

Bazy danych przestrzennych

Semestr zimowy 2021/2022
dr inż. Michał Lupa

Plan przedmiotu

- Wprowadzenie do tematyki danych przestrzennych
- Bazy danych przestrzennych:
 - definiowanie danych
 - zapytania SQL
 - rastry
- Procesy ETL dotyczące danych przestrzennych
- Integracja z chmurą
- Serwowanie danych geo

Zaliczenie przedmiotu:

- Każde zajęcia = 1 punkt
- Końcowe kolokwium zaliczeniowe
- Egzamin



**Ocena z ćwiczeń
(średnia arytmetyczna)**

Ocena końcowa: 50% ćwiczenia + 50% egzamin

Wprowadzenie

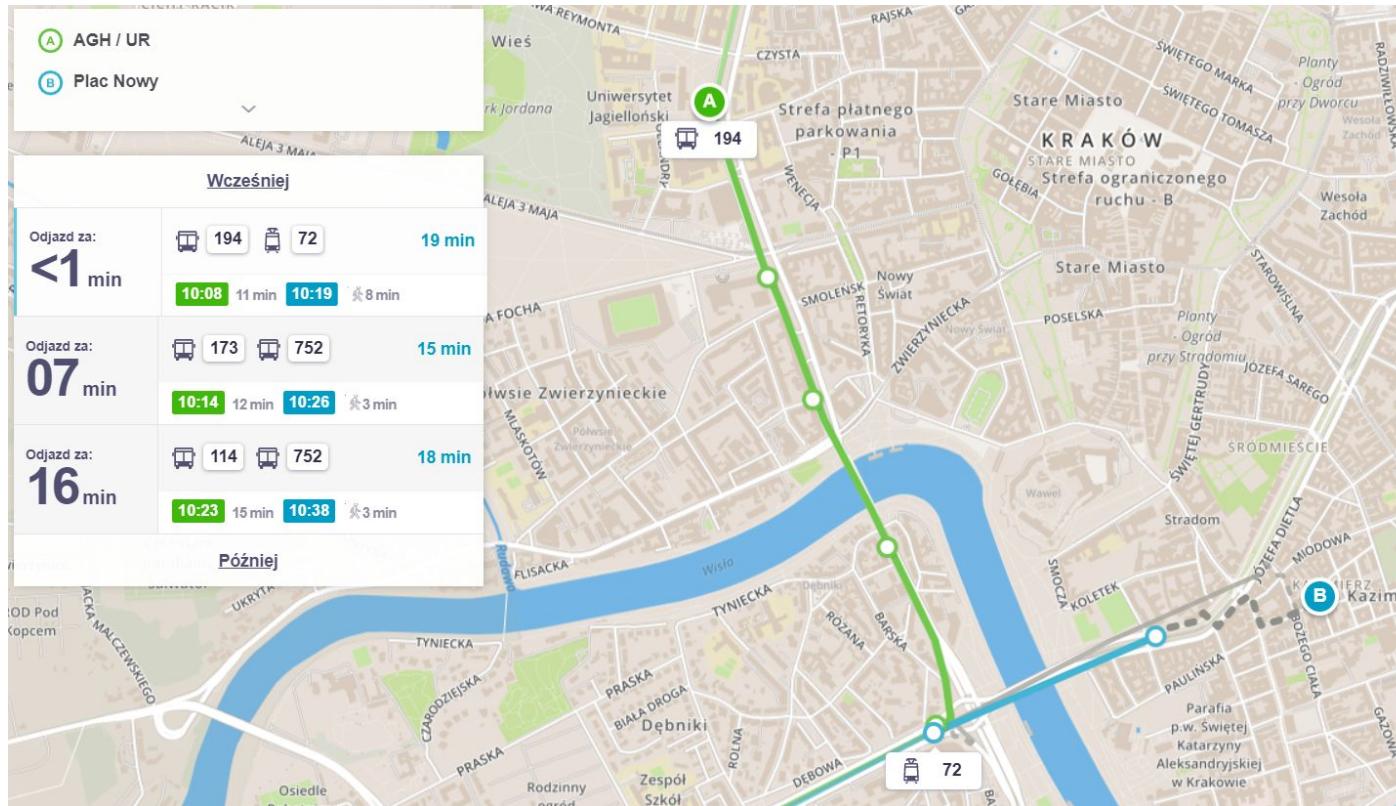


Wprowadzenie

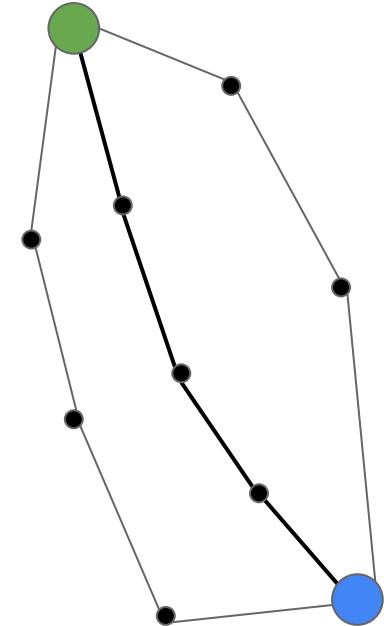
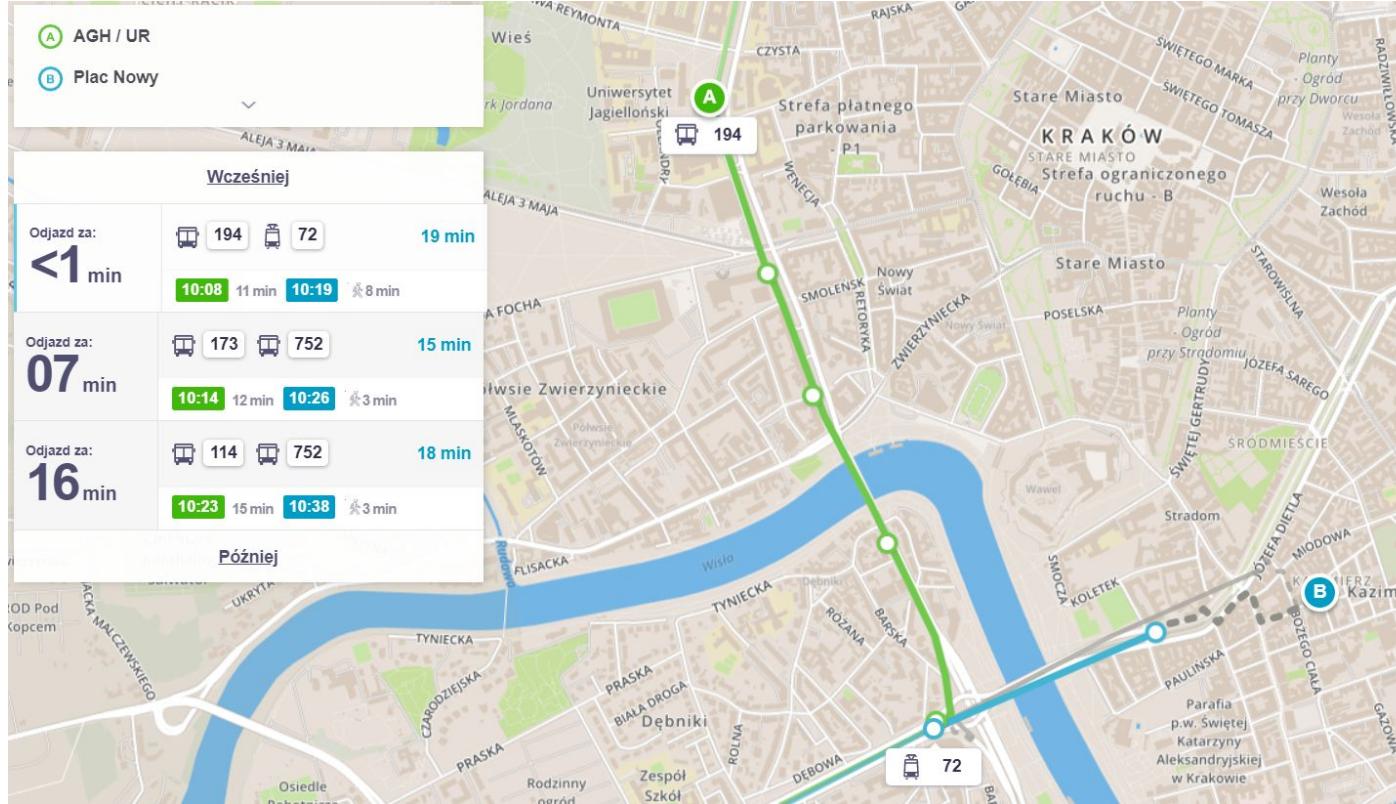


ID	X	Y	Z	R	G	B
4343	14.52	50.20	124.0	94	253	3
4344	14.52	50.21	102.0	100	243	5
4345	14.52	50.23	100.2	190	252	4

Wprowadzenie



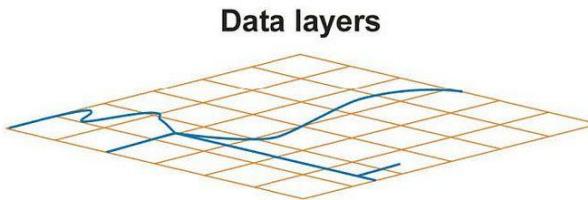
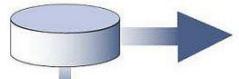
Wprowadzenie



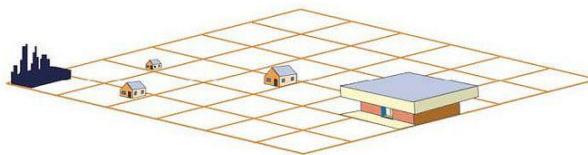
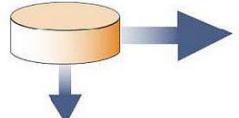
Wprowadzenie

Data source

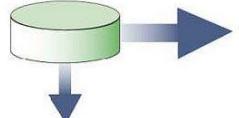
Street data



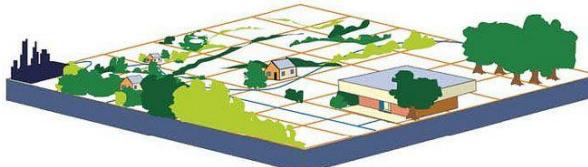
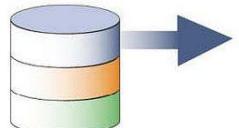
Buildings data



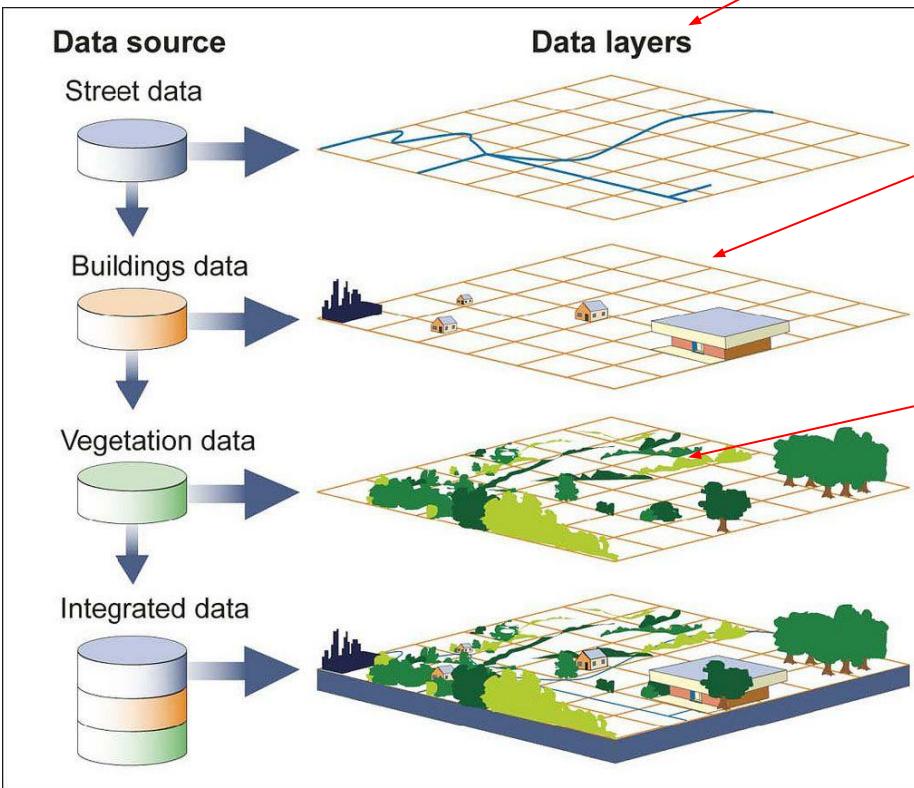
Vegetation data



Integrated data



Wprowadzenie



Streets	
ID	int
Name	varchar
Length	float
Geometry	geometry

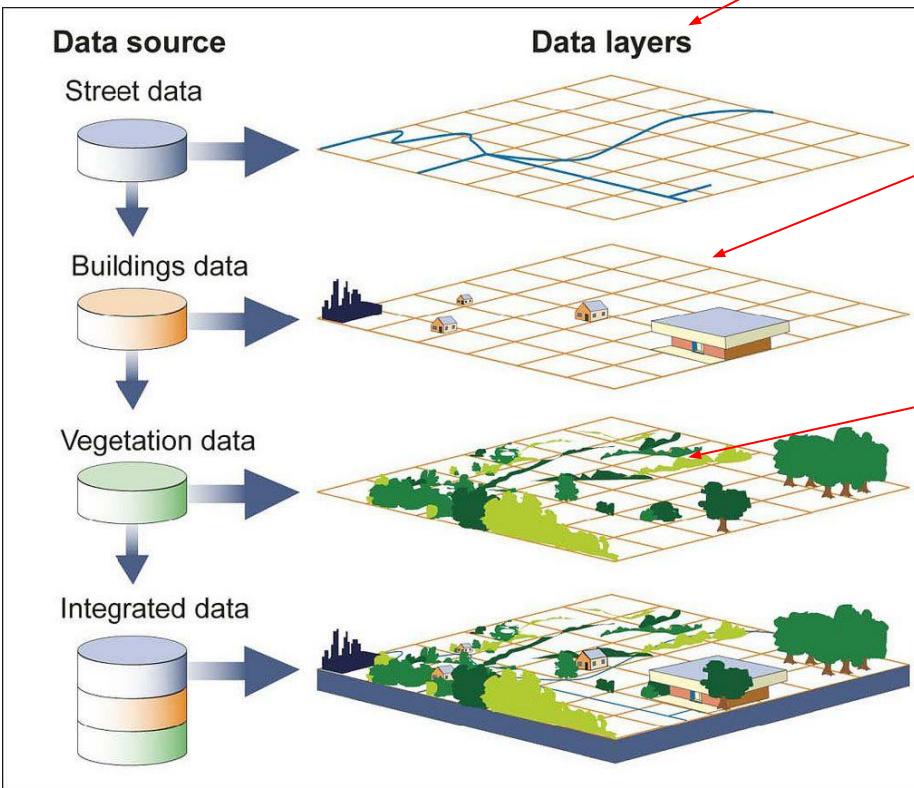
Buildings	
ID	int
Name	varchar
Floors	int
Area	float
Geometry	geometry

Forests	
ID	int
Name	varchar
Area	float
Geometry	geometry

Grasslands	
ID	int
Area	float
Geometry	geometry

Swamps	
ID	int
Area	float
Geometry	geometry

Wprowadzenie



Streets	
ID	int
Name	varchar
Length	float
Geometry	geometry

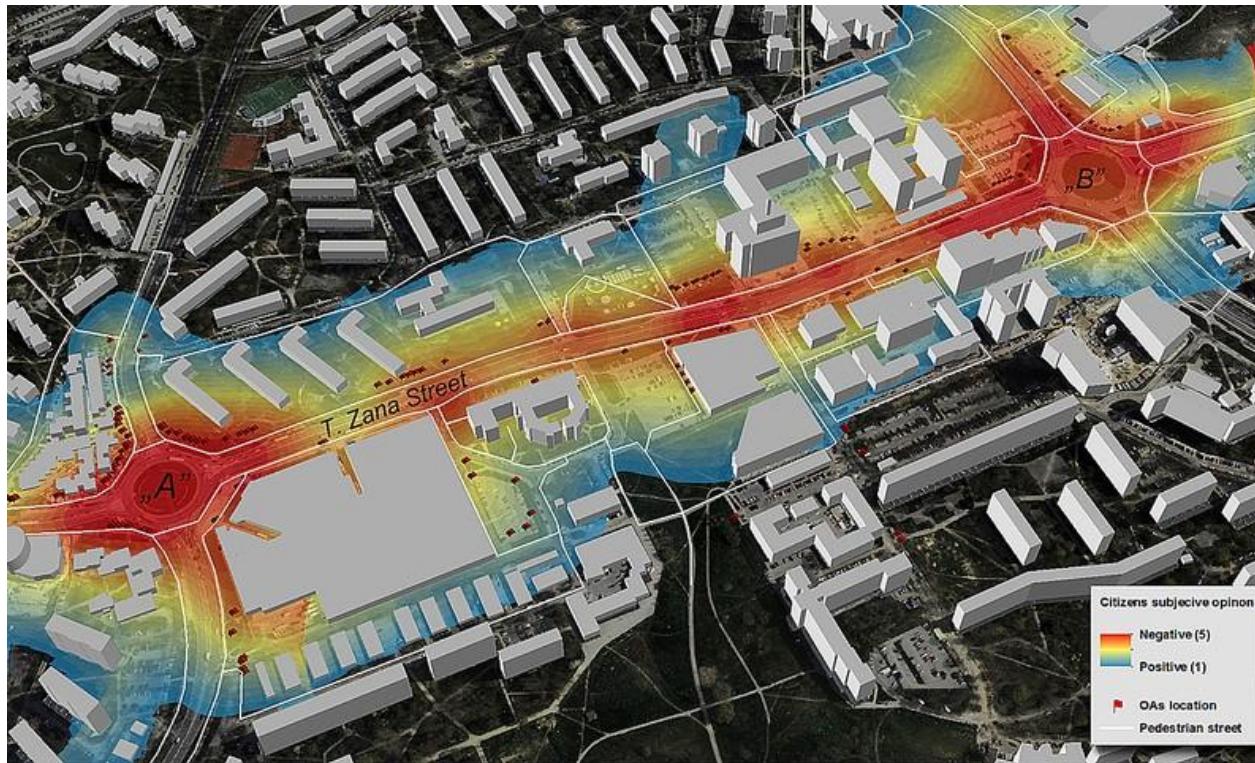
Buildings	
ID	int
Name	varchar
Floors	int
Area	float
Geometry	geometry

Forests	
ID	int
Name	varchar
Area	float
Geometry	geometry

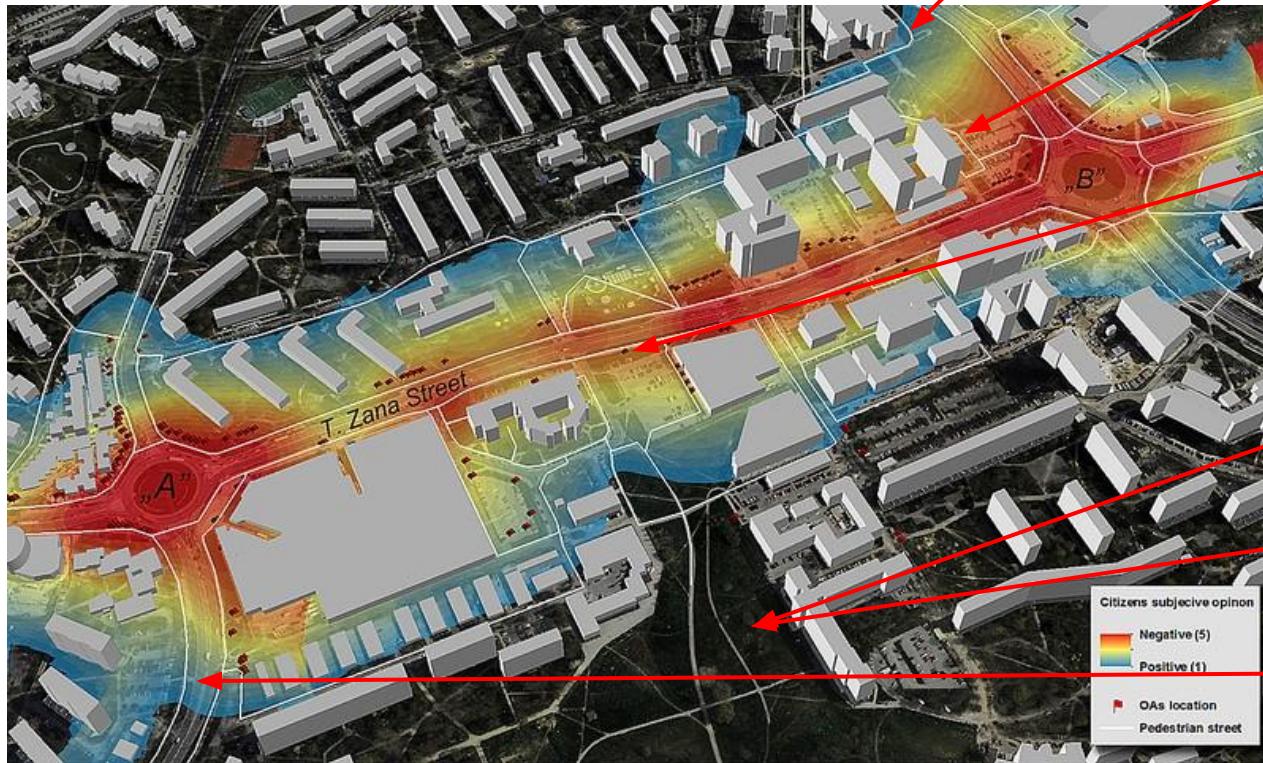
Grasslands	
ID	int
Area	float
Geometry	geometry

Swamps	
ID	int
Area	float
Geometry	geometry

Wprowadzenie



Wprowadzenie



Streets	
ID	int
Name	varchar
Length	float
Geometry	geometry

Buildings	
ID	int
Name	varchar
Floors	int
Area	float
Geometry	geometry

Advertisements	
ID	int
Type	varchar
Geometry	geometry

DEM	
ID	int
Rast	Raster

RGBImagery	
ID	int
Rast	Raster

VisualPollutionAnalysis	
ID	int
Rast	Raster

Streets	
ID	int
Name	varchar
Length	float
Geometry	geometry

GeometryType: Linestring



Buildings	
ID	int
Name	varchar
Area	float
Floors	int
Geometry	geometry

GeometryType: Polygon



Advertisements	
ID	int
Type	float
Geometry	geometry

GeometryType: Point



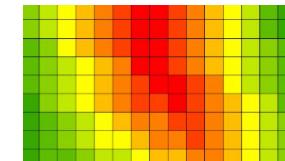
RGBImagery	
ID	int
Rast	Raster

Raster



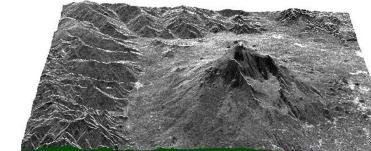
VisualPollutionAnalysis	
ID	int
Rast	Raster

