran SQL standard (SQL:2003)
- *De facto*, nema ograničenja u modeliranju GI aplikacija
- Proizvoljno kompleksni objekti i operacije
- Minorna proširenja SQL-a
- Podržan od strane vodećih proizvođača DB softvera
  - **Oracle (Spatial)**
  - **SQL Server 2008 (*geography* and *geometry* data types)**
  - **Informix (Spatial DataBlade Module, Geodetic DataBlade Module)**
  - **DB2 (Spatial Extender)**
- Podržan od strane vodećih proizvođača GIS softvera
  - **Autodesk (AutoCAD Map 3D, MapGuide)**
  - **INTERGRAPH (GeoMedia Pro)**
  - **ESRI (ArcGIS)**
  - **MapInfo (SpatialWare/Informix, GIS Extension/Oracle Spatial)**

Contra (?!):

- Zahtijeva poznavanje relacijskog modela i OO/ATP paradigme

Indeksiranje prostornih podataka: R-stablo

- Slično B-stablima
- Indeksira se položaj objekta u bazi (koordinate)
- Dijeli prostor na minimalne ograničavajuće pravokutnike (*MBR – minimal bounding ractangle / bounding box*)
- Svaki čvor može sadržavati više elemenata
- Element unutrašnjeg čvora sadrži pokazivače na svoju djecu te MBR unutar kojeg se njegova djeca nalaze
- Element lista sadrži identifikator objekta te MBR unutar kojeg se taj objekt nalazi

Indeksiranje prostornih podataka: R-stablo

- Prilikom pretraživanja stabla, minimalni ograničavajući pravokutnici koriste se kod odluke koje će se čvorove pretraživati
  - Značajno se smanjuje broj čvorova koji treba pretražiti

  - R* stabla
  - R+ stabla
  - Hilbetovo R-stablo
  - Prioritetno R-stablo

Indeksiranje prostornih podataka: R-stablo