Raster

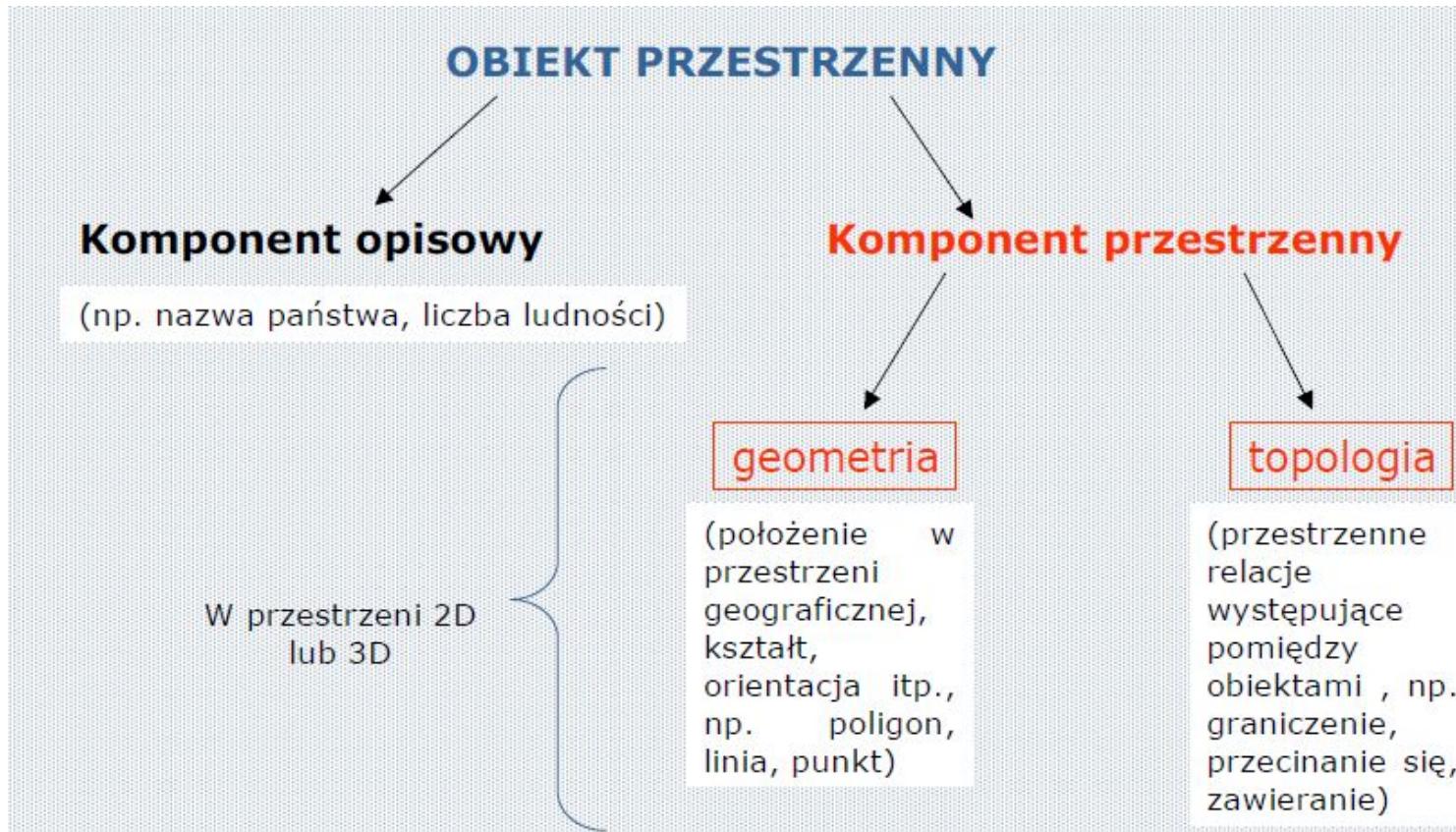


DEM	
ID	int
Rast	Raster

Raster



Dane przestrzenne

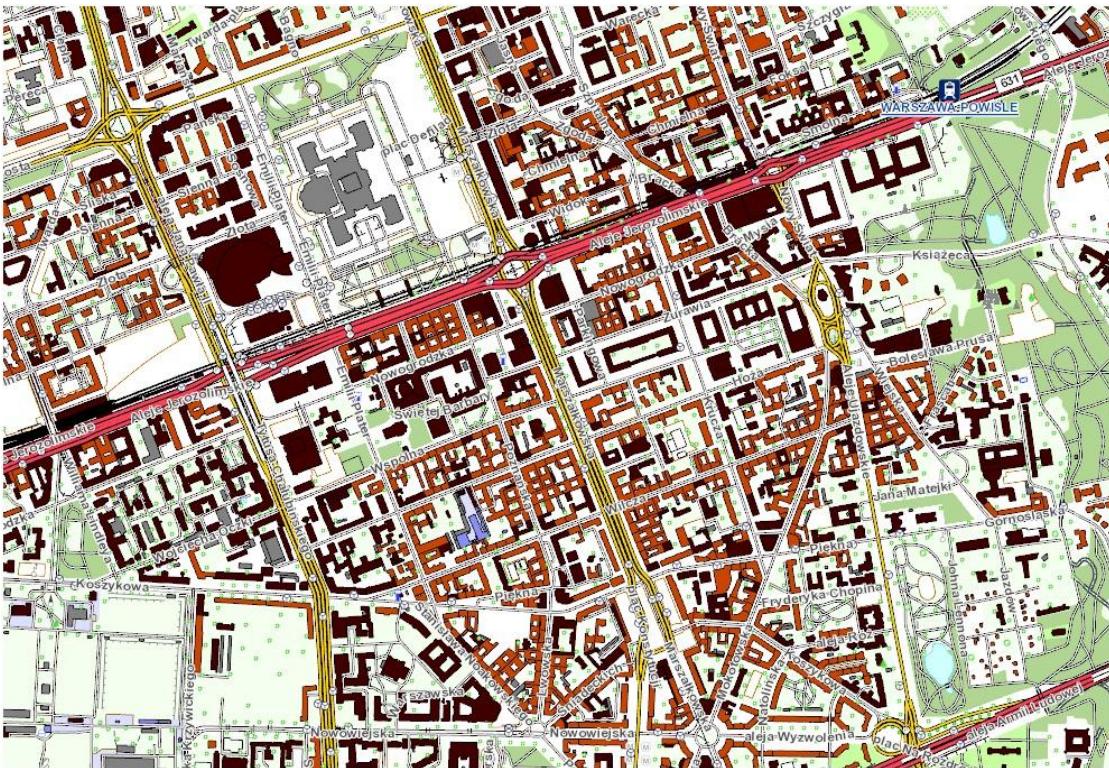


Geometria

Geometria:

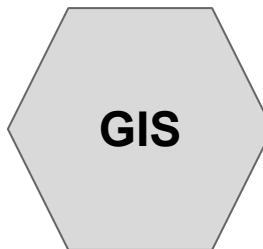
- opisuje położenie w przestrzeni (metrycznej) 2- lub 3-wymiarowej,
- informacje odnośnie położenia dostarczane są za pomocą wybranego układu współrzędnych,
- zaimplementowana z wykorzystaniem wybranego typu danych:
 - **Geometry (Point, LineString, Polygon)**
 - **Raster**

Mapa a dane przestrzenne



- Zbudowana z wielu warstw.
- Każda warstwa odpowiada jednej tabeli w bazie danych.
- W większości przypadków model relacyjny.
- Tabele (warstwy) odpowiadające poszczególnym obiektom mogą, ale nie muszą być połączone relacjami.
- Warstwy podzielone tematycznie (budynki, drogi, lasy itd.).
- Warstwa bazowa obrazująca rzeźbę terenu jest rastrem (Numerycznym Modelem Terenu).

Jak to jest zrobione?



- surowe dane,
- wizualizacja wprost z bazy danych,
- barwy nadane losowo,
- brak etykiet
- linie, krawędzie i punkty o jednakowej szerokości.

- symbolizacja,
- barwy przypisane do zgodnie ze sztuką kartograficzną,
- etykiety,
- linie, krawędzie i punkty o różnej szerokości.

Reprezentacja środowiska a dane

Reprezentacja środowiska przyrodniczego to **różnoskalowa** reprezentacja wybranej części powierzchni Ziemi lub jej najbliższego otoczenia.

Wierna reprezentacja środowiska przyrodniczego nie jest możliwa ze względu na jego **nieskończoną złożoność**. Wybór stopnia szczegółowości reprezentacji środowiska odgrywa kluczowe znaczenie w tworzeniu reprezentacji.

Środowisko przyrodnicze może być reprezentowane przez:

- **obiekty dyskretne**
- **pola (wektorowe, skalarne)**

Środowisko przyrodnicze może być reprezentowane przez



**Obiekty dyskretne
(object-based model)**

Cechy:

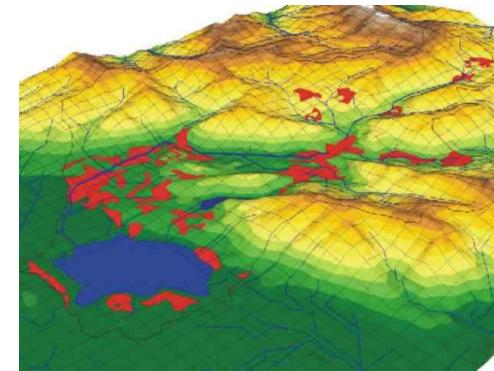
- policzalność,
- zdefiniowane granice,
- rozróżnialne na podstawie rozmiaru,
- należą do jednej z kategorii:
 - punkty
 - linie
 - poligony



**Pola (skalarne, wektorowe;
field-based model)**

Cechy:

- brak wyraźnych granic,
- reprezentacja ciągła wyrażona przez zmienne, których wartości mogą być określone w dowolnym punkcie pola (poprzez funkcję matematyczną),
- mogą być rozróżniane na podstawie stopnia zróżnicowania/wygładzenia



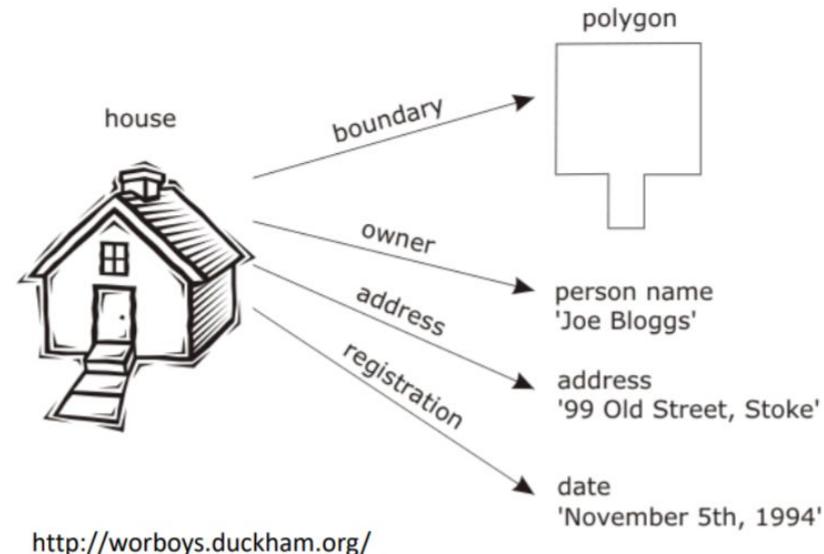
Modele danych przestrzennych

W przypadku obiektów dyskretnych przestrzeń zdekomponowana jest na obiekty (encje).

Encja musi być:

- **rozróżnialna,**
- **możliwa do opisania,**
- **aktualna (o ile to możliwe).**

Zarówno poszczególne encje, jak i zbiory encji (tabele), są zorientowane w wybranym układzie współrzędnych.



Modele danych przestrzennych

Model danych jest uporządkowanym zestawem struktur, które służą do opisu i reprezentacji wybranych **obiektów i procesów** świata rzeczywistego w postaci cyfrowej.

Model danych



Model przestrzenny

Dotyczy struktury w ramach której ujmuje się informacje. Jest to zasadniczo **opis formy lub wyglądu obiektów** i zjawisk przestrzennych.

Przykład: model rastrowy danych

Jest to przybliżony sposób opisu funkcjonowania środowiska przyrodniczego.

Przykład: model erozji

Modele danych przestrzennych

Wybór odpowiedniego modelu danych warunkuje m.in.:

- sposób przechowywania danych (typ **Geometry** vs **Raster**),
- rodzaje analiz, jakie można przeprowadzić,
- sposób wyświetlania informacji (**grafika wektorowa** vs **grafika rastrowa**),
- rodzaje zależności pomiędzy obiektami (relacje topologiczne).

Modele danych przestrzennych

Wybór odpowiedniego modelu danych warunkuje m.in.:

- sposób przechowywania danych (typ **Geometry** vs **Raster**),
- rodzaje analiz, jakie można przeprowadzić,
- sposób wyświetlania informacji (**grafika wektorowa** vs **grafika rastrowa**),
- rodzaje zależności pomiędzy obiektami (relacje topologiczne).

W projektach dotyczących danych przestrzennych wybór odpowiedniego modelu danych ma kluczowe znaczenie.

Modele danych przestrzennych

Wybór odpowiedniego modelu danych warunkuje m.in.:

- sposób przechowywania danych (typ **Geometry vs Raster**),
- rodzaje analiz, jakie można przeprowadzić,
- sposób wyświetlania informacji (**grafika wektorowa vs grafika rastrowa**),
- rodzaje zależności pomiędzy obiektami (relacje topologiczne).

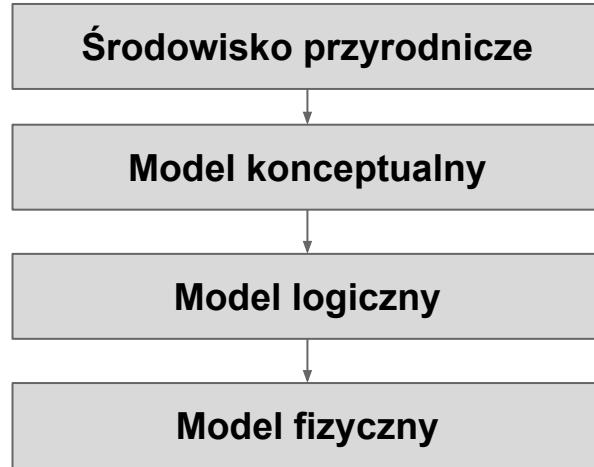
W projektach dotyczących danych przestrzennych wybór odpowiedniego modelu danych ma kluczowe znaczenie.

Od przyjętego modelu świata rzeczywistego zależy bowiem stopień dokładności i przydatności otrzymanych wyników.

Wybór często podyktowany jest dostępem do danych i możliwościami ich pozyskania.

Modele danych przestrzennych

Wzrastający poziom abstrakcji



Rozpoznanie źródeł danych.

Wybór sposobu reprezentacji wybranych cech środowiska (modele dyskretne vs pola).

Model wybranych obiektów oraz ich powiązań.
Przeważnie zapis zgodny z **diagramem ER**.

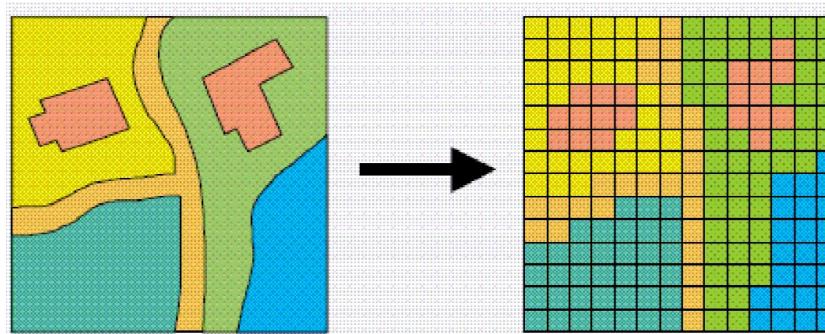
Model relacyjny, relacyjno-obiektowy, powstały na bazie diagramu ER.

Precyjnie opisane relacje oraz związki pomiędzy relacjami.

Zdefiniowane operacje, które można wykonać na danych.

Model rastrowy

Rastrowy model danych wykorzystuje do reprezentacji obiektów **macierz elementów** (pikseli). W komórkach rastra przechowywane są wartości atrybutów.



Każdy raster posiada metadane, które zawierają informacje o:

- współrzędnych geograficznych górnego lewego narożnika siatki,
- wielkości piksela,
- liczbie pikseli w wierszach i kolumnach,
- układzie współrzędnych.

Model rastrowy

Cechy rastra:

- rozdzielcość przestrzenna - długość i szerokość pikseli,
- rozdzielcość radiometryczna - mówi o tym, jak szczegółowo mogą być rozróżniane na obrazie różnice w jasnościach. Satelitarne zobrazowanie 8-bitowe ma 256 odcienni szarości, zaś 16-bitowe (satelity ERS) z kolei 65.536.
- rozdzielcość spektralna - liczba kanałów spektralnych.

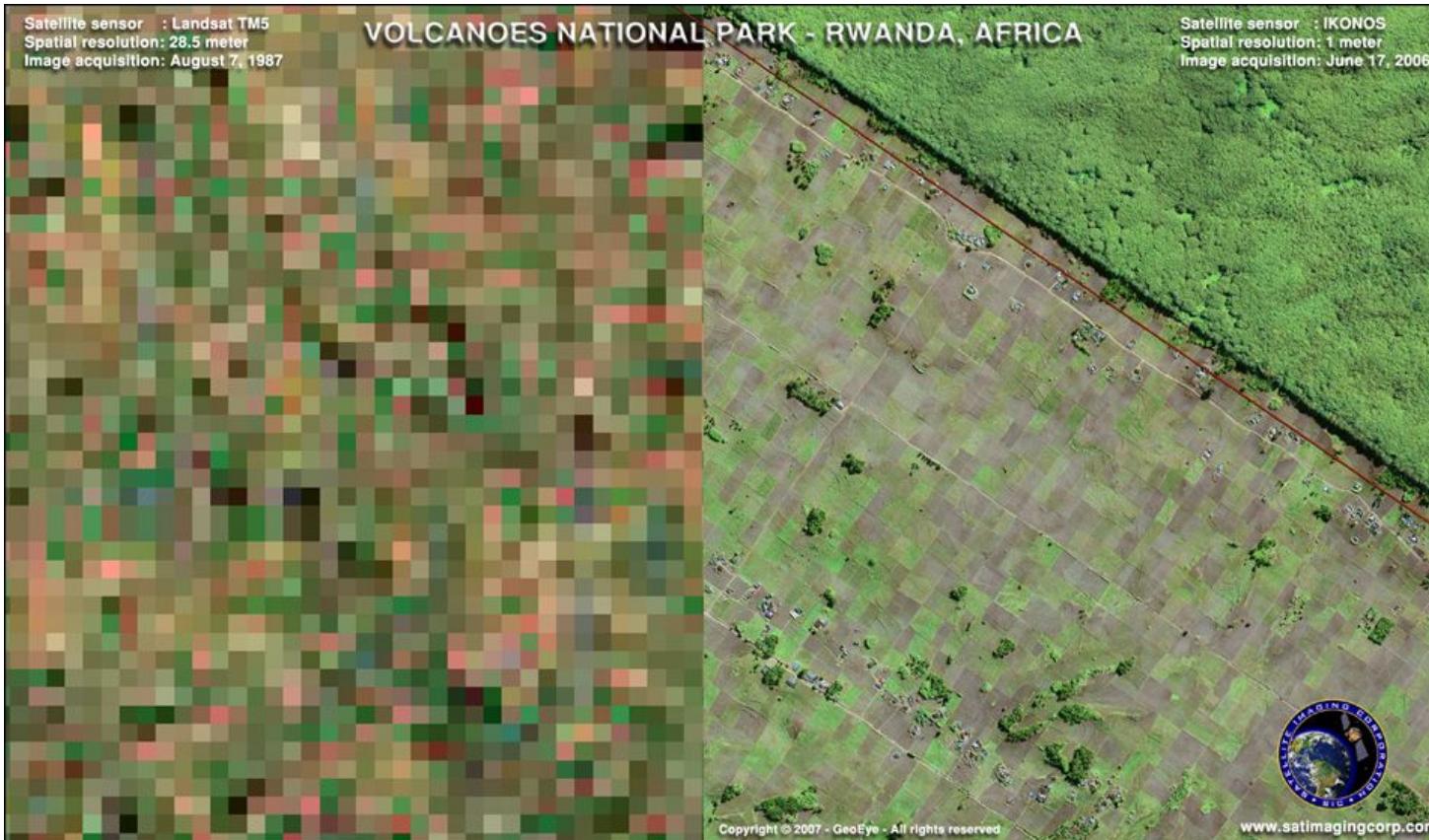
➤ **Operacje:** digitalizacja, podstawowe statystyki, kompozycje barwne, analizy statystyczne i geostatystyczne, klasyfikacje..

➤ Rastry zazwyczaj używane są jako **podkład** do analiz.

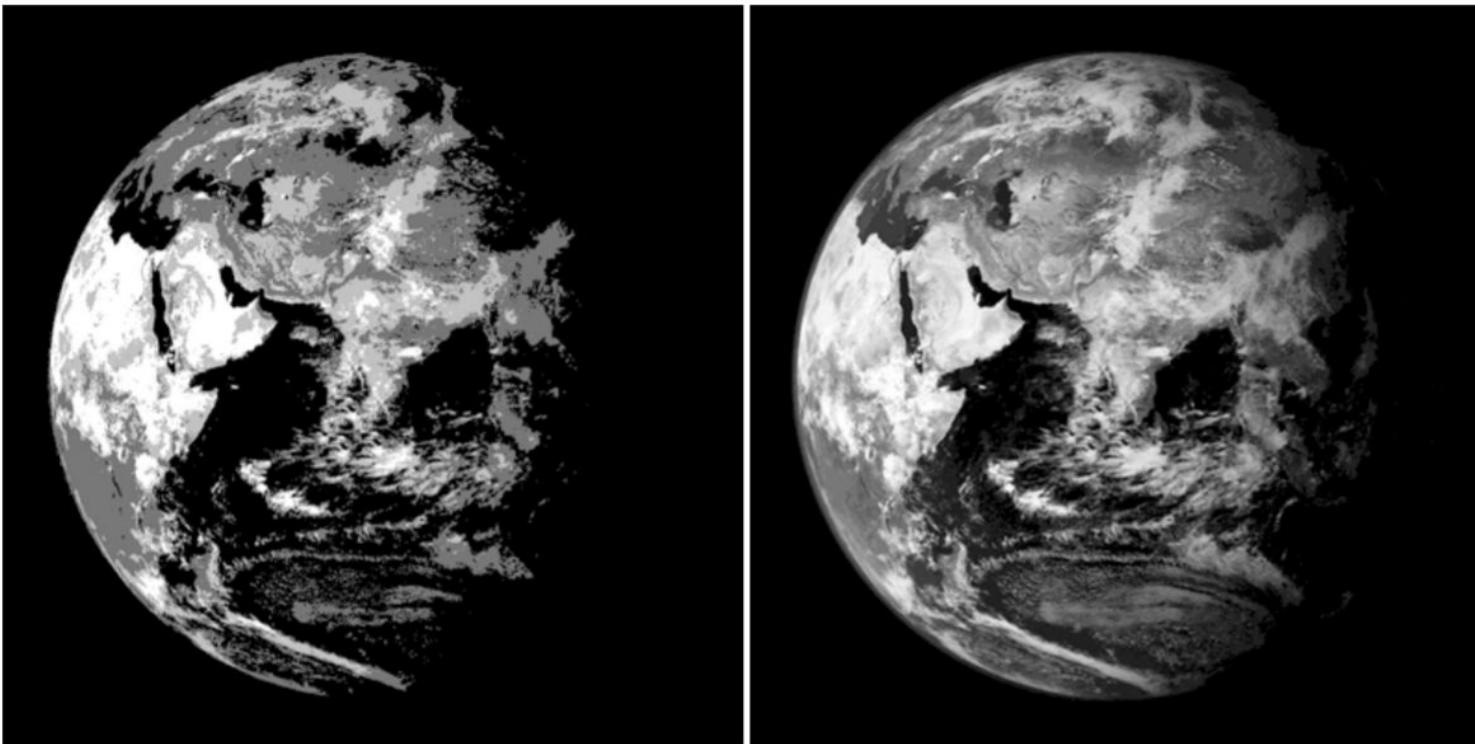
➤ O użyteczności w projekcie GIS decyduje **rozdzielcość**.

Rastrowy model danych jest najczęściej związany z **koncepcjonalnym modelem pól ciągłych**.

Rozdzielcość przestrzenna



Rozdzielcość radiometryczna



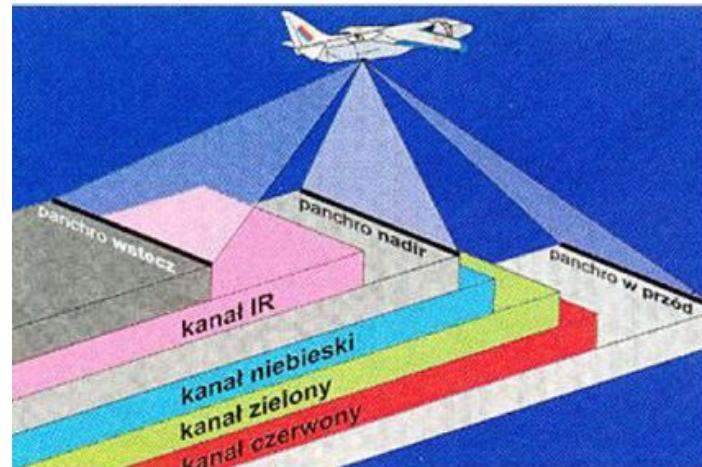
Rozdzielcość spektralna

Kanał spektralny – określony (wąski) zakres spektrum elektromagnetycznego rejestrowany jako pojedynczy obraz.

Obrazowanie wielospektralne – jednoczesna rejestracja wielu zakresów spektralnych (dla każdego zakresu powstaje osobny obraz).

Rozdzielcości spektralne:

- Wielospektralne (LANDSAT)
- Superspektralne (ASTER)
- Hiperspektralne (HyMap)
- Ultraspektralne (HYPERION)



Materiał panchromatyczny



B

G

R



Model wektorowy

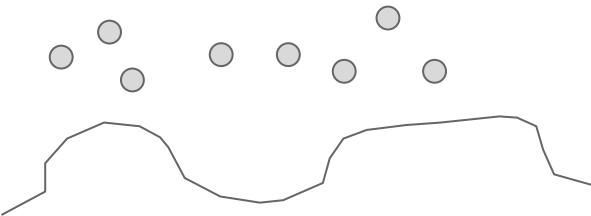
Wektorowy model danych dzielimy na:

- model prosty,
- model złożony - topologiczny.

Model wektorowy prosty

Elementami prostego modelu wektorowego są:

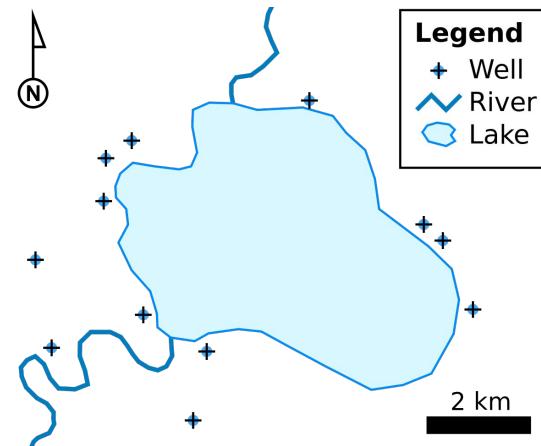
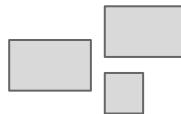
- punkty,



- linie,



- poligony.

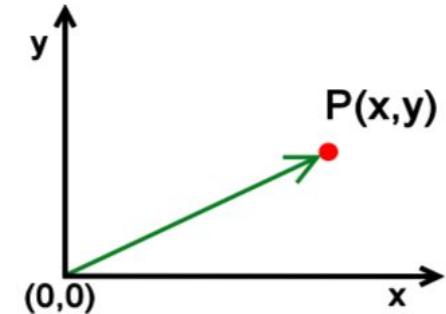


Każdy obiekt definiowany jest niezależnie od innych obiektów.

Model wektorowy prosty

Punkty:

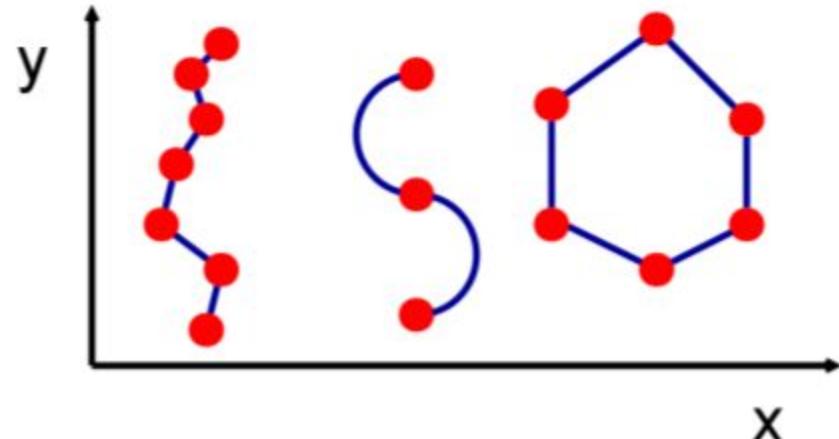
- są zero-wymiarowe
- zdefiniowane przez współrzędne (X, Y) w 2D oraz (X, Y, Z) w 3D,
- ogólna definicja punktu: **POINT(X Y)**



Model wektorowy prosty

Linie:

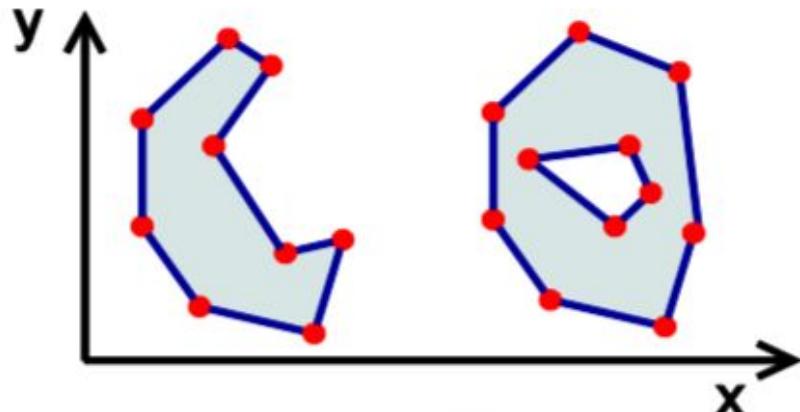
- segment linii to odcinek zdefiniowany przez dwa punkty $(X_1 Y_1)$ $(X_2 Y_2)$,
- linia złożona jest ze stycznych w punkcie segmentów linii,
- ogólna definicja linii: **LINESTRING(0 0, 1 1, 2 1, 2 2)**,
- linie są 1-wymiarowe.



Model wektorowy prosty

Poligony:

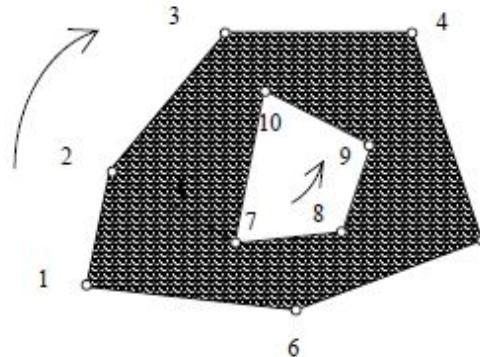
- są liniami (łamanyimi) zamkniętymi,
- mogą zawierać pierścienie wewnętrzne,
- ogólna definicja: **POLYGON(0 0, 1 1, 2 1, 2 2, 3 3, 0 0)**,
- poligony są 2-wymiarowe.



Model wektorowy prosty

Poligony z pierścieniami:

- w sytuacji złożonych obiektów powierzchniowych pojawia się problem ich opisu za pomocą jednego ciągu punktów,
- w celu rozwiązania tego problemu tworzy się dodatkowe (fikcyjne) poligony w obrębie obiektu.

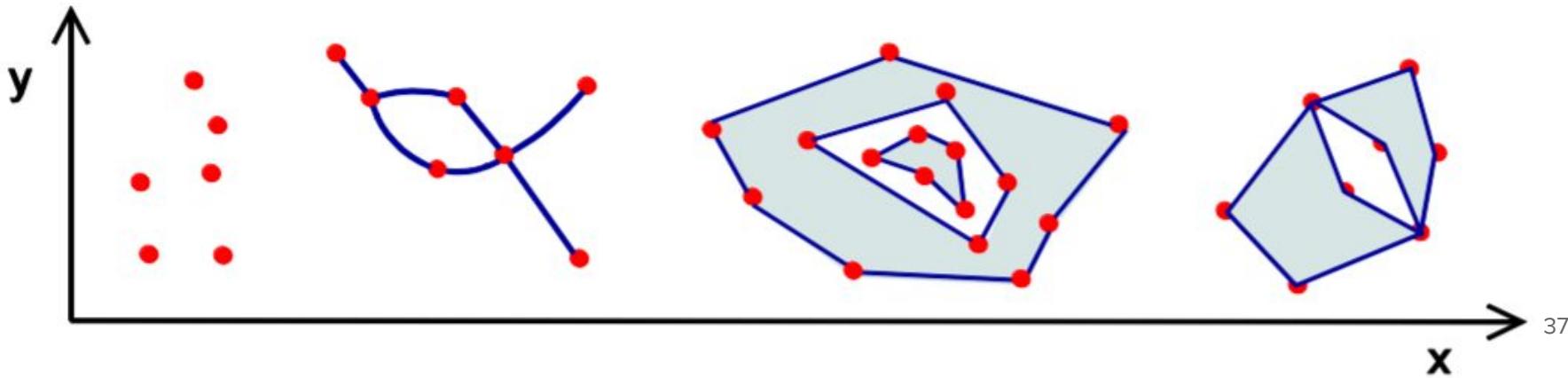


$$C(P_1, P_2, P_7, P_8, P_9, P_{10}, P_7, P_2, P_3, P_4, P_5, P_6, P_1)$$

Model wektorowy prosty

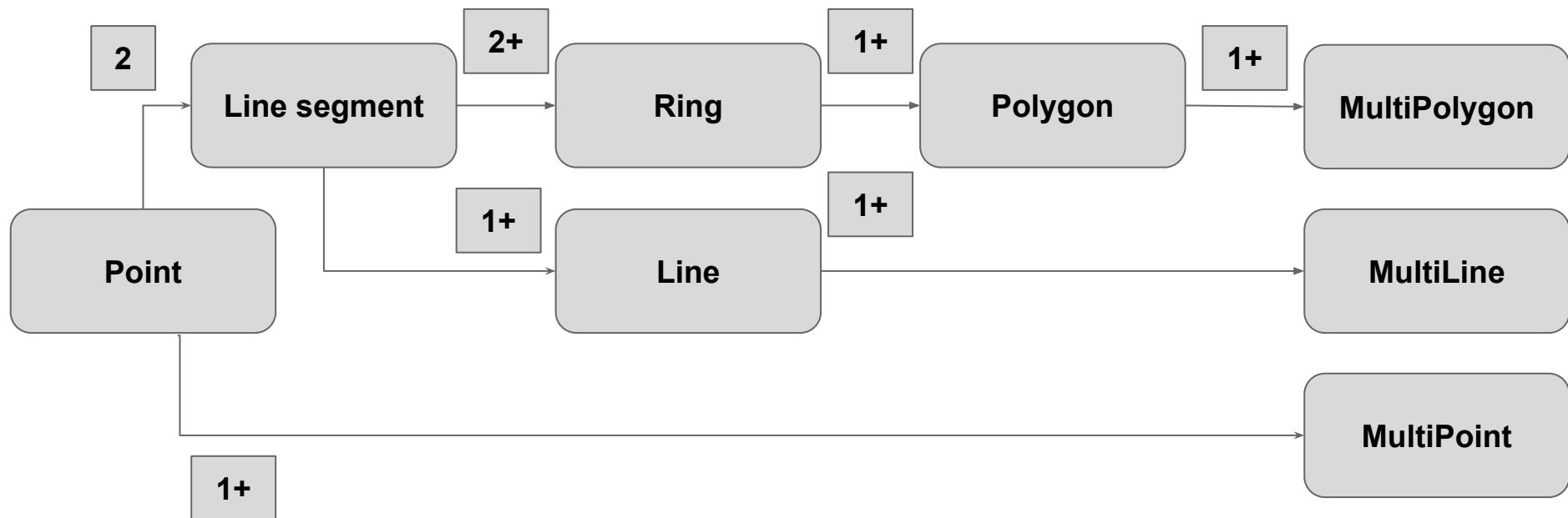
Kolekcje geometrii:

- definicja wielu obiektów geometrycznych tego samego typu jako jeden obiekt,
- przykładowa definicja: MULTIPOLY((0 0), (1 1))
- ich właściwości są zgodne z typem geometrii pojedynczego obiektu,
- dopuszczalne kolekcje różnych typów geometrii (np. punkty i poligony).



Model wektorowy prosty

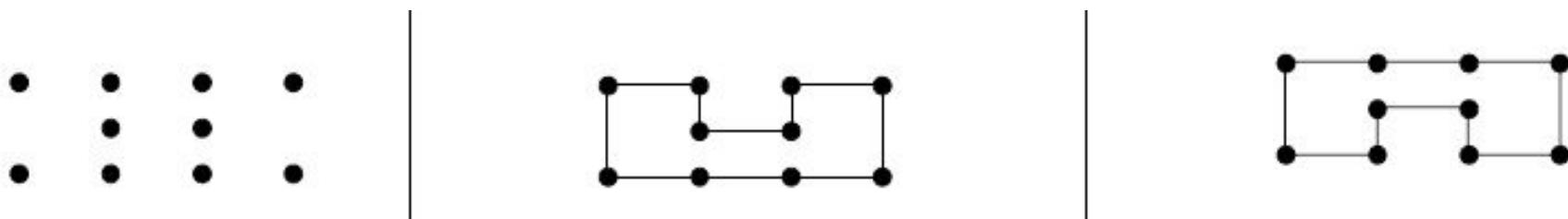
Diagram związku poszczególnych klas geometrii:



Model wektorowy prosty

Cechy modelu wektorowego prostego:

- idealny do reprezentowania obiektów dyskretnych,
- **względnie** małe wymagania dotyczące pamięci,
- teoretycznie nieskończona precyzja,
- łatwe wprowadzanie do bazy (obiekty są niezależne),
- istotne jest uporządkowanie (kolejność wprowadzania współrzędnych).



Model wektorowy prosty

Wady:

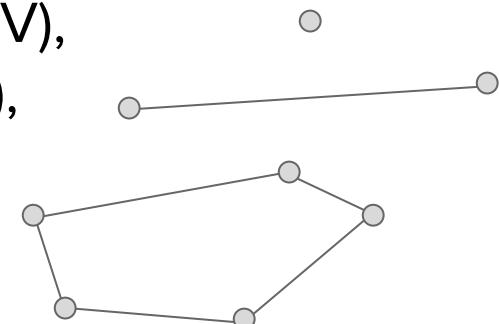
- brak informacji o topologii - celu wykrycia związków przestrzennych między obiektami konieczne jest wykorzystanie metod geometrii analitycznej,
- redundancja danych,
- ryzyko niezgodności danych,
- brak ograniczeń dotyczących położenia obiektów np. brak punktów przecięcia dla przecinających się poligonów czy linii.

Model wektorowy złożony

Przestrzeń metryczna implikuje **przestrzeń topologiczną**, tj. możemy określić relacje topologiczne pomiędzy obiektami, jeżeli znana jest ich geometria.

Topologiczny model wektorowy wyróżnia trzy typy obiektów topologicznych:

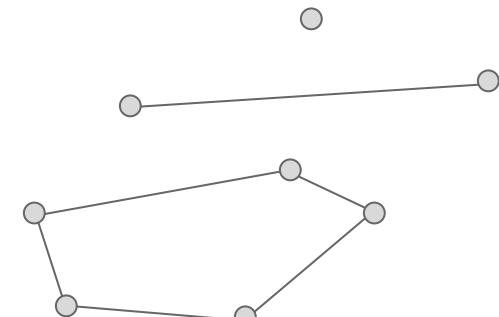
- zerowymiarowe (punkty węzłowe; ang. vertex - V),
- jednowymiarowe (linie graniczne; ang. edge - E),
- dwuwymiarowe (obszary; ang. face, F).



Model wektorowy złożony

Trzy typy obiektów:

- zerowymiarowe (punkty węzlowe; ang. vertex - V),
- jednowymiarowe (linie graniczne; ang. edge - E),
- dwuwymiarowe (obszary; ang. face, F).

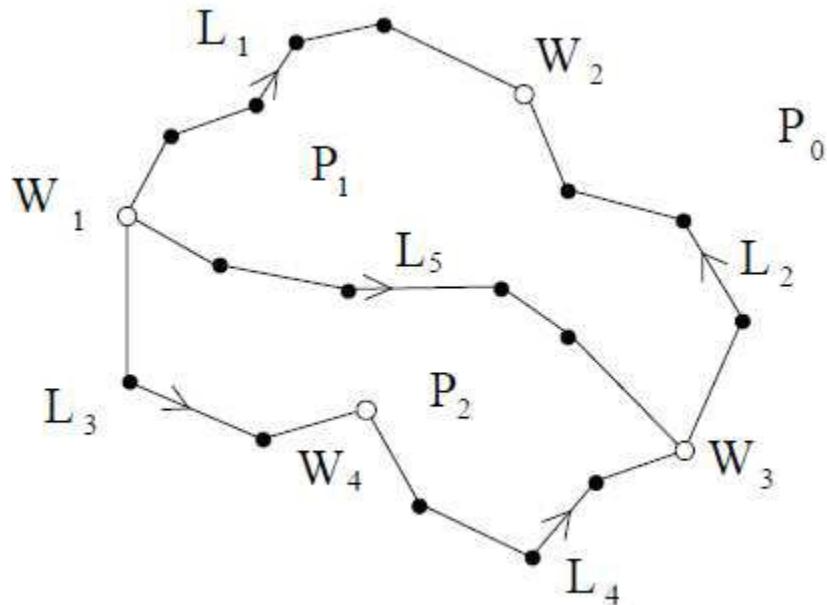


Dla wszystkich wymienionych typów obiektów możemy zapisać wzajemne relacje.

Elementy klasy wyższej, budowane są zawsze z elementów klasy niższej.

Model wektorowy złożony

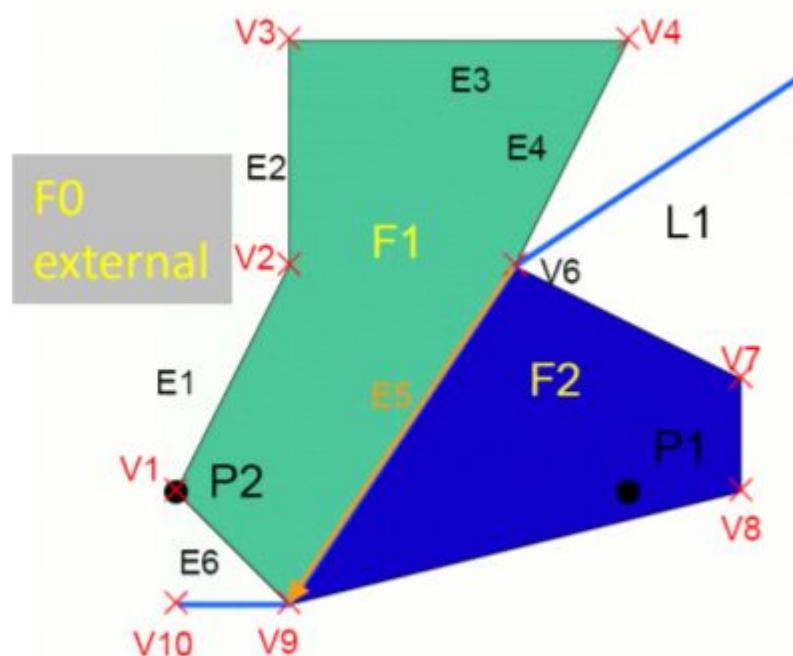
Przypisywanie relacji topologicznych do elementów zerowymiarowych.



	kolejne pary ($L_g P_p$)
W_1	$L_1 P_1, L_5 P_2, L_3 P_0$
W_2	$-L_1 P_0, -L_2 P_1$
W_3	$L_2 P_0, -L_5 P_1, -L_4 P_2$
W_4	$L_4 P_0, -L_3 P_2$

Model wektorowy złożony

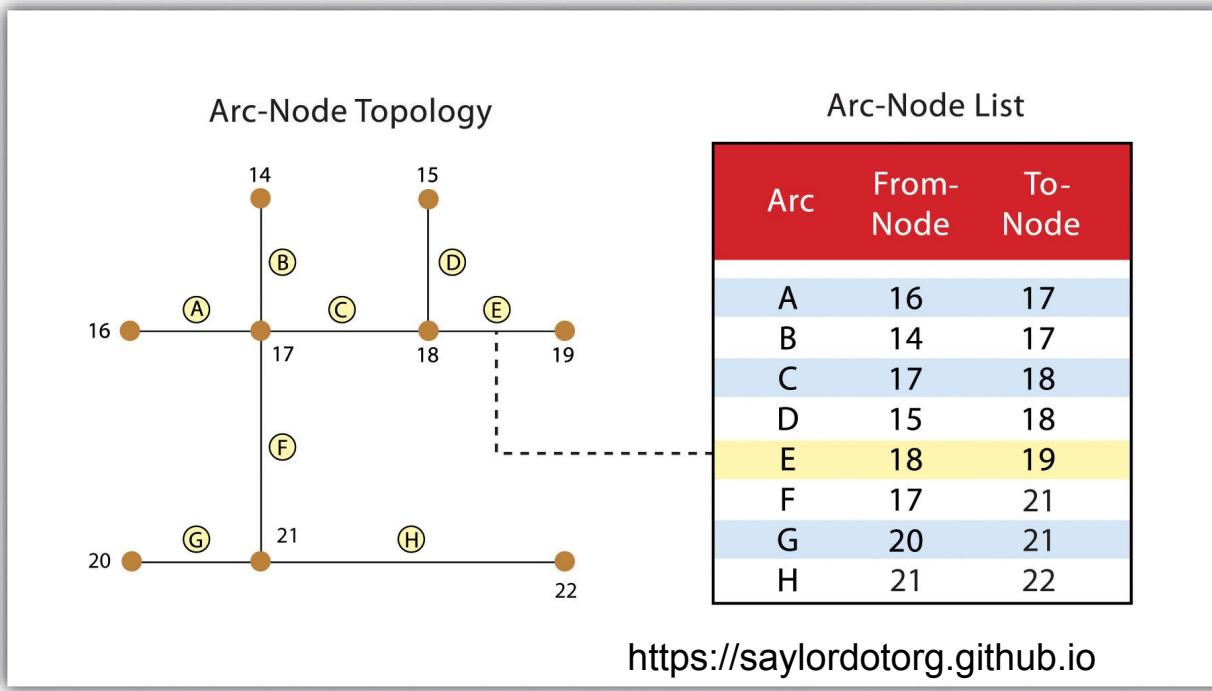
Arc-node topology: relacje topologiczne pomiędzy punktami i liniami są przechowywane jawnie.



edge	start node	end node	right face	left face
E1	V1	V2	F1	F0
E2	V2	V3	F1	F0
E3	V3	V4	F1	F0
E4	V4	V6	F1	F0
E5	V6	V9	F1	F2
E6	V9	V1	F1	F0

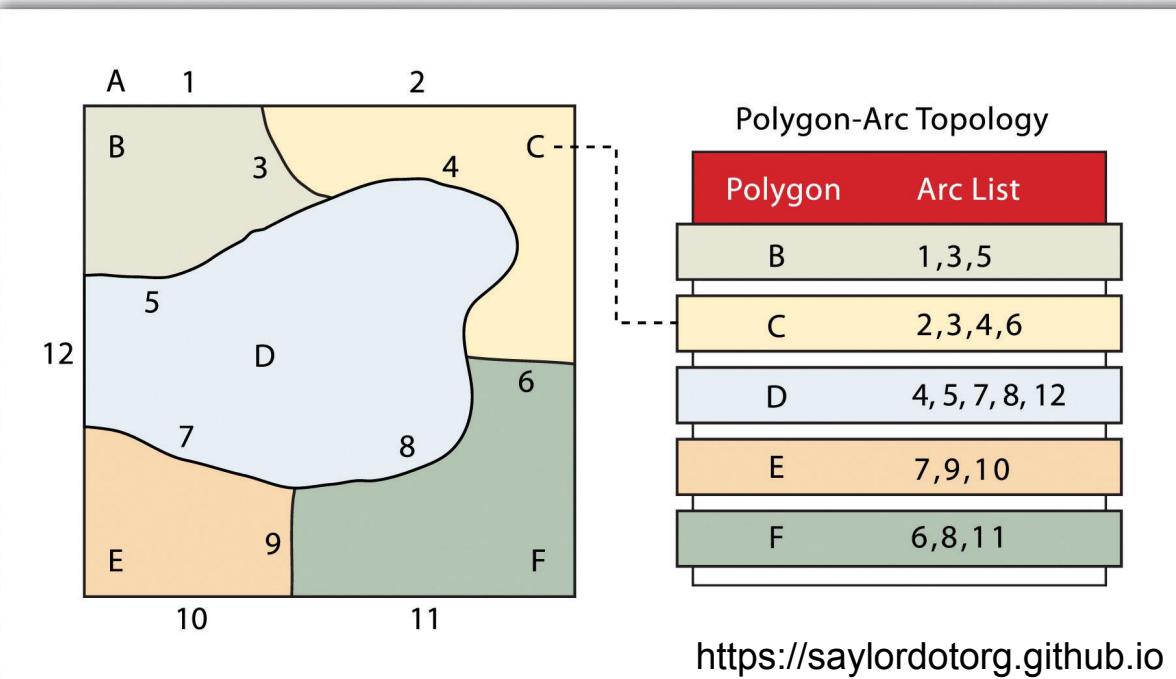
Model wektorowy złożony

Arc-node topology: relacje topologiczne pomiędzy punktami i liniami są przechowywane jawnie.



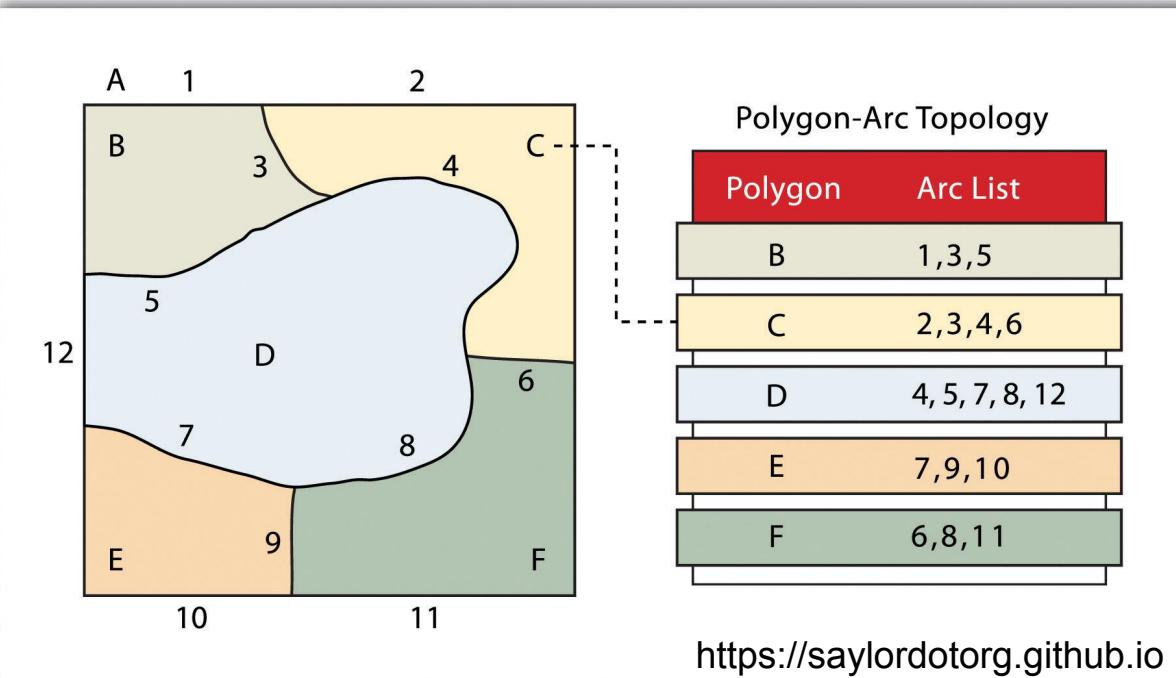
Model wektorowy złożony

Area definition (polygon-arc topology): krawędzie definiują poligony. Każda krawędź jest przechowywana tylko raz.

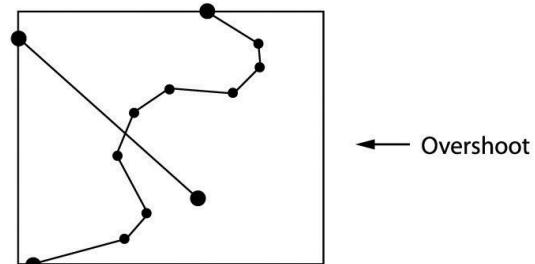
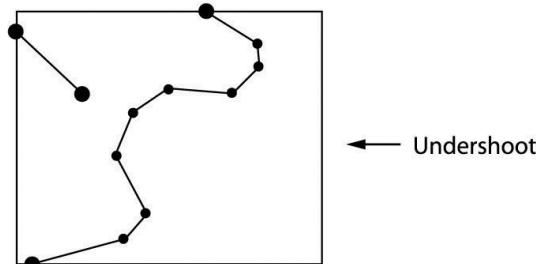
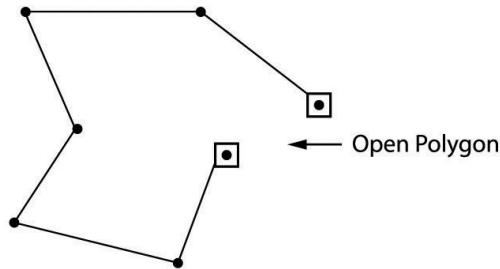


Model wektorowy złożony

Contiguity (polygon topology). Każda krawędź ma kierunek. Poligony o wspólnej krawędzi są styczne krawędziowo.



Typowe błędy topologiczne (silvery)



Model wektorowy złożony

Relacje topologiczne spełniają funkcję **więzów integralnościowych** dla baz danych przestrzennych (oraz map):

- każda krawędź ma dwa styczne wierzchołki,
- każda krawędź ma dwa przyległe obszary,
- każdy obszar opisany jest na przemian krawędziami i wierzchołkami,
- każdy wierzchołek jest na przemian opisany krawędziami i obszarami,
- krawędzie się nie przecinają.

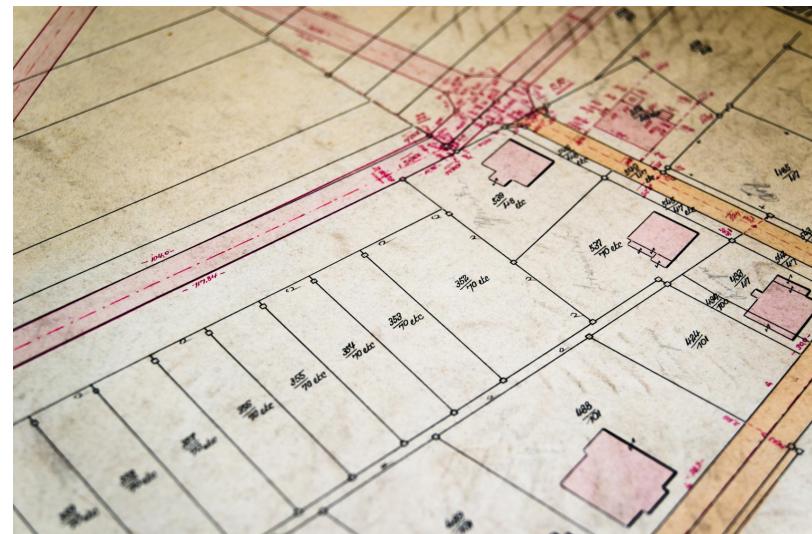
Model wektorowy złożony

Te więzy integralnościowe **nie zawsze jednak się sprawdzają.**

Świetnie nadają się do problemów **katastru nieruchomości.**

Bywają natomiast problematyczne:

- dla źródeł,
- kanałów,
- ślepych zaułków.



Ogólne zastosowania modelu topo

Sprawdzanie poprawności danych na podstawie:

- spójności sieci (czy wszystkie elementy sieci są połączone),
- przecięcia linii (czy polilinie mają węzły w miejscach przecięć),
- częściowego nakładania się wieloboków (czy sąsiadujące wieloboki częściowo się nakładają),
- istnienia duplikatów linii (czy istnieją dokładnie pokrywające się elementy sieci lub wieloboki).

Wydajność edycji danych (np.: funkcja śledzenia, automatyczne zamykanie wieloboku, dociąganie, przemieszczanie elastyczne).

Optymalizacja zapytań (śledzenie sieci, sąsiedztwo wieloboków, zawieranie, przecinanie się).

Model sieciowy

Model sieciowy danych to bardzo często stosowana w GIS odmiana modelu topologicznego.

Modele sieci składają się z **węzłów i krawędzi**. Relacje topologiczne precyzyjnie określają, w jaki sposób linie łączą się ze sobą w węzłach. Istotne jest **oznaczenie drogi przepływu** przez sieć. Tempo przepływu modelowane jest jako obciążenie w węzłach i na liniach.

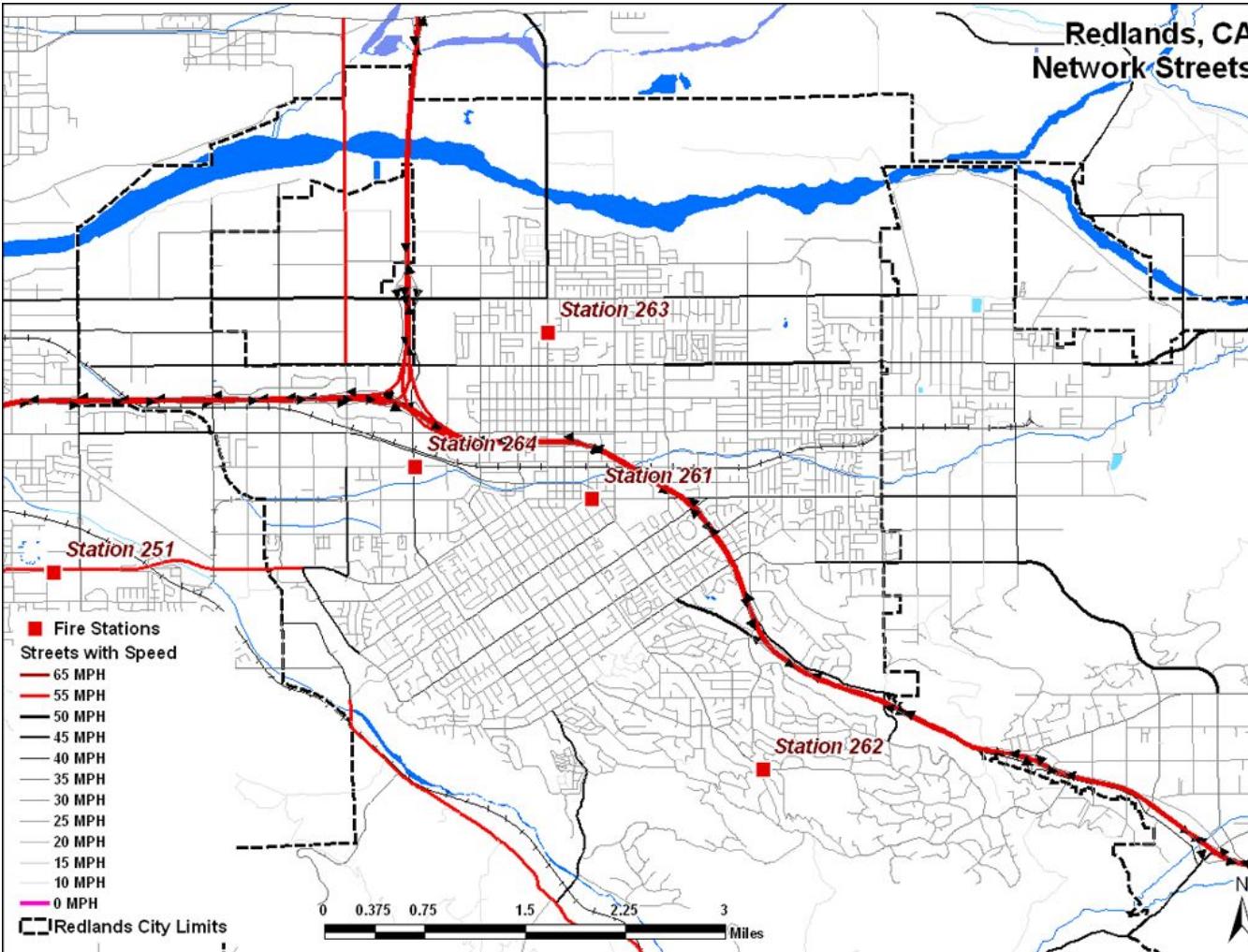
Typy sieci:

- **Promieniowa** (dendrytyczna) – określony kierunek przepływu np. system rzeczny;
- **Pętlowa** – pojawiają się punkty przecięcia np. sieci wodociągowe;

Model sieci składa się z:

- Punktów (skrzyżowanie ulic, bezpieczniki, zawory wody itp.)
- Linii (ulice, linie przemysłowe, przewody itp.)

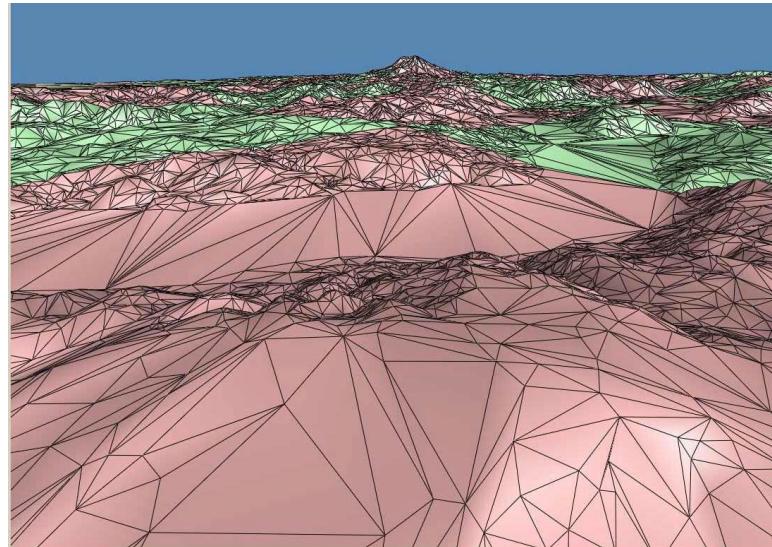
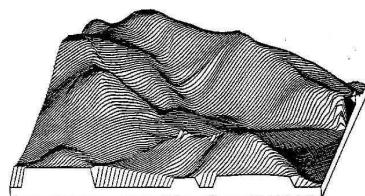
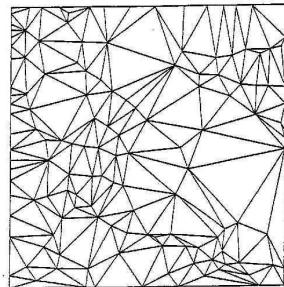
Redlands, CA Network Streets



Model TIN

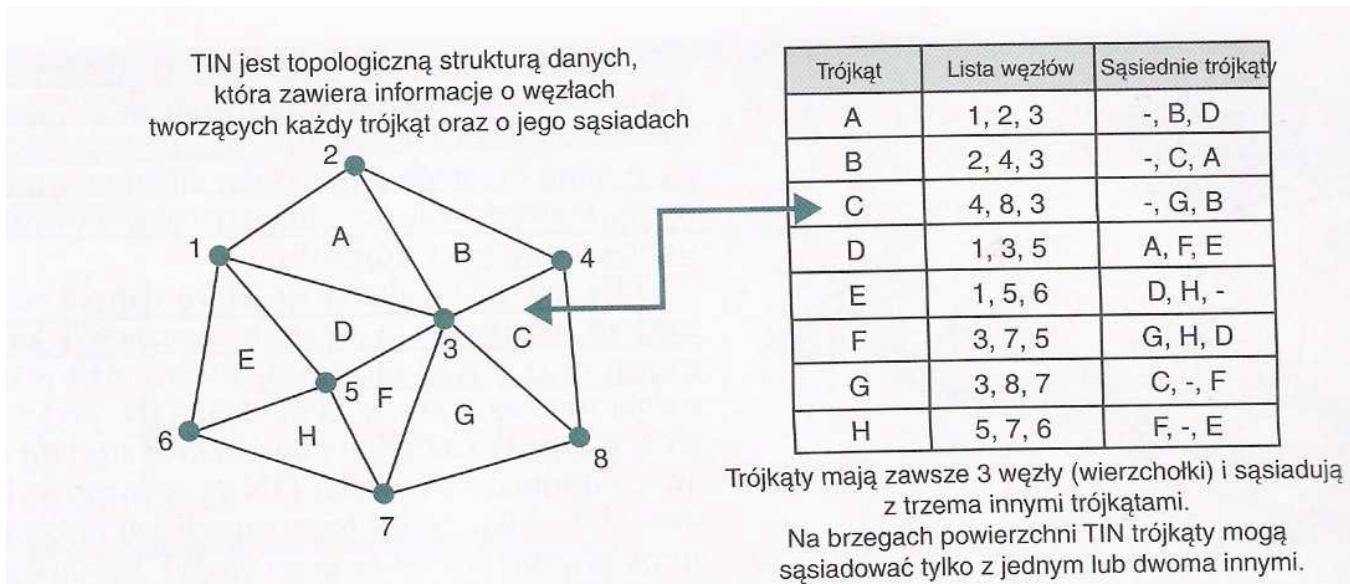
TIN (ang. Triangulated Irregular Network) służy do reprezentacji powierzchni (model 2.5 D), za pomocą sąsiadujących ze sobą i nie pokrywających się trójkątów.

TIN tworzy się na podstawie zbioru punktów o znanych współrzędnych x, y, z.



Model TIN

Powierzchnie TIN mogą być tworzone w wyniku przeprowadzenia triangulacji Delaunaya.



TIN jest **topologiczną strukturą danych**, która zawiera informacje o węzłach tworzących każdy trójkąt oraz o jego sąsiadach.

Model TIN

Zalety:

- Dobra reprezentacja powierzchni o znaczącej zmienności rzeźby.
- Ułatwia obliczenie: wysokości, spadku, ekspozycji i linii widoku między punktami.

Wady:

- W przypadku występowania wartości ekstremalnie odstających brak możliwości wygładzenia oryginalnych danych.
- Jakość zależy od danych wejściowych.
- Niemożność odwzorowania nieciągłości rzeźby terenu.
- Konieczność zwiększenia liczby danych dotyczących szczytów, obniżeń, grzbietów i dolin jeżeli zależy nam na wygenerowaniu dokładnej reprezentacji.

Literatura

Rigaux P., Scholl M., Voisard A. Spatial Databases with application to GIS. Morgan Kaufmann Publishers

Longley P.A., Goodchild M.F., Maguire D.J., Rhind D.W. 2006 GIS. Teoria i Praktyka. PWN SA, Warszawa

Zeiler M. 1999 Modeling Our World: The ESRI Guide to Geodatabase Design. Redlands, CA: ESRI Press

Baza danych - przypomnienie

“A database is a collection of interrelated data items that are managed as a single unit”

A. J. Oppel, Databases: A Beginner’s Guide

“Bazą danych nazywa się utrwalony zbiór danych, opisujących pewien fragment rzeczywistości i przeznaczonych do wspólnego wykorzystania przez różne programy”

Z. Jurkiewicz, MIMUW

“A database is a collection of structured data. A database captures an abstract representation of the domain of an application”

C. A. Curino, MIT

Baza danych przestrzennych

“A **spatial database** is a database that **is enhanced** to **store** and **access** spatial data or data that defines a geometric space.” *Techopedia*.

“(...) database system that offers **spatial data types** in its **data model** and **query language** and supports spatial data types in its implementation, providing at least **spatial indexing and spatial join methods**.”

An introduction to Spatial Database Systems. R. H. Güting.

Baza danych przestrzennych – baza danych zoptymalizowana do składowania i odpytywania danych powiązanych z obiektami w przestrzeni, takimi jak: punkty, linie i wielokąty. *Wiki*

Rys historyczny

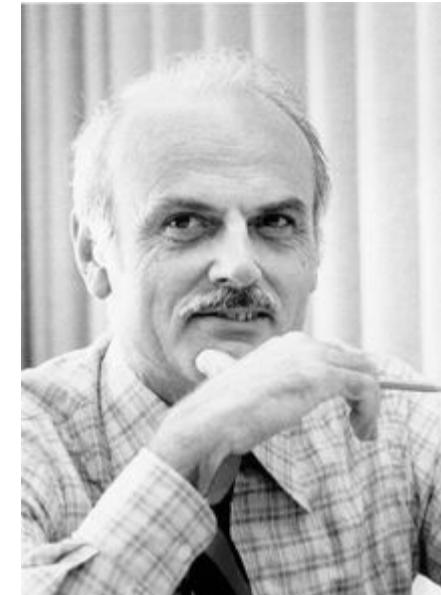
Edgar F. (Ted) Codd

W 1970 roku zaproponował relacyjny model baz danych.

IBM prowadziło projekt badawczy o nazwie System R.
Podczas jego realizacji opracowano SQL.

W 1979 firma Relational Software Inc. (dziś Oracle) opublikowała pierwszą, komercyjną wersję SQL.

SQL został uznany za standard przez ANSI w 1986 roku.



Rys historyczny

Od lat 70 XX wieku wiele agencji/firm tworzyło własne rozwiązania w zakresie Systemów Informacji Geograficznej (GIS).

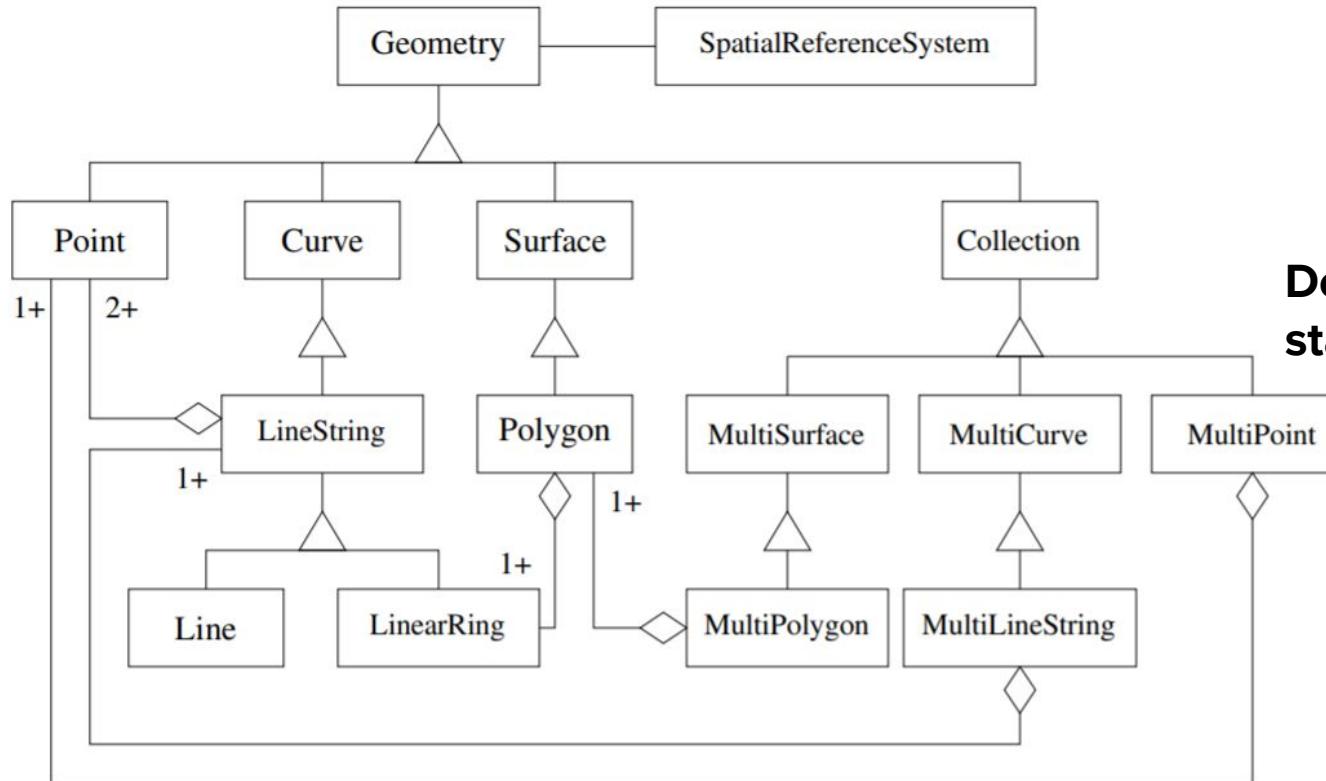
Początki lat 80 to powstanie GRASS - rastrowego systemu GIS na potrzeby armii USA.

W 1994 powstało Open Geospatial Consortium, tworząc dokument "Open Geodata Interoperability Specification" (OGIS).

W 1999 OGC wydało OpenGIS Simple Features Specification For SQL w wersji 1.1 (ostatnia aktualizacja - wersja 1.2.1 - rok 2010). Zdefiniowano **"Geometry model"**.

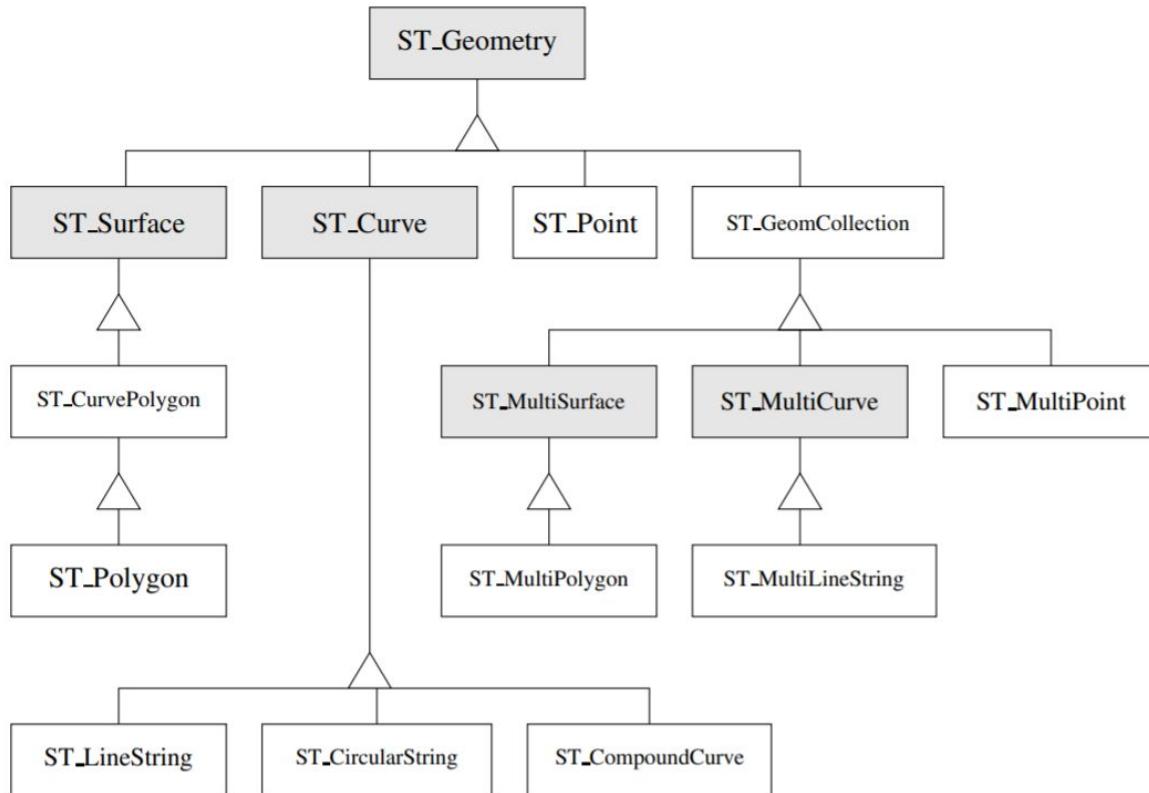
Na bazie standardu OGC, ISO w 2002 wprowadziło standard dotyczący SQL: ISO/IEC 13249-3 SQL/MM Part 3: Spatial. **Standard ten określa sposób definiowania i dostępu do danych przestrzennych za pomocą SQL.**

Typy danych przestrzennnych



**Definicja według
standardu OGC**

Typy danych przestrzennych



**Definicja według
standardu ISO**

Typy danych przestrzennych

Standardowe typy dla DBMS są rozszerzone o nowy typ danych: **Geometry**.

Geometry jest typem bazowym, a jego trzema podstawowymi podtypami:

- **ST_Point**: punkt określony przez współrzędne X, Y,
- **ST_LineString**: polilinia złożona z wielu punktów X, Y,
- **ST_Polygon**: poligon/powierzchnia - łamana zamknięta tworzona przez punkty o współrzędnych X, Y.

Wybrane funkcje geoprzestrzenne wg ISO

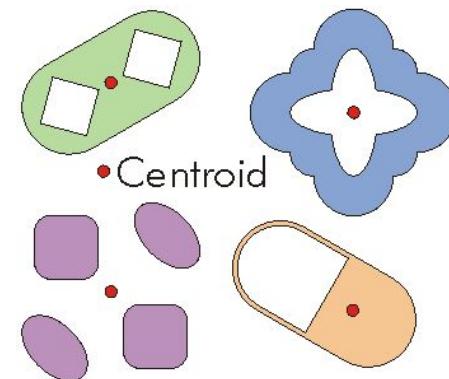
ST_Length() - zwraca wartość typu double, stanowiącą długość polilinii,
ST_Area(), **ST_Perimeter()** - analogicznie, pole powierzchni, obwód.

ST_Distance() - zwraca odległość euklidesową między punktem A i B.

ST_Contains() - zwraca wartość TRUE, jeżeli obiekt A zawiera obiekt B.

ST_Centroid() zwraca obiekt **ST_Point()**

będący centroidem obiektu.



Wybrane DBMS implementujące standard

- Oracle (Oracle Spatial),
- MS SQL Server,
- MySQL,
- PostgreSQL (PostGIS),
- IBM DB2 Spatial Extender,
- SQLite (SpatiaLite),

NoSQL:

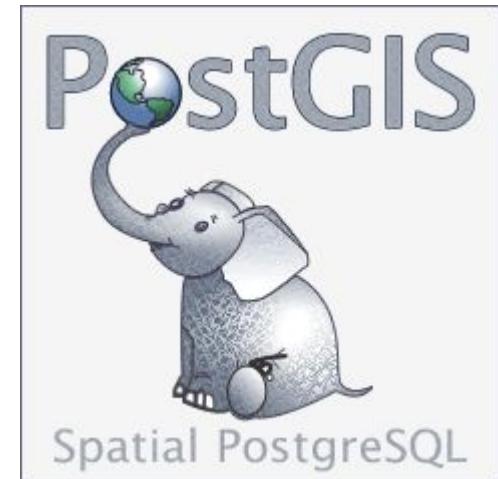
- Kinota
- **MongoDB wspiera dane przestrzenne, ale nie jest zgodne z OGC.**

PostGIS

PostGIS to rozszerzenie **PostgreSQL**, rozszerzające DBMS o możliwość pracy z danymi przestrzennymi. PostGIS jest zaimplementowany zgodnie ze **standardem OGC/ISO**.

PostGIS umożliwia m.in.:

- definiowanie i składowanie danych przestrzennych,
- korzystanie z ponad kilkuset dodatkowych funkcji,
- tworzenie prostej topologii,
- kontrolę poprawności danych przestrzennych,
- transformację układów współrzędnych.



PostGIS

PostGIS umożliwia pracę z trzema typami danych przestrzennych:

- **geometry** (układ odniesienia to płaszczyzna),
 - **geography** (układ odniesienia to sferoida, współrzędne podajemy w stopniach, minutach i sekundach kątowych),
 - **raster** - dedykowany obiektom rastrowym.
- } wektory

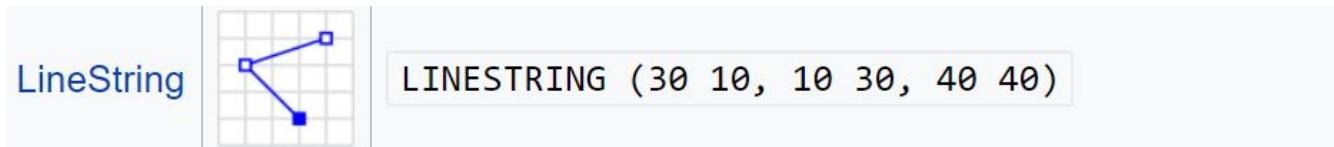
Geometryczne (wektorowe) typy danych obsługują:

- A. zbiór obiektów prostych 2D („*Simple Features*”) zdefiniowanych przez OGC;
- B. zbiór obiektów zdefiniowanych w układach 3DZ, 3DM i 4D;

WKB/WKT

Specyfikacja OGC pozwala na definicję obiektów przestrzennych z wykorzystaniem:

- formy tekstowej **WKT** (*Well-Known Text*):



- formy binarnej **WKB** (*Well-Known Binary*):

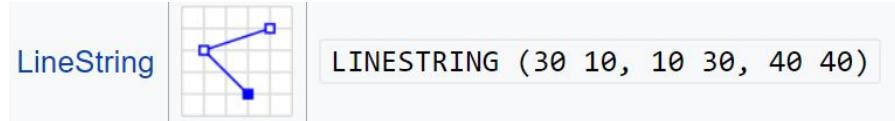
POINT(2.0 4.0) = ,

- 1-byte integer or 0: big endian
- 4-byte integer or 1: POINT (2D)
- 8-byte float or 2.0: x-coordinate
- 8-byte float or 4.0: y-coordinate

WKB/WKT

WKB/WKT zawierają informacje na temat:

- typu obiektu (np. punkt, poligon),
- współrzędnych obiektu.



WKB/WKT nie zawiera informacji na temat układu współrzędnych!

- podczas **INSERT'ów** musimy dodatkowo wskazać numer SRID,
- numer SRID (ang. Spatial Referencing System Identifier), jednoznacznie identyfikuje układ współrzędnych (SRS - Spatial Reference System),
- dla układu WGS84 (GPS) SRID = 4326, zaś SRID = 0 wskazuje na brak SRS.

WKB/WKT

bytea **ST_AsBinary(geometry);**

text **ST_AsText(geometry);**

Nie zwracają informacji o SRID

st astext

```
POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))  
(1 row)
```

WKB/WKT

```
geometry ST_GeomFromWKB(WKB, SRID);
geometry ST_GeomFromText(WKT, SRID);
integer ST_SRID(geometry);
```

```
INSERT INTO app(p_id, the_geom)
VALUES (2, ST_GeomFromText('POINT (-71.060316 48.432044)', 4326));

SELECT ST_SRID(ST_GeomFromText('POINT (-71.1043 42.315)', 4326));
--result
4326
```

WKB/WKT

```
geometry ST_GeomFromWKB(WKB, SRID);  
geometry ST_GeomFromText(WKT, SRID);  


---

integer ST_SRID(geometry);
```

Definicja SRID

```
INSERT INTO app(p_id, the_geom)  
VALUES (2, ST_GeomFromText('POINT (-71.060316 48.432044)', 4326));

---


```

```
SELECT ST_SRID(ST_GeomFromText('POINT (-71.1043 42.315)', 4326));  
--result  
4326
```

EWKB/EWKT

Obiekty przestrzenne obsługiwane przez PostGIS:

- A. zbiór obiektów prostych 2D („*Simple Features*”) zdefiniowanych przez *OGC*;
- B. zbiór obiektów zdefiniowanych w układach 3DZ, 3DM i 4D;

- Definiowane za pomocą **EWKT** (Extended WKT) lub **EWKB** (Extended WKB).
- Połączony zapis z informacją o SRID.
- Możliwość wystąpienia błędów.

3DZ : (X Y Z-Wysokość n.p.m.)

3DM : (X Y M-Pomierzony parametr)

4D : (X Y Z M)

EWKB/EWKT

```
bytea ST_AsEWKB(geometry);  
text ST_AsEWKT(geometry);
```

Zwracają informację o SRID!

st asewkt

```
SRID=4326; POLYGON( (0 0, 0 1, 1 1, 1 0, 0 0))  
(1 row)
```

EWKB/EWKT

geometry **ST_GeomFromEWKB(EWKB);**
geometry **ST_GeomFromEWKT(EWKT);**

```
INSERT INTO app(p_id, the_geom)
VALUES (3, ST_GeomFromEWKT('SRID=4269;POINT(-71.064544 42.28787 220.28787)'));
```



```
SELECT ST_GeomFromEWKT('SRID=4269;MULTILINESTRING((-71.160281 42.258729,-71.160837
42.259113,-71.161144 42.25932))');
```

Metadane

„Simple Features Specification for SQL” definiuje:

- typy danych przestrzennych,
- funkcje niezbędne do zarządzania danymi,
- metadane.

Mamy dwie tabele metadanych:

- SPATIAL_REF_SYS
- GEOMETRY_COLUMNS

SPATIAL_REF_SYS

- SRID (np. 4326-WGS84)
- AUTH_NAME
- AUTH_SRID
- SRTEXT
- PROJ4TEXT

GEOMETRY_COLUMNS

- F_TABLE_CATALOG
- F_TABLE_SCHEMA
- F_TABLE_NAME
- F_GEOMETRY_COLUMN
- COORD_DIMENSION
- SRID
- TYPE

SPATIAL_REF_SYS

Tabela **SPATIAL_REF_SYS** zawiera informacje o ponad 3000 najczęściej wykorzystywanych układach współrzędnych. Zawiera również szczegóły umożliwiające transformację jednego układu współrzędnych w inny.

Często wykorzystywane układy to:

- WGS84 Long Lat (SRID=4326)
- WGS84 World Mercator (SRID=3395)

```
CREATE TABLE spatial_ref_sys (
    srid INTEGER NOT NULL PRIMARY KEY,
    auth_name VARCHAR(256),
    auth_srid INTEGER,
    srtext VARCHAR(2048),
    Proj4text VARCHAR(2048)
)
```

SPATIAL_REF_SYS

SRID – liczba naturalna jednoznacznie identyfikująca układ współrzędnych wewnątrz bazy danych; (SRID=4326)

AUTH_NAME – nazwa standardu danego układu odniesienia; ('EPSG:4326')

AUTH_SRID – numer ID nadany układowi w standardzie określonym w AUTH_NAME

SRTEXT – reprezentacja (WKT) przestrzennego układu odniesienia;

PROJ4TEXT – zawiera informacje umożliwiające transformację jednego układu współrzędnych w inny;

	srid integer	auth_name character varying(256)	auth_srid integer	srtext character varying(2048)	proj4text character varying(2048)
1	3819	EPSG	3819	GEOGCS["HD 1909",DATUM["Hungarian_Datum", +proj=longlat +ellps=bessel]	+proj=longlat +ellps=bessel
2	3821	EPSG	3821	GEOGCS["TWD67",DATUM["Taiwan_Datum", +proj=longlat +ellps=aust_SA"]]	+proj=longlat +ellps=aust_SA
3	3824	EPSG	3824	GEOGCS["TWD97",DATUM["Taiwan_Datum", +proj=longlat +ellps=GRS80"]]	+proj=longlat +ellps=GRS80
4	3889	EPSG	3889	GEOGCS["IGRS",DATUM["Iraqi_Geodetic", +proj=longlat +ellps=GRS80"]]	+proj=longlat +ellps=GRS80
5	3906	EPSG	3906	GEOGCS["MGI 1901",DATUM["MGI_1901", +proj=longlat +ellps=bessel"]]	+proj=longlat +ellps=bessel
6	4001	EPSG	4001	GEOGCS["Unknown datum based upon the Pseudo-Mercator projection", +proj=longlat +ellps=airy +nodelta"]]	+proj=longlat +ellps=airy +nodelta
7	4002	EPSG	4002	GEOGCS["Unknown datum based upon the Pseudo-Mercator projection", +proj=longlat +ellps=airy +nodelta"]]	+proj=longlat +ellps=airy +nodelta

GEOMETRY_COLUMNS

Jest **widokiem** czytającym z katalogów DBMS. Służy do **rejestracji kolumn przechowujących geometrię**.

Column	Type	Modifiers
f_table_catalog	character varying(256)	
f_table_schema	character varying(256)	Nazwa schematu, tabeli, kolumny
f_table_name	character varying(256)	
f_geometry_column	character varying(256)	
coord_dimension	integer	Wymiar: 2D, 3D, 4D
srid	integer	
type	character varying(30)	Typ: POINT, POLYLINE, itd.

Tworzenie bazy danych przestrzennych

Tworzymy pustą bazę danych standardowym poleceniem SQL.

```
CREATE DATABASE myspatialdb;
```

Rozszerzamy bazę o funkcjonalności PostGIS'a.

```
CREATE EXTENSION postgis;
```

Możemy pominąć krok drugi, jeżeli korzystamy z template'a.

```
CREATE DATABASE my_spatial_db TEMPLATE=template_postgis;
```

Tworzenie tabeli geo

Wykorzystujemy standardowe polecenie **CREATE TABLE**, definiując kolumnę odpowiedzialną za geometrię. Wykorzystujemy typ bazowy **geometry**.

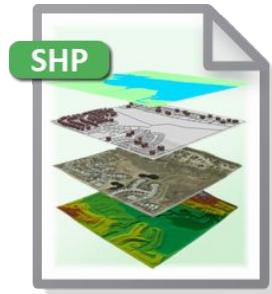
```
CREATE TABLE parks (
    park_id INTEGER,
    park_name VARCHAR,
    park_date DATE,
    the_geom GEOMETRY
);
```

Wprowadzanie danych

Wykorzystujemy polecenie **INSERT INTO**. Do standardowej składni polecenia, w miejscu wartości odpowiadającej kolumnie z geometrią wykorzystujemy funkcję **ST_GeomFromText/ST_GeomFromEWKT**.

```
INSERT INTO buildings (id, geom, name)
VALUES ( 1, ST_GeomFromText('POINT(12 34)' , -1), 'GreenH' );
```

Wprowadzanie danych



Drugim sposobem jest wykorzystanie plików **Shapefile** (.shp).

Shapefile to format przechowujący dane wektorowe (punkty, linie, poligony).

Składa się z kilku odrębnych plików o tej samej nazwie, lecz różnym rozszerzeniu:

.shp – główny plik, przechowujący geometrię obiektów, a ścisłej współrzędne ich wierzchołków,

.dbf – tabela atrybutów w formacie dBase, przechowująca w kolejnych wierszach atrybuty obiektów z pliku shp,

.shx – indeks obiektów; ten plik umożliwia odnajdywanie obiektów w pliku shp.

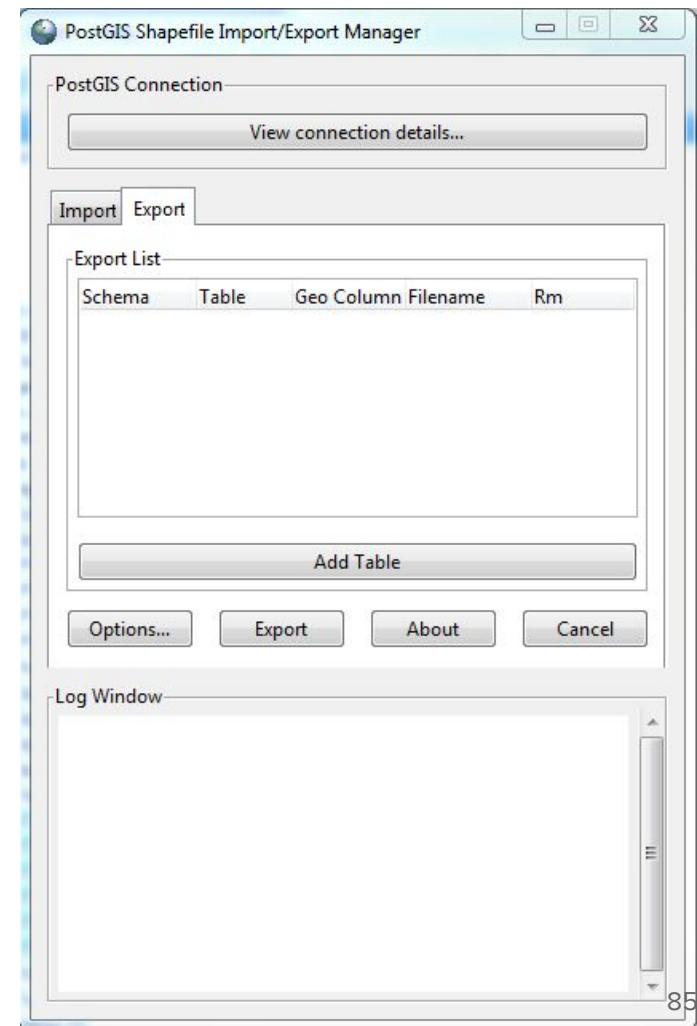
.prj lub **.qpj** – deklaracja układu współrzędnych warstwy;

Wprowadzanie danych

Pliki **Shapefile** są przekładalne w stosunku 1:1 na tabelę bazy danych.

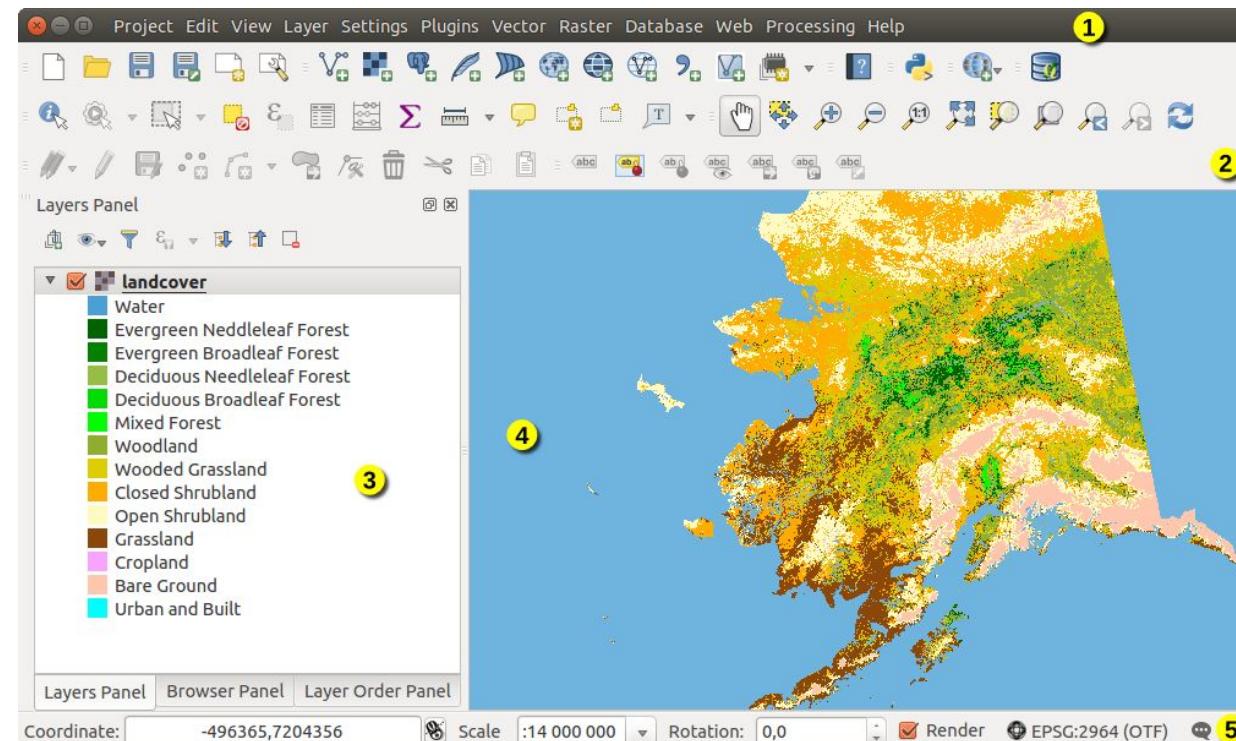
Oznacza to, iż import pliku pozwala nam zdefiniować pełną tabelę, wraz z rekordami i metadanymi.

Import realizowany jest za pomocą wtyczki **Shapefile Loader**.



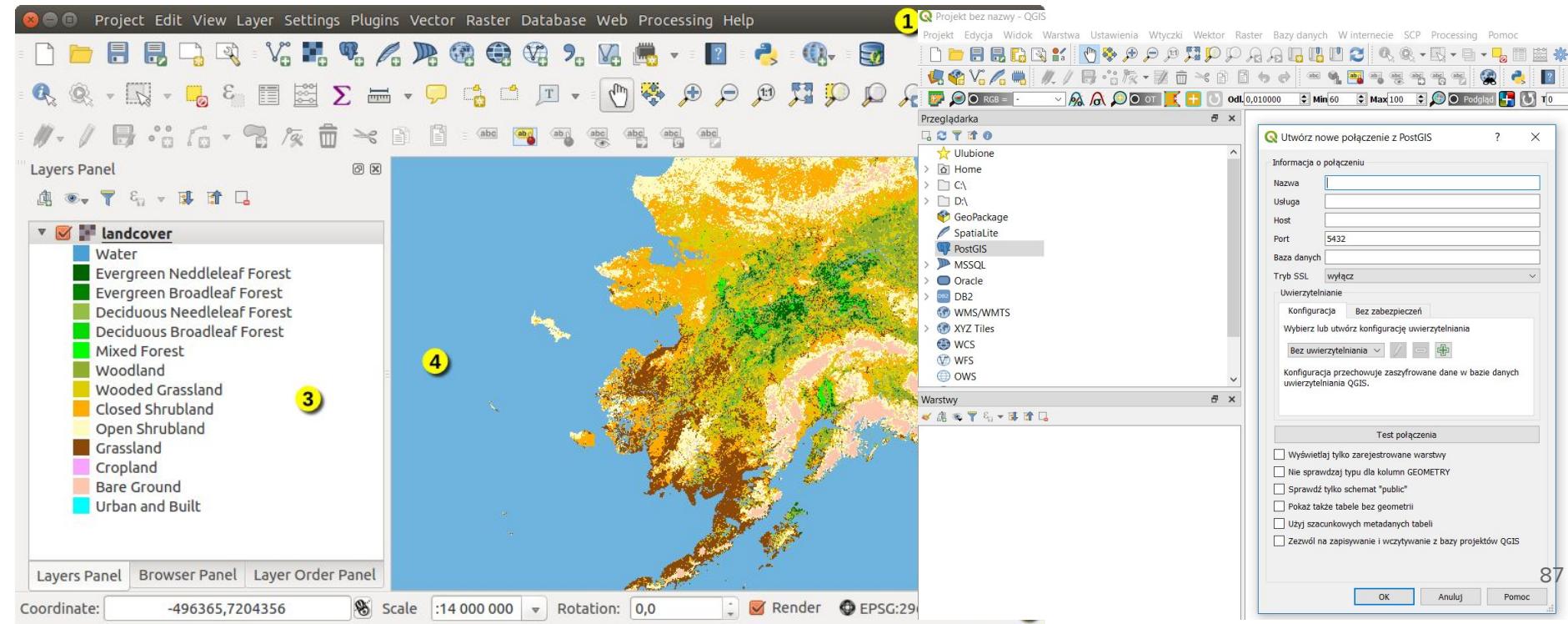
Wprowadzanie danych

Trzecim sposobem jest wykorzystanie oprogramowania GIS do połączenia z bazą danych.



Wprowadzanie danych

Trzecim sposobem jest wykorzystanie oprogramowania GIS do połączenia z bazą danych.



Wprowadzanie danych

Ostatnia możliwość to wykorzystanie oprogramowania **ETL** (ang. Extract Transform Load).

ETL to proces, w którym dane przepływają ze źródeł do systemów docelowych.

- odczyt danych następuje z dowolnego źródła danych przestrzennych (i nie tylko),
- dane następnie przechodzą proces transformacji - na przykład zmianę schematu tabeli, transformację układu współrzędnych, czy zmianę wymiaru 2D na 3D,
- dane końcowe ładowane są do dowolnego DBMS, łącząc informacje z wielu różnych źródeł w jeden spójny model.

Wprowadzanie danych



Zapytania SQL



Zapytania SQL

Wyświetlanie zawartości tabeli:

SELECT * FROM nazwaTabeli; (“*” wskazuje na wszystkie kolumny)

SELECT imie, nazwisko FROM pracownicy; (wyświetl tylko dwie kolumny)

Tworzenie aliasów kolumn:

SELECT imie AS ImiePracownika, pensja+100 AS NowaPensja FROM firma.pracownicy;

“AS” nadaje aliasy nazwom kolumn lub wynikom danego wyrażenia.

Zapytania z warunkiem wybierania

Warunek wybierania (opcjonalny) określa się przy pomocy słowa kluczowego **WHERE**. W warunku tym można stosować **operatorы арифметичные, логические, текстовые** oraz **специальные конструкции, упрощающие поиск**.

```
SELECT * FROM tabela WHERE kolumna1 > 100;  
SELECT * FROM tabela WHERE kolumna1 > 100 OR kolumna2 < 300;  
SELECT * FROM tabela WHERE (kolumna1 > 100 OR kolumna2 < 300) AND kolumna 3 <> 0;
```

Sprawdzanie obecności wartości atrybutu: IS NULL lub IS NOT NULL:

```
SELECT * FROM tabela WHERE data_obrony IS NULL AND data_odejscia IS NOT NULL;
```

Funkcje geoprzestrzenne

Funkcje geoprzestrzenne, które definiuje standard OGC posiadają następujące cechy:

- jako argument przyjmują jeden lub więcej obiektów geometrycznych,
- zwracają wartości liczbowe, logiczne lub geometryczne,
- w zależności od typu, występują jako element wyrażenia lub jako predykat w klauzuli WHERE/HAVING,
- są wykorzystywane do budowy złączeń przestrzennych (ang. spatial joins).

- ❑ **ST_Length();**
- ❑ **ST_Area();**
- ❑ **ST_Perimeter();**
- ❑ **ST_Distance();**
- ❑ **ST_Buffer();**
- ❑ **ST_Contains();**
- ❑ **ST_Difference();**
- ❑ **ST_SymDifference();**
- ❑ **ST_Intersection();**
- ❑ **ST_Disjoint();**
- ❑ **ST_Intersects();**
- ❑ **ST_Overlaps();**
- ❑ **ST_Touches();**
- ❑ **ST_Within();**
- ❑ **ST_NRings();**
- ❑ **ST_NumInteriorRings();**
- ❑ **ST_AsText();**
- ❑ **ST_GeomFromText();**
- ❑ **ST_DWithin**
- ❑ **ST_Equals**
- ❑ **ST_RemoveRepeatedPoints**
- ❑ **ST_Union**

Length, Area, Perimeter

Funkcje **ST_Length**, **ST_Area**, **ST_Perimeter** przyjmują jako argument obiekt geometryczny, a zwracają liczbę zmiennoprzecinkową.

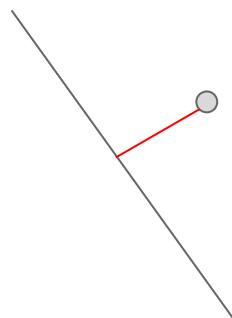
```
float ST_Length(geometry);  
float ST_Area(geometry);  
float ST_Perimeter(geometry);
```

Argumentem funkcji jest nazwa kolumny przechowującej geometrię

```
SELECT riverName, ST_Length(the_geom) AS riverLength  
FROM rivers  
WHERE riverName = 'Wisła';
```

Distance

Funkcja **ST_Distance** zwraca **odległość euklidesową** (w [m]) pomiędzy dwoma obiektami geometrycznymi.

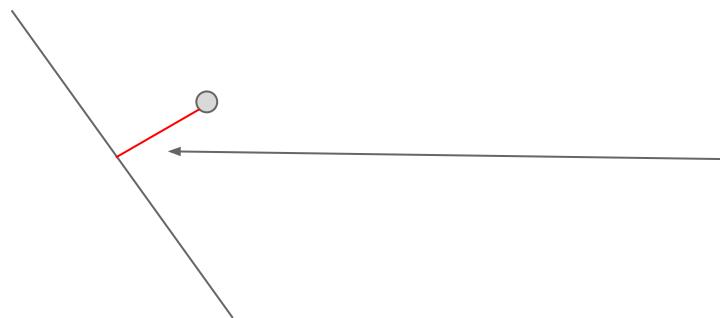


$$\text{distance}(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
SELECT ST_Distance(schools.the_geom, parks.the_geom) AS DistanceToParks  
WHERE schools.name = 'L.O. nr 5' AND parks.name = 'Krakowski';
```

Shortest line

Funkcja **ST_ShorestLine** zwraca **obiekt geometryczny**, będący najkrótszą linią pomiędzy dwoma obiektami geometrycznymi.



Zwracana jest geometria tego odcinka w WKB

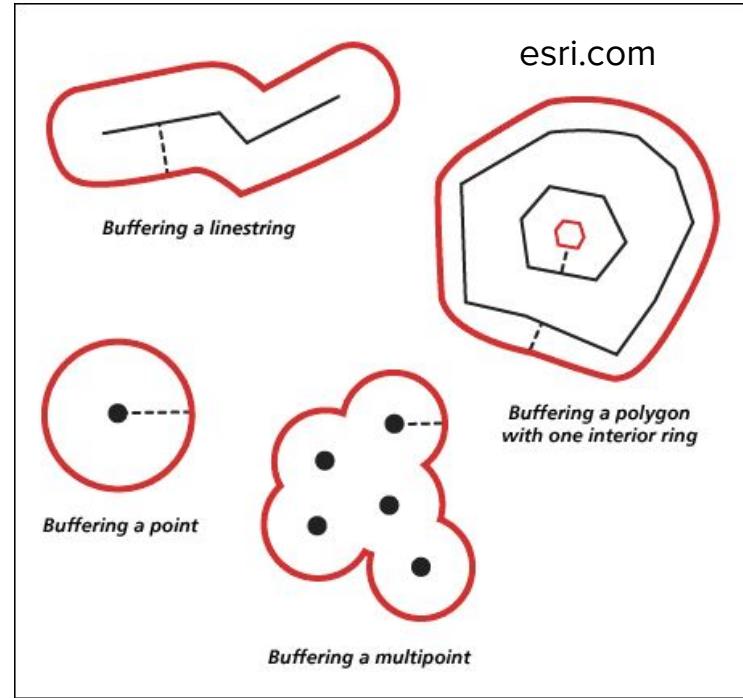
```
SELECT ST_ShorestLine(schools.the_geom, parks.the_geom) AS ShortestLine  
WHERE schools.name = 'L.O. nr 5' AND parks.name = 'Krakowski';
```

Buffer

Buforowanie jest jedną z najważniejszych operacji w GIS. Bufor to strefa, której granice są odległe od obiektu o zadaną wielkość.

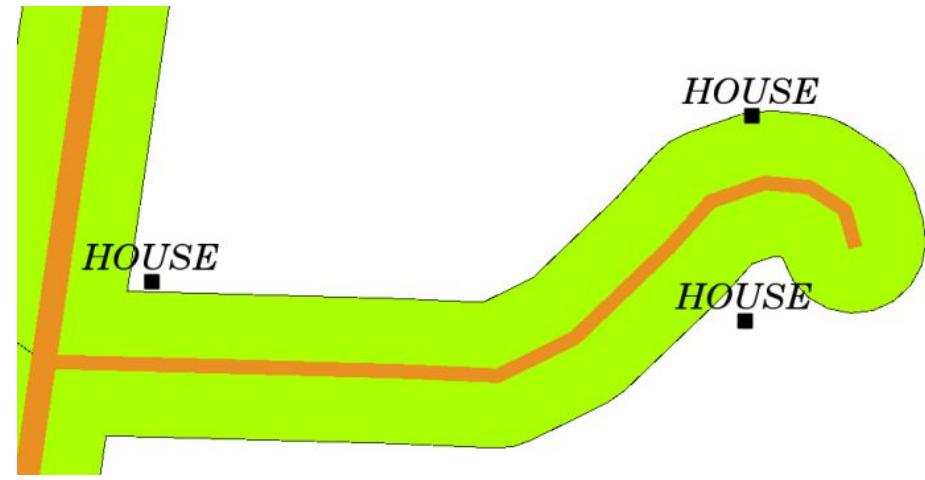
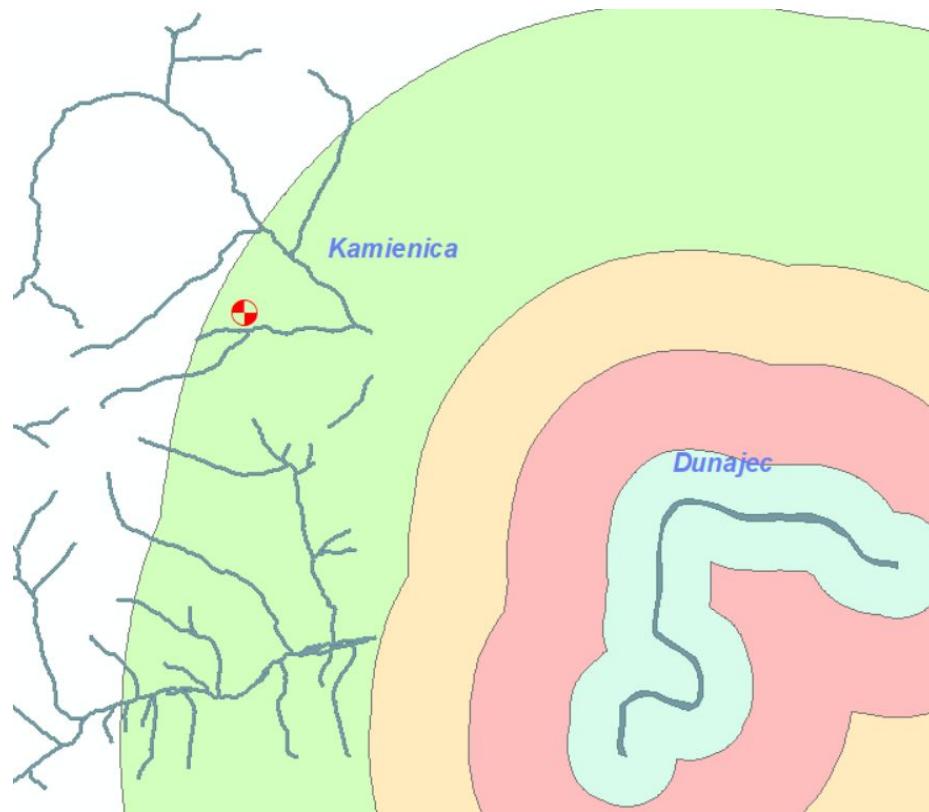
Wejście: wszystkie rodzaje geometrii.

Wynik: poligon reprezentujący strefę.



```
SELECT ST_Buffer(the_geom, 100.0) AS BufferedArea  
FROM schools;
```

Buffer

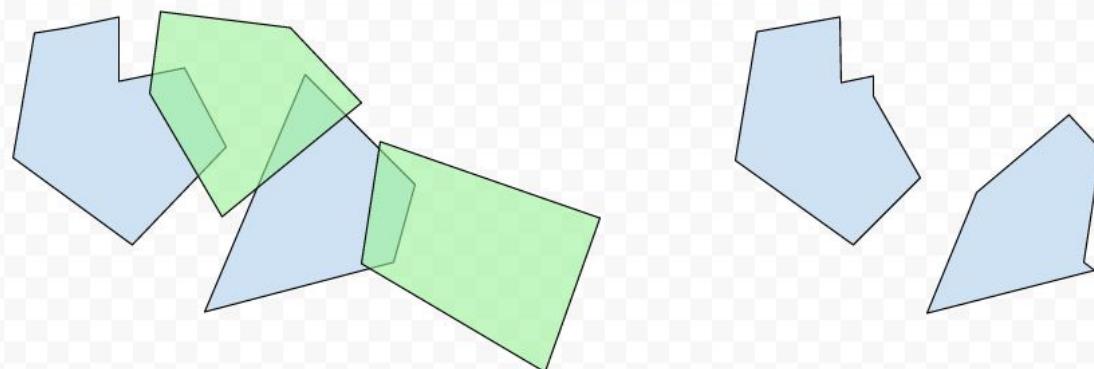


Difference, SymDifference

Różnica i różnica symetryczna to dwie operacje teoriomnogościowe.

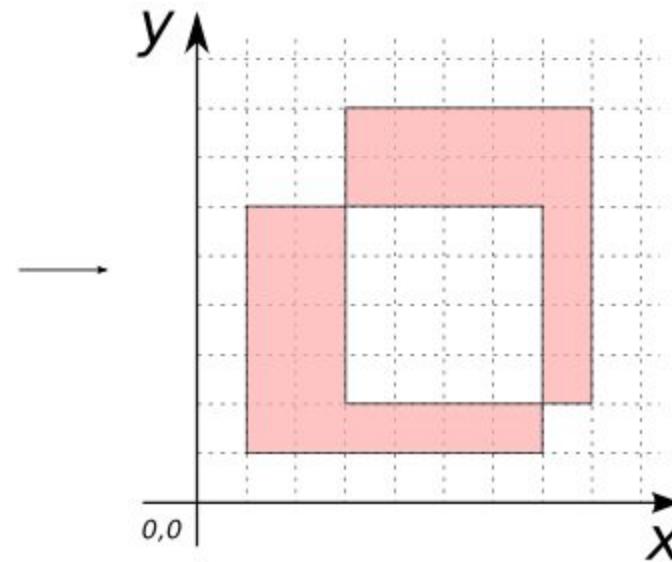
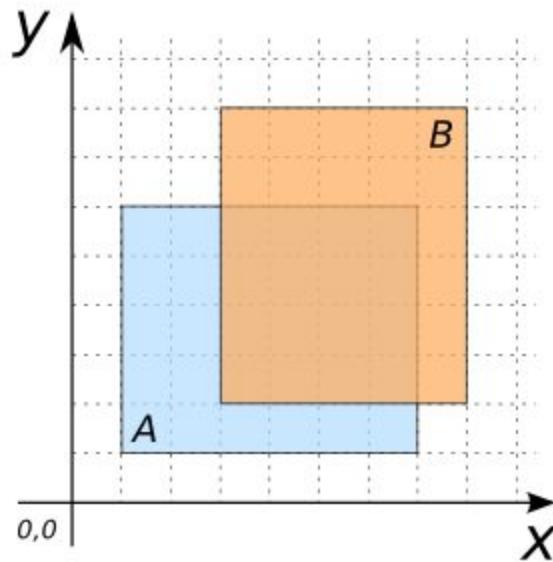
Wejście: wszystkie rodzaje geometrii oprócz **GEOMETRYCOLLECTION**.

Wynik: poligon reprezentujący
różnicę lub różnicę symetryczną.



```
SELECT ST_Difference(t.the_geom, s.the_geom) FROM trees t, swamps s
WHERE t.name = 'Puszcza Niepołomicka';
```

Difference, SymDifference



h2gis.com

```
SELECT ST_SymDifference(t.the_geom, s.the_geom) FROM trees t, swamps s  
WHERE t.name = 'Puszcza Niepołomicka';
```

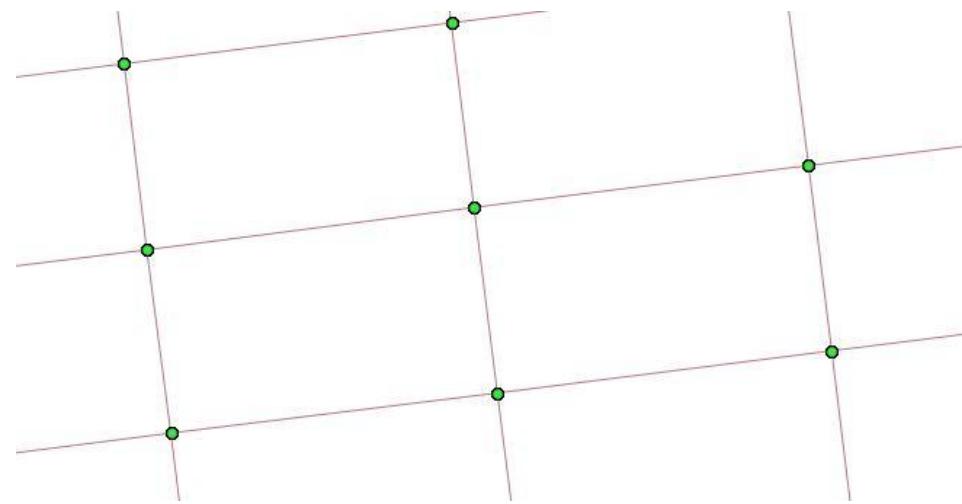
Intersection

Funkcja ***geometry ST_Intersection(geometry, geometry)*** odpowiada za realizację operacji wyznaczania części wspólnej obiektów geometrycznych.

Wejście: dowolne obiekty geometry

Wyjście: obiekt(y) geometry

o typie zależnym od wejścia.

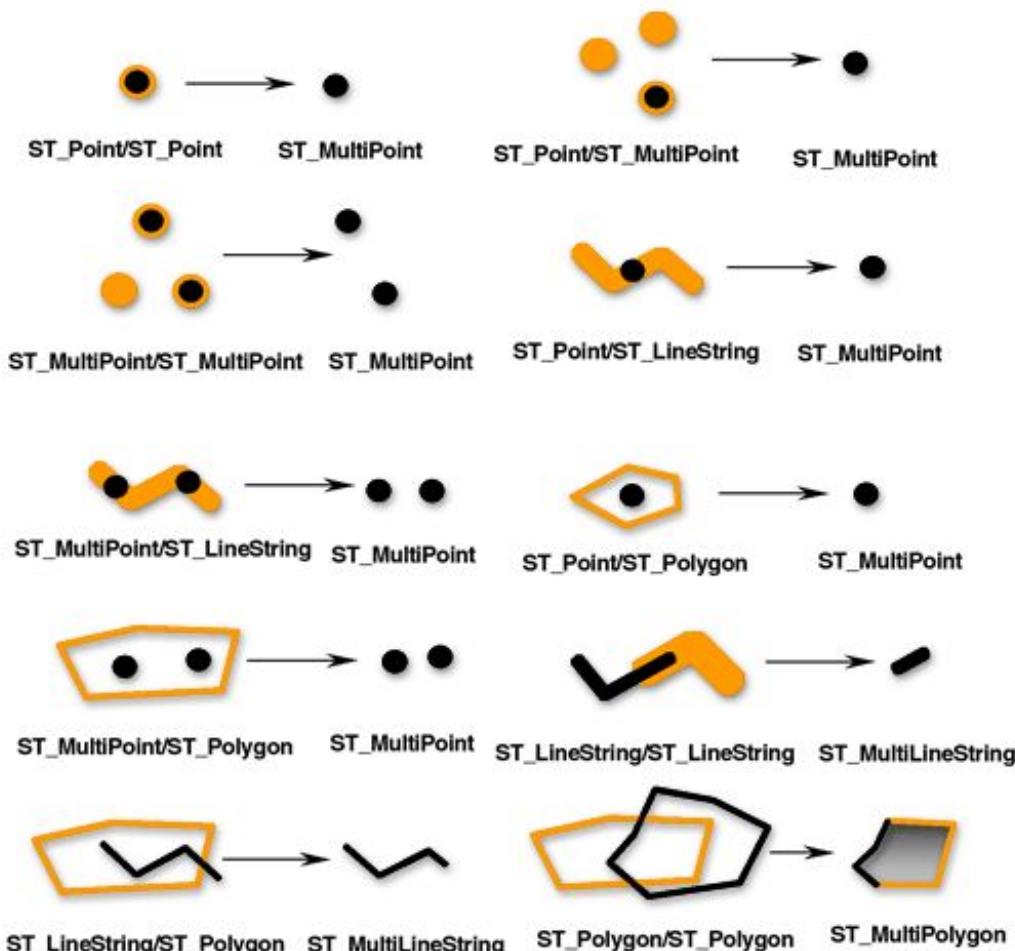


```
SELECT ST_Intersection(ro.the_geom, ra.the_geom) FROM roads ro, railroads ra;
```

Intersection

Wejście: dowolne obiekty geometry

Wyjście: obiekt(y) geometry o typie zależnym od wejścia.



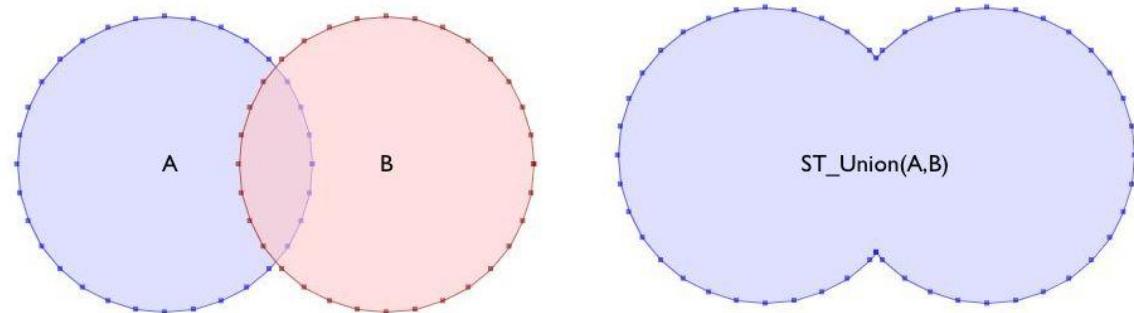
Union

Funkcja ***geometry ST_Union(geometry, geometry)*** odpowiada za realizację operacji wyznaczania sumy teoriomnogościowej obiektów geometrycznych.

Wejście: dowolne obiekty geometry

Wyjście: obiekt(y) geometry

o typie zależnym od wejścia.

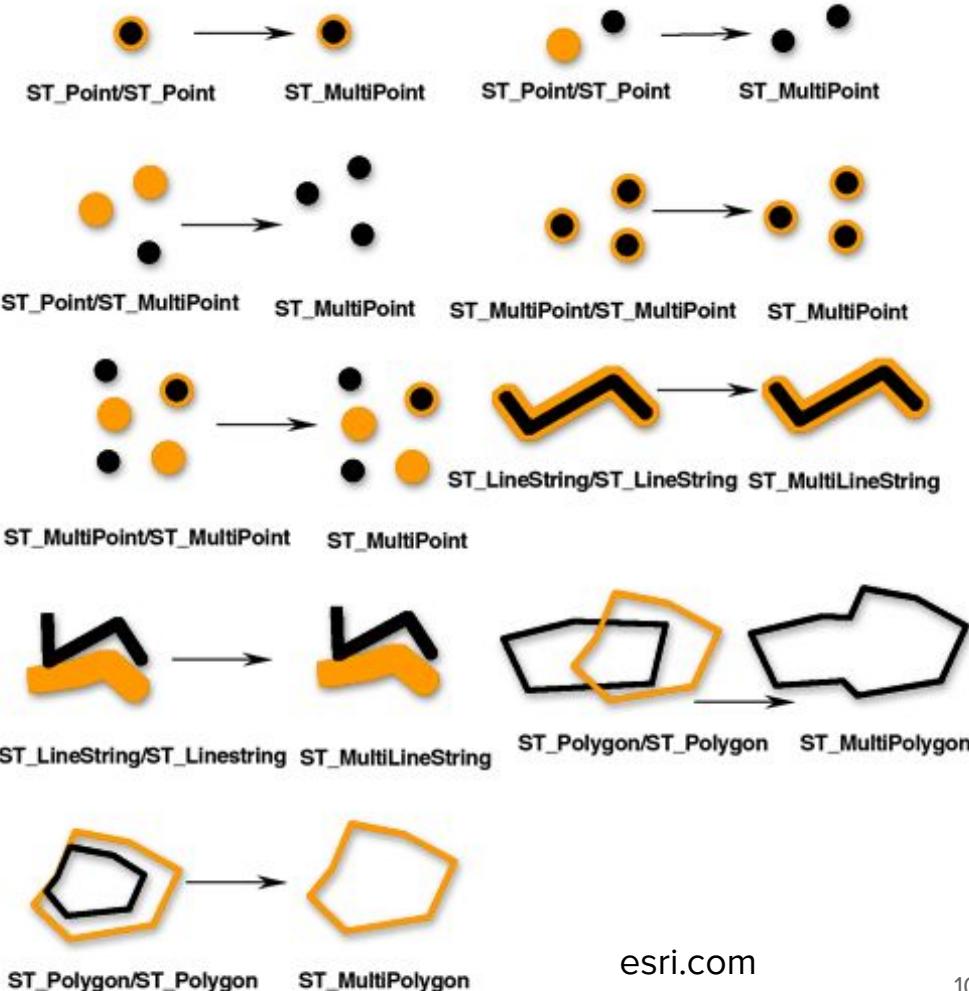


```
SELECT ST_Union(t.the_geom, g.the_geom) FROM trees t, grasslands g;
```

Union

Wejście: dowolne obiekty geometry

Wyjście: obiekt(y) geometry o typie zależnym od wejścia.



Intersects

Funkcja ***boolean ST_Intersects(geometry, geometry)*** należy do grupy funkcji testujących. Zwraca wartość true, jeżeli obiekty się przecinają.

Wejście: dowolne obiekty geometry

Wyjście: TRUE/FALSE

Analogiczne funkcje: **ST_Crosses, ST_Overlaps**

```
SELECT ro.name, ri.name FROM roads ro, rivers ri  
WHERE ST_Intersects(ro.the_geom, ri.the_geom);
```

Intersects



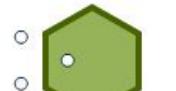
Point & Multipoint



Point & Linestring



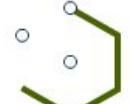
Linestring & Linestring



Multipoint & Polygon



Multipoint & Multipoint



Multipoint & Linestring



Linestring & Polygon



Linestring & Multipolygon

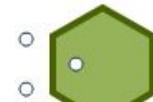
Cross



Multipoint & Linestring



Linestring & Linestring



Multipoint & Polygon



Linestring & Multipolygon

Overlap



Multipoint & Multipoint



Linestring & Linestring



Polygon & Polygon

Dowolny typ geometrii

TRUE, jeżeli przecięcie jest typem geometrii o 1 wymiar niższy niż obiekty wejściowe (linia, linia) -> punkt oraz część wspólna leży we wnętrzu porównywanych obiektów

Tylko ten sam typ geometrii

Contains

Funkcja ***boolean ST_Contains(geometry A, geometry B)*** należy do grupy funkcji testujących. Zwraca wartość **true**, jeżeli **obiekt A zawiera obiekt B** oraz **żadne punkty B nie leżą na zewnątrz A**.

Wejście: dowolne obiekty typu geometry

Wyjście: TRUE/FALSE

Analogiczne funkcje: **ST_Within, ST_ContainsProperly**

```
SELECT l.name, p.name FROM lakes s, parks p  
WHERE ST_Contains(p.the_geom, l.the_geom);
```

Within/Contains

Contains

Funkcja ***boolean ST_Contains(A, B)***

Zwraca wartość **true**, jeżeli **obiekt A** za
obiekt **B** oraz żadne punkty **B** nie leżą
zewnętrz A.



Point & Multipoint



Multipoint & Multipoint



Point & Linestring



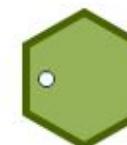
Multipoint & Linestring



Linestring & Linestring



Linestring & Polygon



Point & Polygon



Multipoint & Polygon

DWithin

Funkcja ***boolean ST_DWithin(geometry A, geometry B, distance)*** należy do grupy funkcji testujących. Zwraca wartość **true**, jeżeli **obiekt B znajduje się w odległości distance od obiektu A**.

Wejście: dowolne obiekty typu geometry

Wyjście: TRUE/FALSE

Znajdź punkty znajdujące się w promieniu 1609 m od drogi.

```
SELECT roads.road_name, pois.poi_name  
FROM roads INNER JOIN pois  
ON ST_DWithin(roads.geom, pois.geom, 1609);
```

Equals

Funkcja ***boolean ST_Equals(geometry A, geometry B)*** należy do grupy funkcji testujących. Zwraca wartość **true**, jeżeli **obiekty A i B są identyczne pod względem geometrycznym**.

Wejście: dowolne obiekty typu geometry

Wyjście: TRUE/FALSE

Uwaga: funkcja jest zależna od wartości SRID!

```
SELECT name FROM nyc_subway_stations  
WHERE ST_Equals(geom, '0101000020266900000EEBD4CF27CF2141BC17D69516315141');
```

Equals

Equals

Funkcja ***boolean ST_Equals(A, B)***
zwraca wartość **true**, jeżeli **obiekty A i B są identyczne pod względem geometrycznym.**

Wejście: dowolne obiekty typu geometry

Wyjście: TRUE/FALSE

Uwaga: funkcja jest zależna od wartości SRID!



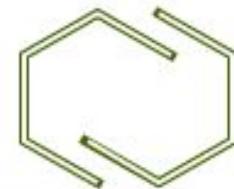
Point & Multipoint



Multipoint & Multipoint



Linestring & Linestring



Multilinestring & Multilinestring



Polygon & Polygon



Multipolygon & Multipolygon

Touches

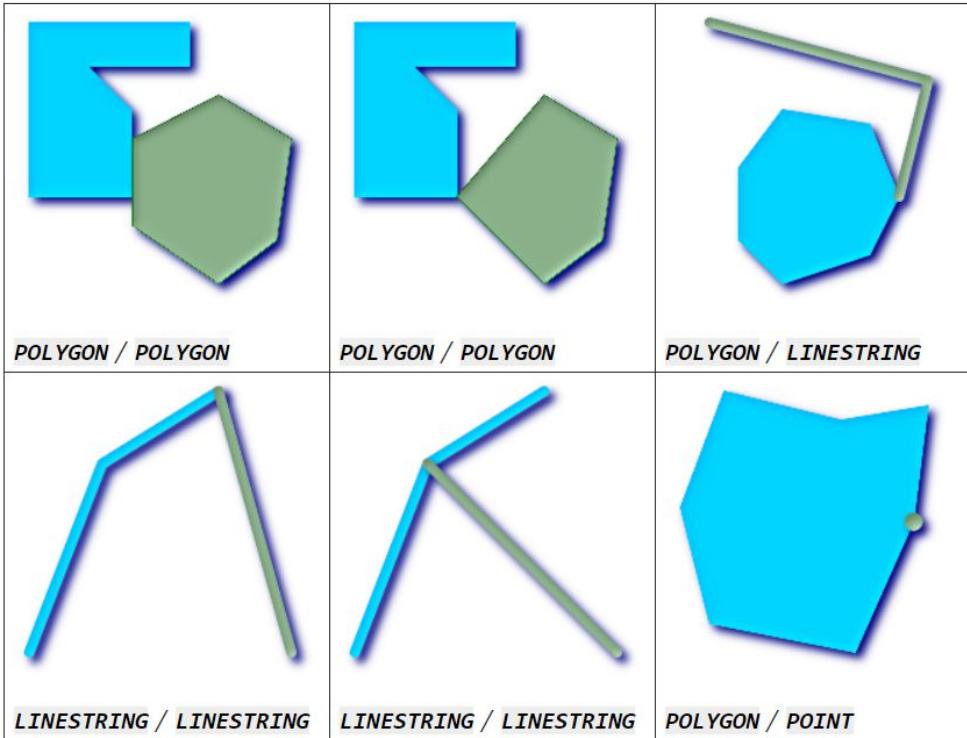
Funkcja ***boolean ST_Touches(geometry A, geometry B)*** należy do grupy funkcji testujących. Zwraca wartość **true**, jeżeli **obiekty A i B są styczne**.

Wejście: dowolne obiekty typu geometry

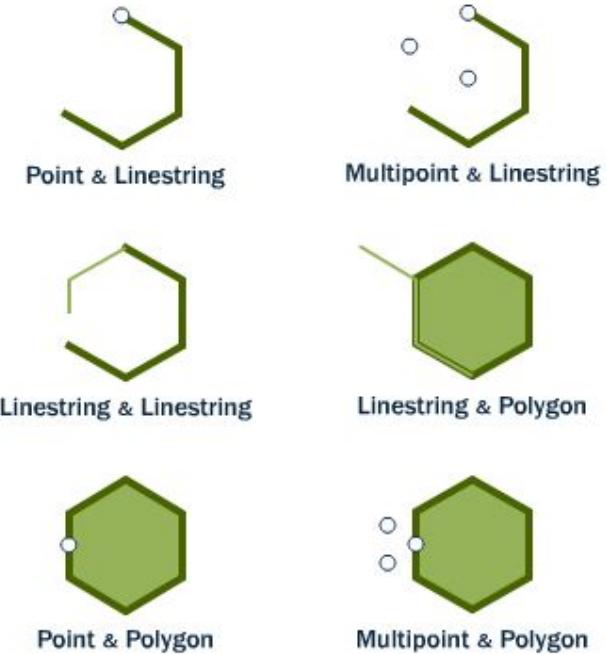
Wyjście: TRUE/FALSE

```
SELECT h.name, r.name FROM highways h, roads r  
WHERE ST_Touches(h.geom, r.geom);
```

Touches



Touch



Disjoint

Funkcja ***boolean ST_Disjoint(geometry A, geometry B)*** należy do grupy funkcji testujących. Zwraca wartość **true**, jeżeli **obiekty A i B są rozłączne**.

Wejście: dowolne obiekty typu geometry

Wyjście: TRUE/FALSE

```
SELECT b.name FROM buildings b, districts d  
WHERE ST_Disjoint(b.geom, d.geom);
```

Disjoint

Disjoint

Funkcja ***boolean ST_Disjoint(A, B)***
należy do grupy funkcji testujących.
Zwraca wartość **true**, jeżeli **obiekty A i B są rozłączne**.

Wejście: dowolne obiekty typu geometry

Wyjście: TRUE/FALSE



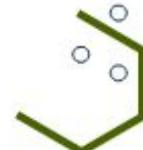
Point & Multipoint



Multipoint & Multipoint



Point & Linestring



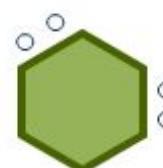
Multipoint & Linestring



Linestring & Linestring



Linestring & Polygon



Multipoint & Polygon



Polygon & Polygon

Zapytania zagnieżdżone

- Język SQL umożliwia tworzenie zapytań **zagnieżdżonych - podzapytań**.
- Podzapytania to zapytania SQL, które znajdują się wewnętrz innego zapytania. Są zawsze otoczone nawiasami ().

Cechy podzapytania:

- podzapytanie musi być ujęte w nawiasy i nie może być zakończone średnikiem,
- podzapytanie może być umieszczone we frazach: SELECT, FROM, HAVING, WHERE,
- podzapytanie może zawierać odwołania do kolumn wymienionych w zapytaniu zewnętrznym.

Zapytania zagnieżdżone

Zapytanie nadzędne

```
SELECT ProductName, Price FROM [Products]
    WHERE ProductID IN (SELECT ProductID FROM [OrderDetails] GROUP BY ProductID HAVING
COUNT(*) > 5);
```

Podzapytanie

Zapytania zagnieżdżone

Zapytania skorelowane:

- podzapytania są powiązane z zapytaniem nadzędnym
- wykorzystujemy **aliasy tabel**

```
SELECT last_name, salary, department_id  
FROM employees outer  
WHERE salary > (SELECT AVG(salary)  
                  FROM employees  
                  WHERE department_id =  
                        outer.department_id);
```

Alias tabeli nadzędnej

Powiązanie zapytania
nadzędnego i podzapytania
warunkiem

Zapytania zagnieżdżone

Podzapytanie jako zwracana kolumna:

```
SELECT ProductName, Price, (SELECT AVG(Price) FROM [Products]) AS AveragePrice  
FROM [Products];
```

ProductName	Price	AveragePrice
Chais	18	28.866363636363637
Chang	19	28.866363636363637
Aniseed Syrup	10	28.866363636363637
Chef Anton's Cajun Seasoning	22	28.866363636363637
Chef Anton's Gumbo Mix	21.35	28.866363636363637
Grandma's Boysenberry Spread	25	28.866363636363637
Uncle Bob's Organic Dried Pears	30	28.866363636363637

Zapytania zagnieżdżone

Podzapytanie jako element klauzuli FROM

- zwraca wiele wartości
- wyniki traktowane są jako tabela

```
SELECT AVG(TotalQuantity) AS AVGQuantity, ProductID  
    FROM (SELECT ProductID, SUM(Quantity) AS TotalQuantity  
          FROM [OrderDetails] GROUP BY ProductID)  
        GROUP BY ProductID;
```

AVGQuantity	ProductID
159	1
341	2
80	3

```
pracownicy (pesel INT, dochody INT)
```

```
=> SELECT * FROM pracownicy;
```

```
pesel | dochody
```

pesel	dochody
1	2000
2	3000
3	4000
4	5000

```
zatrudnienie (regonfirmy INT,  
peselpracownika INT, pensja INT)
```

```
=> SELECT * FROM zatrudnienie;
```

```
regonfirmy | peselpracownika | pensja
```

regonfirmy	peselpracownika	pensja
111	4	5000
112	2	3000
113	3	3000
113	1	2000

Podzapytanie proste - wykonywane tylko raz.

```
=> SELECT regonfirmy FROM zatrudnienie  
      WHERE peselpracownika  
IN (SELECT pesel FROM pracownicy WHERE  
      dochody > 3000);
```

```
regonfirmy
```

111
113

Zagnieżdżone: zapytanie nadzędne przekazuje dane do podzapytanego. Podzapytanie wykonywane dla każdej krotki.

```
=> SELECT peselpracownika FROM  
      zatrudnienie WHERE pensja <  
      (SELECT dochody FROM pracownicy  
      WHERE pesel = peselpracownika);
```

```
peselpracownika
```

3

Zapytania zagnieżdżone

Podzapytania a związki

- Dla części zapytań zagnieżdżonych można skonstruować równoważne związki tabel:

```
SELECT DISTINCT * FROM TabA WHERE kolA1 IN (SELECT kolB1 FROM TabB);  
SELECT DISTINCT TabA.* FROM TabA INNER JOIN TabB ON TabA.kolA1 = TabB.kolB1;
```

```
SELECT * FROM TabA WHERE kolA1 NOT IN (SELECT kolB1 FROM TabB);  
SELECT kolA1, kolA2 FROM TabA LEFT OUTER JOIN TabB ON TabA.kolA1 = TabB.kolB1  
                           WHERE TabB.kolB1 IS NULL;
```

Indeksy

Indeks:

- Jest strukturą fizyczną w bazie danych.
- Jego zadaniem jest optymalizacja czasu potrzebnego na dostęp (przeszukanie) tabel.
- Indeks zakładamy na pojedynczych atrybutach lub ich zbiorach.
- Model fizyczny to uporządkowany plik **rekordów indeksu**.
- Rekord indeksu, składający się z:
 - klucza reprezentującego jedną z wartości występujących w atrybutach indeksowych relacji
 - wskaźnika do bloku danych zawierający krotkę, której atrybut indeksowy równy jest kluczowi

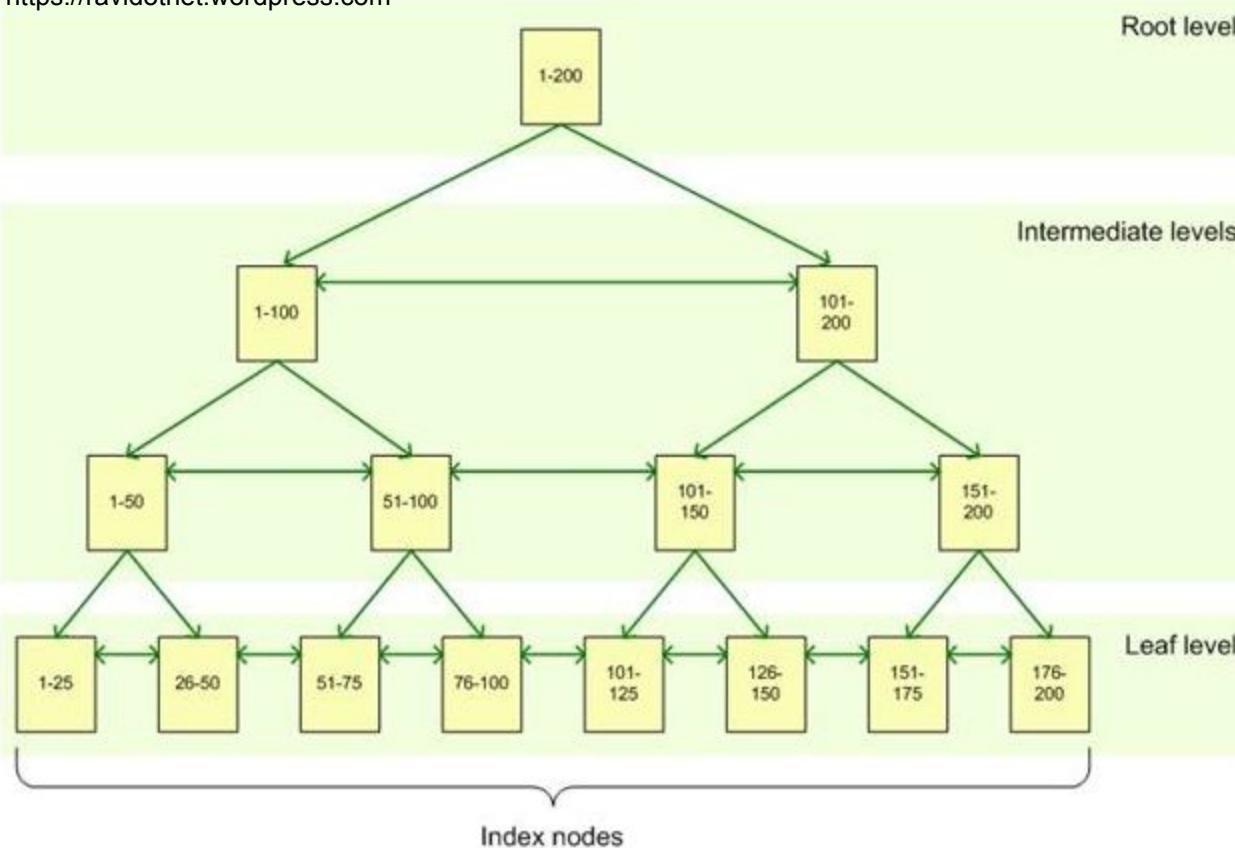
Indeks podstawowy (primary index)

Składnia:

```
CREATE INDEX nazwaIndeksu ON tabela (atrybut 1, atrybut 2, ...);
```

przykład:

```
CREATE INDEX city_idx ON customers (customers.city);
```



Indeks przestrzenny

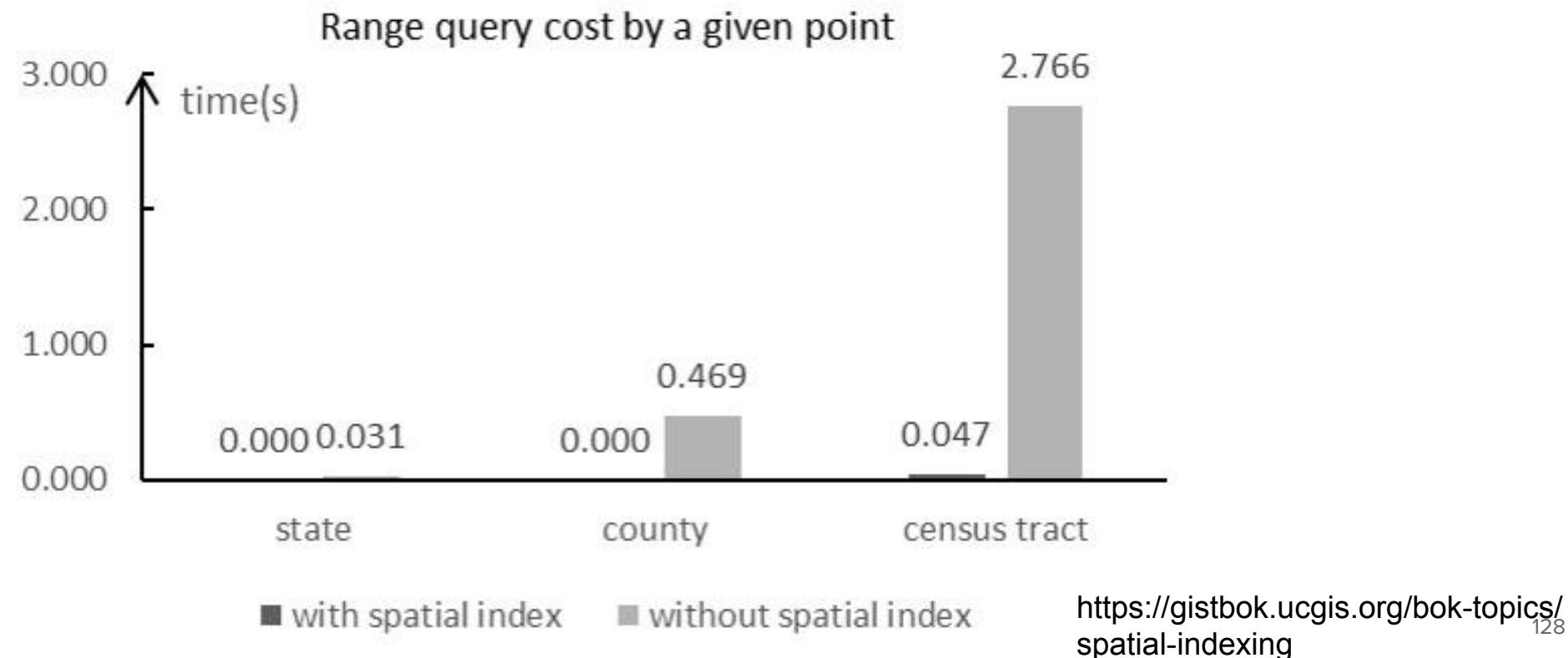
Indeks przestrzenny to **struktura danych**, która umożliwia efektywny dostęp do obiektów przestrzennych.

Jest to powszechna technika stosowana w przestrzennych bazach danych.

Różnorodne operacje przestrzenne wymagają wsparcia z indeksu przestrzennego w celu wydajnego przetwarzania:

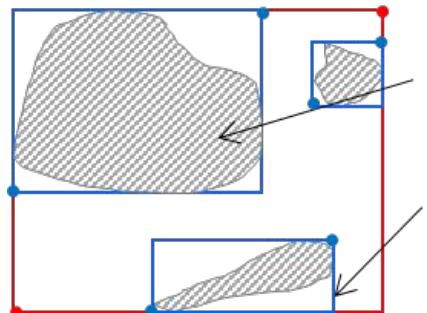
- Zapytanie o zakres: znajdowanie obiektów zawierających dany punkt lub nakładających się na obszar zainteresowania.
- Spatial joins.
- KNN - znajdowanie K-najbliższych sąsiadów w przestrzeni.

Indeks przestrzenny



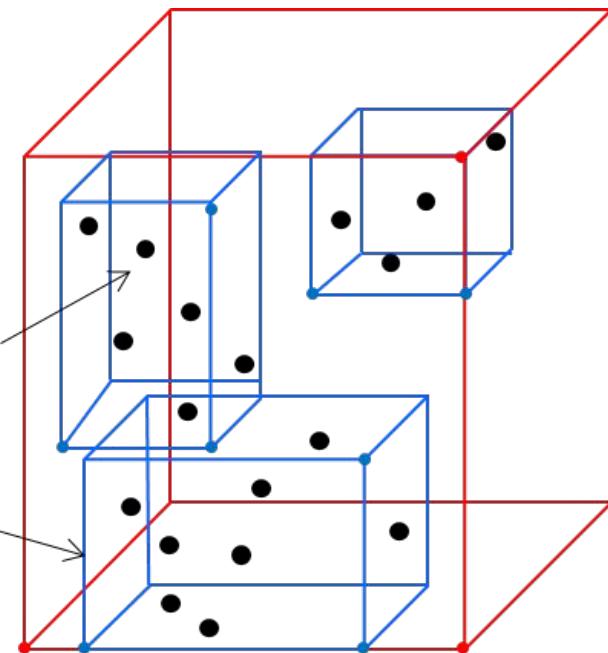
Indeks przestrzenny

Najpierw przeszukujemy MBR-y



(a)

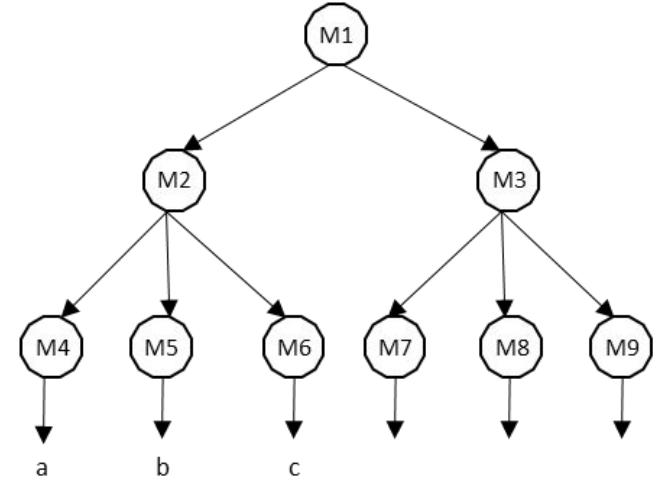
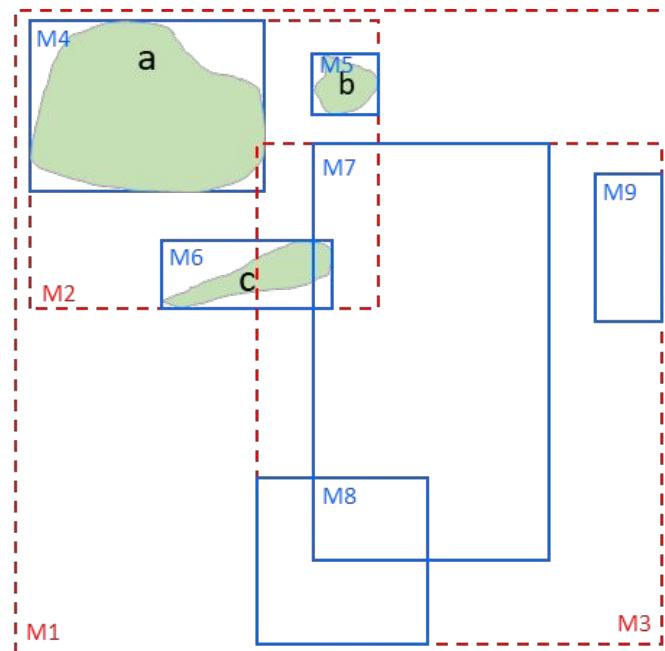
Spatial objects
MBR



(b)

Indeks przestrzenny

R-drzewa



R-drzewa bazują na hierarchicznej strukturze MBR

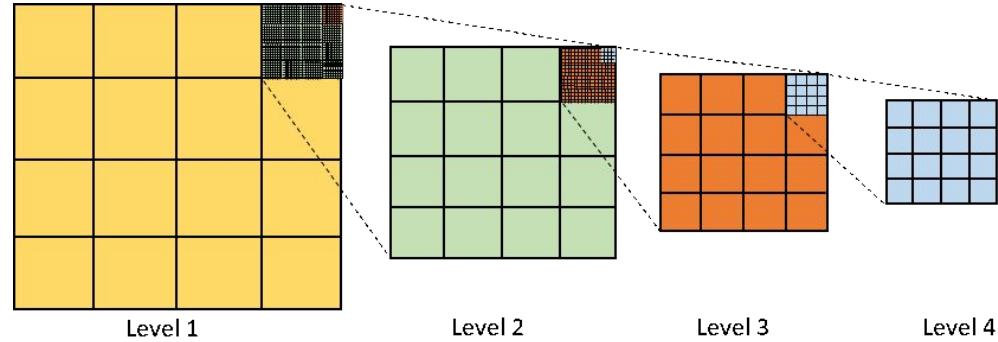
Indeks przestrzenny

Fixed grid index

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	16

Diagram illustrating a fixed grid index. A 4x4 grid of cells is shown, each containing a unique identifier from 01 to 16. Overlaid on the grid are three green-shaded regions labeled 'a', 'b', and 'c'. Region 'a' covers cells 02, 06, and 10. Region 'b' covers cells 04 and 08. Region 'c' covers cells 11 and 12.

01	a
02	a
03	a
04	b
...	
11	c
12	c



Indeks ten to tablica $n \times n$ komórek o równej wielkości. Każdy z nich jest powiązany z listą obiektów przestrzennnych, które przecinają się lub pokrywają z komórką.

Rastry

Celem projektu **PostGIS Raster** było wprowadzenie typu **RASTER** na podobnej zasadzie jak istniejący już typ **GEOMETRY** dla danych wektorowych.

PostGIS Raster dostarcza szereg różnych funkcji SQL, które funkcjonują zarówno dla danych wektorowych, jak i rastrowych.

PostGIS RASTER

- typ RASTER - uzupełnia istniejący typ GEOMETRY;
- funkcje i operatory przygotowane dla typu RASTER są bardzo podobne jak te, które pracują z danymi wektorowymi (np. **ST_Resample**);
- część funkcji i operatorów pracuje zarówno dla danych typu RASTER jak również dla danych typu GEOMETRY (np. **ST_Intersects**);
- umożliwia prosty sposób wczytywania danych rastrowych różnego typu do bazy (**gdal2wktraster.py**);

GDAL

GDAL to biblioteka służąca do odczytu i zapisu rastrowych danych geoprzestrzennych.

Lista formatów rastrowych obsługiwanych przez GDAL :

http://www.gdal.org/formats_list.html

Przykładowe formaty obsługiwane przez GDAL:

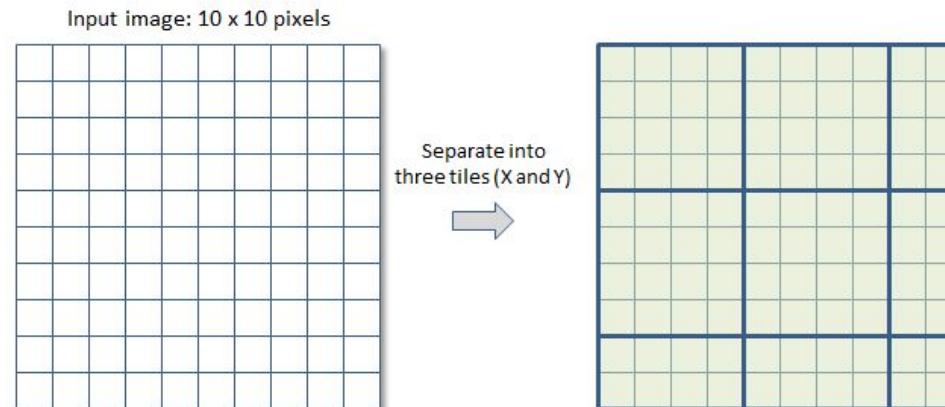
- .tiff
- .jpg
- .bmp
- .ers (ErMapper)
- .n1 (ENVISAT Image Product)

Rastry

Kafel rastra (ang. raster tile)

W wielu systemach GIS rastry dzielone są na tzw. tafle (ang. tile). Korzyścią wynikającą z takiego podziału jest przede wszystkim optymalizacja składowania i przetwarzania.

Optymalny rozmiar tafli zależy od wielu czynników takich jak: typ danych, rozdzielcość, skala, czy przeznaczenie.



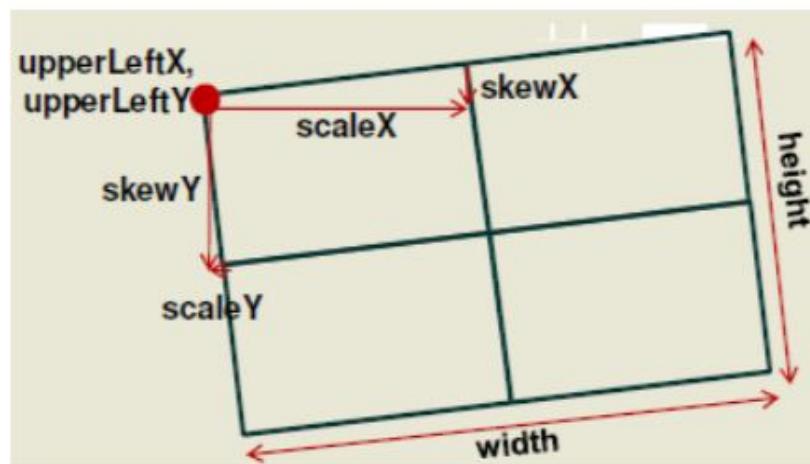
Rastry

1 tabela = 1 powierzchnia

1 wiersz tabeli = 1 raster/ jedna „tafla” rastra/obiekt rastrowy

Każda tafla rastra posiada zdefiniowaną:

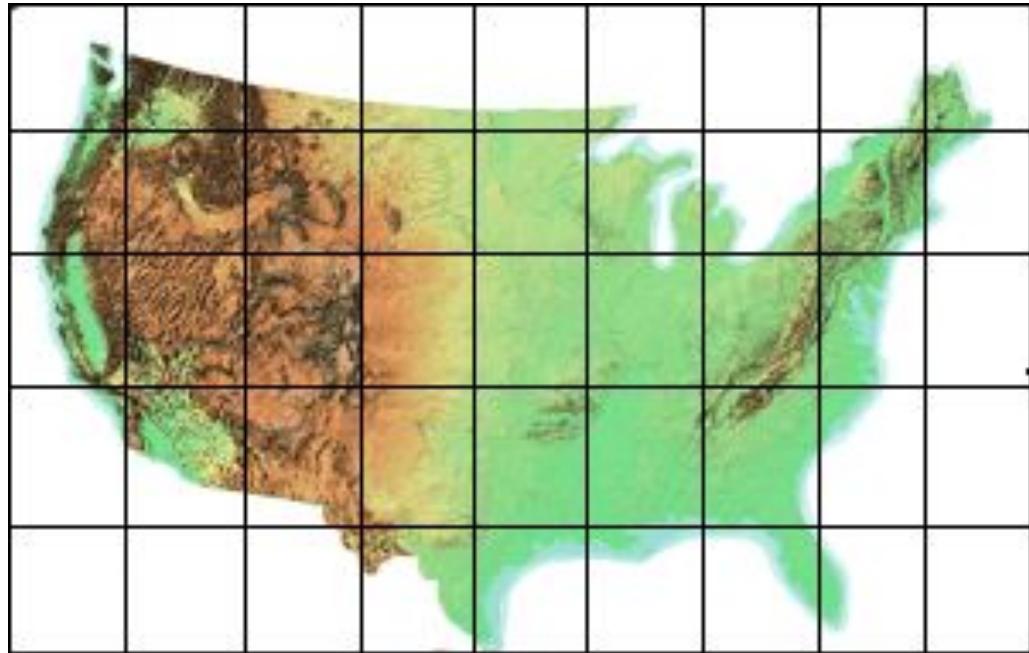
- wielkość piksela;
 - wysokość i szerokość;
 - liczbę kanałów;
- a także..
- wartości w pikselach dla poszczególnych kanałów;
 - oznaczenie braku danych dla każdego kanału.



Rastry

Maksymalna wielkość danych:

- 1GB dla rastra („tafli”);
- 32 TB dla powierzchni;



Możliwość przechowywania obrazów wielospektralnych.

Brak realnego ograniczenia co do liczby możliwych do przechowywania kanałów.

Rastry

Struktura przechowywania danych rasterowych w PostGIS jest bardzo podobna do struktury danych wektorowych w PostGIS.

Atrybuty rastrów (tafli) przechowywane w jednej tabeli nie muszą być ze sobą powiązane i nie muszą tworzyć jednej znaczącej powierzchni.

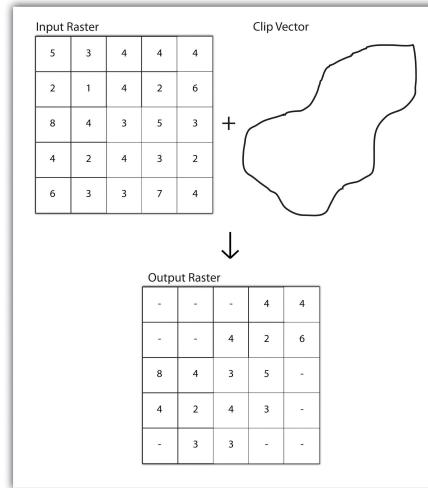
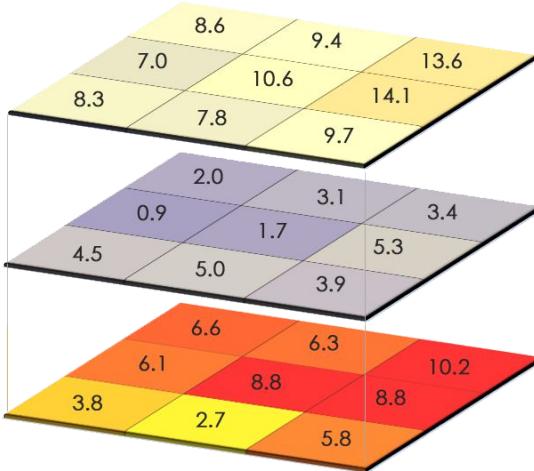
W PostGIS Raster rastry (tafle) w jednej tabeli:

- mogą **mieć różny** rozmiar (np. jeden raster/tafla ma wymiary 256x256 pikseli natomiast drugi ma wymiary 64x64 pikseli);
- mogą być „nałożone” na inną siatkę (piksele dwóch identycznych rastrów nie znajdują się na siebie);
- mogą na siebie nachodzić (identycznie jak poligony w warstwach wektorowych); własność istotna przy zamianie wektorów na rastry;

Rastry

Powysze własności pozwalają na łatwą integrację danych rastrowych z wektorowymi. Z drugiej strony umożliwiają tworzenie powierzchni, które nie są odpowiednie do analiz przestrzennych.

Najlepiej aby tafle rastra tworzyły ciągłą powierzchnię, miały ten sam rozmiar, były nałożone na tą samą siatkę i nie nakładały się na siebie ☺

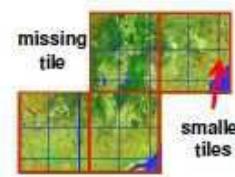


Tafle (Kafle)

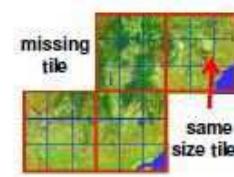
W PostGIS Raster mamy możliwość przechowywania **nieregularnych tafli** oraz **nakładających się powierzchni**.



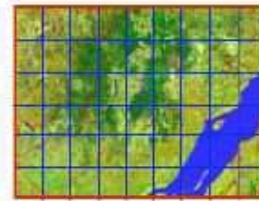
a) warehouse of untiled and unrelated images (4 images)



b) irregularly tiled raster coverage (36 tiles)



c) regularly tiled raster coverage (36 tiles)



d) rectangular regularly tiled raster coverage (54 tiles)

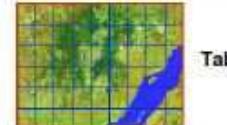
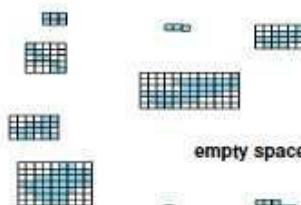


Table 1



Table 2



f) rasterized geometries coverage (9 lines in the table)

W przeciwnieństwie do typowych danych rastrowych użytkownik bazy może usuwać/dodawać wiersze tabeli.

Metadane

Raster jest złożonym typem danych, który dodatkowo opisują metadane. Lista metadanych oraz funkcje, które zwracają daną właściwość prezentuje poniższa tabela.

Atrybut	Funkcja
Number of bands	ST_NumBands()
Width	ST_Width()
Height	ST_Height()
Pixelsizex	ST_PixelSizeX()
Pixelsizey	ST_PixelSizeY()
upperleftx	ST_UpperLeftX()
upperlefty	ST_UpperLeftY()
srid	ST_SRID()
pixeltype	ST_BandPixelType()
values[]	ST_Value()
hasnodatavalue	ST_BandHasNoDataValue()
path	ST_BandPath()

Import

Import/eksport danych



Import: [raster2pgsql](#) (import pojedynczego rastra lub zestawu rastrów do tabeli)

- c (tworzenie nowej tabeli i wczytanie do niej rastra)
- a (dołączanie rastra do istniejącej tabeli)
- C (definiowanie ograniczeń dla rastra)
- s (przypisywanie wartości SRID do rastra)
- b (definiowanie kanału rastra)
- t (podział rastra na tafle)
- N (wartość dla braku danych)
- I (tworzenie indeksów)

```
raster2pgsql raster_options_go_here raster_file someschema.sometable > out.sql
```

Przechowywanie rastrów

Przechowywanie rastrów: „IN-DB” oraz „OUT-DB”

„IN-DB” - użytkownik ładuje w całości raster do tabeli, dzieląc go na tafle.

```
raster2pgsql raster_options_go_here raster_file someschema.sometable > out.sql  
psql -d database_name -f out.sql
```

Zalety przechowywania „in-db”:

- wysoka wydajność obliczeniowa,
- zapewniony wielodostęp do danych,
- możliwość analizy wektor/raster.

Wady przechowywania „in-db”:

- utrudniona skalowalność,
- wszystkie operacje wiążą się z koniecznością ładowania/eksportowania danych z bazy.

Przechowywanie rastrów

Przechowywanie rastrów: „IN-DB” oraz „OUT-DB”

„**OUT-DB**” - użytkownik posiada możliwość ‘rejestracji’ podstawowych metadanych obrazu rastrowego zapisanego w systemie bez konieczności wczytywania jego wartości do bazy danych.

```
raster2pgsql.py -r c:/imageset/landsat/image1.tif -t landsat -R
```

Zalety przechowywania „out-db”:

- możliwość bezpośredniego dostępu do danych rastrowych przez różnego rodzaju aplikacje (read-only),
- możliwość wykorzystania funkcji PostGIS dla zarejestrowanych rastrów,
- skalowalność.

Wady przechowywania „out-db”:

- wolniejsze analizy PostGIS,
- możliwa edycja danych tylko przez jednego użytkownika.

Funkcje

Funkcje w PostGIS Raster umożliwiają m.in.:

- import/ eksport danych;
- edycję danych;
- wykonywanie analiz raster-wektor;
- wyznaczenie statystyk dla rastrów
- tworzenie nowych rastrów

Tworzenie rastrów

raster **ST_MakeEmptyRaster**(integer *width*, integer *height*, float8 *upperleftx*, float8 *upperlefty*, float8 *pixelsize*);

raster **ST_MakeEmptyRaster**(integer *width*, integer *height*, float8 *upperleftx*, float8 *upperlefty*, float8 *scalex*, float8 *scaley*, float8 *skewx*, float8 *skewy*, integer *srid=unknown*);

raster **ST_Band**(raster *rast*, integer *nband*);

- służy do tworzenia nowego rastra z już istniejącego



oryginalny

2 kanały

1 kanał

Tworzenie rastrów

Przykład - wczytywanie zobrazowań multispektralnych Landsat 8.

```
mypath\raster2pgsql.exe -s 3763 -N -32767 -t 128x128 -I -C -M -d  
mypath\rasters\Landsat8_L1TP_RGBN.TIF rasters.landsat8  
| psql -d postgis_raster -h localhost -U postgres -p 5432
```

Wynik:

tabela rasters.landsat8, podzielona na tafle o wielkości 128x128, zawierająca 11 kanałów spektralnych.

Tworzenie rastrów

Przykład - tworzenie tabeli z istniejącej tabeli zawierającej raster.

```
CREATE TABLE schema name.landsat nir AS  
SELECT rid, ST_Band(rast, 4) AS rast  
FROM rasters.landsat8;
```

ST_Band zwraca tylko jeden kanał nr 4 - bliską podczerwień.

Tworzenie rastrów

raster ST_AsRaster(geom) – zamiana typu geometry na typ raster.

Istnieją trzy możliwości ustalenia parametrów dla tworzonego obiektu typu raster:

- pobranie parametrów z istniejącego rastra referencyjnego,
- użytkownik ustala takie parametry jak: scalex, scaley, skewx, skewy, natomiast wartości width i height ustalane są automatycznie na podstawie zasięgu obiektu typu geometry,
- polecenie nie działa dla bardziej złożonych geometrii takich jak krzywe gładkie, czy TIN.

Analizy

$$[a] = [b] + [c]$$
$$\begin{matrix} 7 & 6 \\ 2 & 4 \end{matrix} = \begin{matrix} 4 & 2 \\ 1 & 3 \end{matrix} + \begin{matrix} 3 & 4 \\ 1 & 1 \end{matrix}$$

ST_SummaryStats(raster)

zwraca zbiór statystyk opisowych (min, max, suma, wariancja, odchylenie standardowe, liczebność) dla danego rastra.

ST_Reclass(raster, text reclassexpr, text pixeltype) – reklasyfikacja rastra.

ST_MapAlgebra(raster, band, expression, nodatavalueexpr, pixeltype)

- algebra rastrów,
- umożliwia wykonywanie operacji matematycznych na odpowiadających sobie pikselach,
- w przypadku różnej rozdzielczości przestrzennej rastrów automatycznie konwertuje do rozdzielczości większej.

Analizy

```
CREATE TABLE schema_name.porto_ndvi AS
WITH r AS (
    SELECT a.rid, ST_Clip(a.rast, b.geom, true) AS rast
    FROM rasters.landsat8 AS a, vectors.porto_parishes AS b
    WHERE b.municipality ilike 'porto' and
    ST_Intersects(b.geom, a.rast)
)
SELECT
    r.rid, ST_MapAlgebra(
        r.rast, 1,
        r.rast, 4,
        '([rast2.val] - [rast1.val]) / ([rast2.val] +
[rast1.val]):float', '32BF'
    ) AS rast
FROM r;
```

$$[a] = [b] + [c]$$

7	6	=	4	2	+	3	4
2	4		1	3		1	1

boolean **ST_IsEmpty**(raster)

- zwraca TRUE jeżeli WIDTH=0 oraz HEIGHT=0
-

integer **ST_SRID**(raster)

- zwraca wartość SRID

Uwaga: w PostGIS 2.0 wartość SRID równa 0 oznacza niezdefiniowany układ
(wcześniej była to wartość -1)

integer **ST_NumBands**(raster)

- zwraca liczbę kanałów obiektu typu raster
-

record **ST_MetaData**(raster)

rid	upperleftx	upperlefty	width	height	scalex	scaley	skewx	skewy	srid	numbands	
1	0.5	0.5	10	20	2	3			0	0	0
2	3427927.75	5793244	5	5	0.05	-0.05			0	0	3

text **ST_GeoReference**(raster)

- zwraca informacje o geoodniesieniu rastra w zapisie (GDAL lub ESRI)

double precision **ST_PixelHeight** (raster)

double precision **ST_PixelWidth**(raster)

float **ST_Rotation** (raster)

float **ST_UpperLeftX**(raster)

float **ST_UpperLeftY**(raster)

float8 **ST_SetRotation**(raster *rast*, float8 *rotation*);

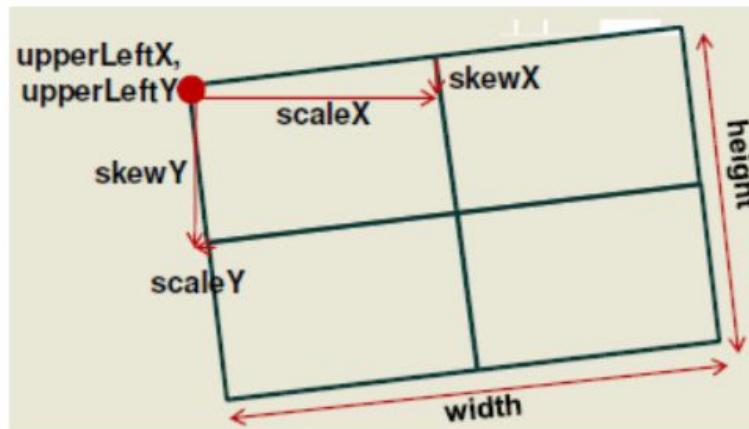
- ustawianie rotacji rastra w radianach

raster **ST_SetSRID**(raster *rast*, integer *srid*);

- ustalanie wartości SRID /to nie jest transformacja układów! /

raster **ST_SetBandNoDataValue**(raster *rast*, double precision *nodatavalue*);

- ustawienie wartości, która będzie rozumiana jako brak danych;



raster **ST_Clip** (raster *rast*, geometry *geom*, boolean *crop*);

- przycinanie rastra za pomocą obiektu typu geometry
- *crop=true* (nowy raster ma zasięg jak część wspólna *rast* i *geom*)
- *crop=false* (nowy raster ma zasięg taki jak raster pierwotny)



raster **ST_Reclass** (raster *rast*, integer *nband*, text *reclasseexpr*, text *pixeltype*, double precision *nodataval=NULL*);

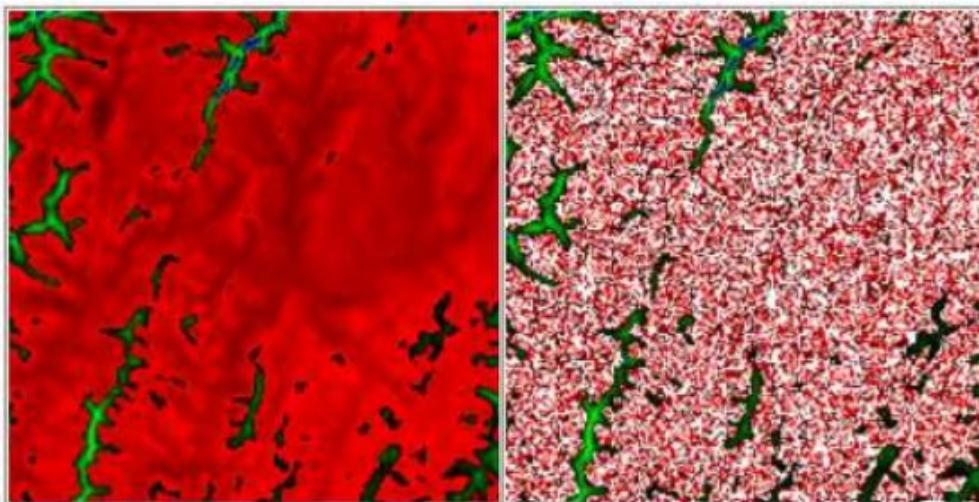
- reklasyfikacja rastra (wybranych kanałów)
- raster wynikowy ma to samo geoodniesienie, wysokość i szerokość jak raster reklasyfikowany

reclasseexpr

1. $[a-b] = a \leq x \leq b$
2. $(a-b) = a < x \leq b$
3. $[a-b) = a \leq x < b$
4. $(a-b) = a < x < b$

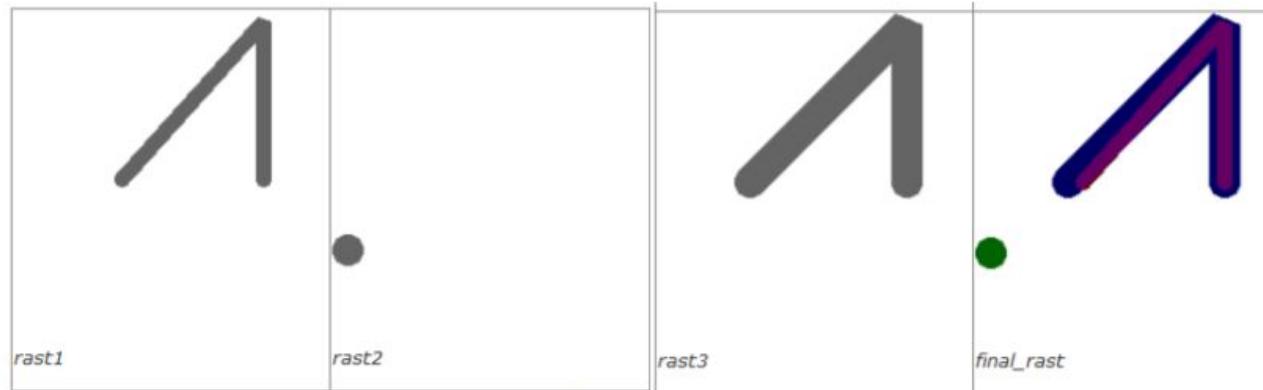
raster **ST_MapAlgebraExpr**(raster *rast*, integer *band*, text *pixeltype*, text *expression*, double precision *nodataval=NULL*);

- tworzy jest jednokanałowy raster na podstawie 1 kanału rastra wejściowego
- wartość domyślna kanału to 1
- raster wynikowy ma to samo geoodniesienie, wysokość i szerokość jak raster wejściowy
- wykorzystywane są operacje algebraiczne PostgreSQL



```
raster ST_MapAlgebraExpr(raster rast1, raster rast2, text expression, text  
pixeltype=same_as_rast1_band, text extenttype=INTERSECTION, text  
nodata1expr=NULL, text nodata2expr=NULL, double precision nodatanodataval=NULL);
```

- tworzony jest jednokanałowy raster na podstawie kanałów dwóch wejściowych rastrów
- wykorzystywane są operacje algebraiczne PostgreSQL (np. [rast1]+[rast2])
- wartość domyślna kanałów to 1
- zasięg wynikowego rastra: INTERSECTION, UNION, FIRST, SECOND
- położenie rastra wynikowego będzie takie jak rastra pierwszego



geometry **ST_Polygon**(raster rast, integer band_num=1);

- zwraca poligon, który zawiera piksele rastra różne od oznaczenia „no data value”

```
SELECT ST_AsText(ST_Polygon(rast)) As geomwkt
FROM dummy_rast
WHERE rid = 2;

geomwkt
-----
POLYGON((3427927.8 5793243.75,3427927.75 5793243.75,3427927.75 5793243.8,3427927.75 5793243.85,3427927.75 5793243.9,
3427927.75 5793244,3427927.8 5793244,3427927.85 5793244,3427927.9 5793244,3427928 5793244,3427928 5793243.95,
3427928 5793243.85,3427928 5793243.8,3427928 5793243.75,3427927.85 5793243.75,3427927.8 5793243.75))
```

raster **ST_Slope**(raster rast, integer band, text pixeltype);

- tworzony jest raster reprezentujący nachylenie stoków;

geometry **ST_Envelope**(raster rast);

- zwracany jest poligon reprezentujący zasięg rastra

```
SELECT rid, ST_AsText(ST_Envelope(rast)) As envgeomwkt
FROM dummy_rast;

rid | envgeomwkt
----+
1 | POLYGON((0 0,20 0,20 60,0 60,0 0))
2 | POLYGON((3427927 5793243,3427928 5793243,
            3427928 5793244,3427927 5793244, 3427927 5793243))
```

raster **ST_Union**(setof raster *rast*);

- łączy tafle rastra w jeden raster jednokanałowy;
- wartość domyślna kanałów to 1;
- zasięg rastra wynikowego jest równy zasięgowi wszystkich łączonych tafli

boolean **ST_Intersects**(raster *rasta* , integer *nbanda* , raster *rastb* , integer *nbandb*);

boolean **ST_Intersects**(raster *rast* , integer *nband* , geometry *geommin*);

- zwraca TRUE jeżeli raster przestrzennie „przecina się” z innym rastrem lub obiektem typu geometry

bytea **ST_AsBinary**(raster rast);

```
SELECT ST_AsBinary(rast) As rastbin  
FROM dummy_rast WHERE rid=1;
```

rastbin

```
-----  
\001\000\000\000\000\000\000\000\000\000@\\000\000\000\000\000\000\010@\\  
000\000\000\000\000\000\340?\000\000\000\000\000\340?\000\000\000\000\000\000\00  
0\000\000\000\000\000\000\000\000\000\000\000\012\000\000\000\012\000\024\000
```

bytea **ST_AsJPEG**(raster rast, integer nband, integer quality);

- jakość: wartość od 0 do 100

bytea **ST_AsPNG**(raster rast, integer nband, integer compression);

- kompresja: wartości od 1 do 9 (im większa wartość tym większa kompresja)

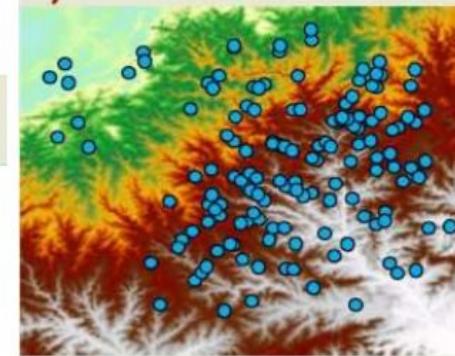
bytea **ST_AsTIFF**(raster rast, text compression="", integer srid=sameassource);

- zapamiętywana informacja o geoodniesieniu

Wyznaczenie wartości wysokości n.p.m. dla danych punktów na podstawie rastra reprezentującego DEM.

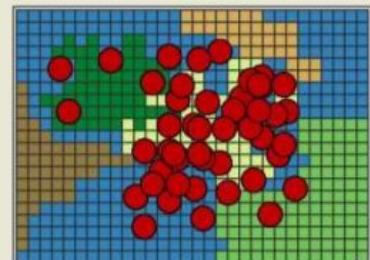
```
- SELECT pointID, ST_Value(rast, geom) elevation  
FROM lidar, srtm WHERE ST_Intersects(geom, rast)
```

double precision ST_Value(raster *rast*, geometry *pt*, boolean
exclude nodata value=true);



Obliczenie średniej temperatury dla każdego poligona.

```
SELECT bufID, (gv).geom buffer, (gv).val temp  
FROM (SELECT bufID, ST_Intersection(geom, rast) gv  
      FROM buffers, temperature  
     WHERE ST_Intersects(geom, rast))
```



buffers	
geom	pointID
polygon	24
polygon	46
polygon	31
polygon	45
...	...

temperature	
raster	
raster	
raster	
polygon	53
raster	
raster	
...	...

=

result		
geom	pointID	temp
polygon	24	11.2
polygon	53	13.4
polygon	24	15.7
polygon	23	14.2
...

