Swiss PGDay 2016, 24. Juni 2016, HSR Rapperswil

POSTGIS

Überblick, Tips und Tricks

Stefan Keller

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Topics

- What is PostGIS?
- Spatial table
- Example with historical data
- OGC queries in PostGIS

- Spatial Joins
- OGC
- Layers / Layerss
- Indexing

About Spatial Databases…

- **Databases**
  - Types: string, float, date
  - Indexes: b-tree
  - Functions: strlen(string), pow(float, float), now()

- **Spatial Databases**
  - Spatial Types: geometry, geography
  - Spatial Indexes: r-tree, quad-tree, kd-tree
  - Spatial Functions: ST_Length(geometry), ST_X(geometry), etc.

Spatial Databases and GIS

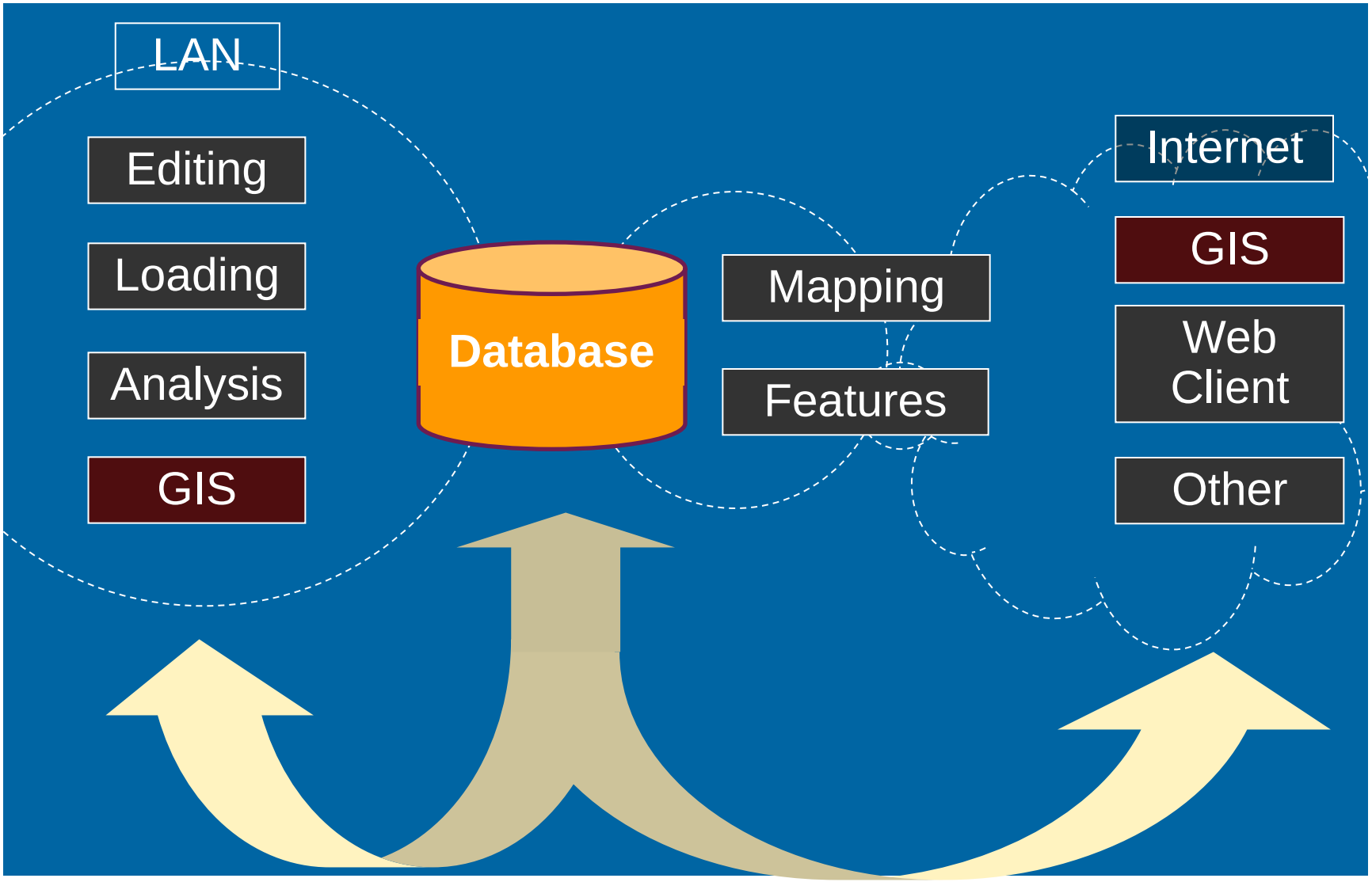


Image from Paul Ramsey Refractions Research

ABOUT POSTGIS

PostGIS – A PostgreSQL extension

- Delivered with PostgreSQL installation
- More rigid license: PostgreSQL => MIT alike, PostGIS => GPL
- Compliant with standards (like PostgreSQL)
- Supports PostgeSQL's 'native types': point, line, box(!), path, polygon, and circle geometric types
- Supports OGC types ("OGS Simple Features for SQL"): point, linestring, polygon, multipoint, etc.
- >300 functions
- Spatial index: GiST (Generalized Search Tree), SP-GiST, KNN

PostGIS S/W components

- **Bulk loader (mostly command line interfaces CLI):**
  - Vector data: shp2pgsql (CLI and plugin for pgAdmin III)
  - Raster data: raster2pgsql (CLI)

  - TIPP: gdal / ogr (CLI) from gdal.org
  - TIPP: geoconverter.hsr.ch (free Webapp)

- **Database Drivers**
  - Open Database Connectivity ODBC connectivity.
  - Java Database Connectivity (JDBC)

S/W internally used by PostGIS (and other FOSS)

- **PROJ.4: Open source library that provides coordinate reprojection to convert between geographic coordinate systems**
- **GEOS (Geometry Engine, Open Source): Open source library to perform operations on geometry OGC types**
- **CGAL/SFCGAL (Geometry Engine, Open Source): Alternative to GEOS**

PostGIS History

2001, May	0.1	Objects / Indexes
2001, July	0.5	Functions
2003, November	0.8	OGC SFSQL
2005, April	1.0	Lightweight Geometry
2010, February	1.5	Geography
2012, April	2.0	Raster
2013, August	2.1	Speed/polish

Ramsey, PostGIS Frenzy, 2015

PostGIS and standards

1. PostGIS implements and is compliant with the “OGC’s Simple_Features for SQL” standard
2. PostGIS supports all OGC types: Point, Line, Polygon, MultiPoint, MultiLine, MultiPolygon, GeometryCollection and operations on those types
3. PostGIS uses OGC Well-Known Text (WKT) format for I/O and constructors

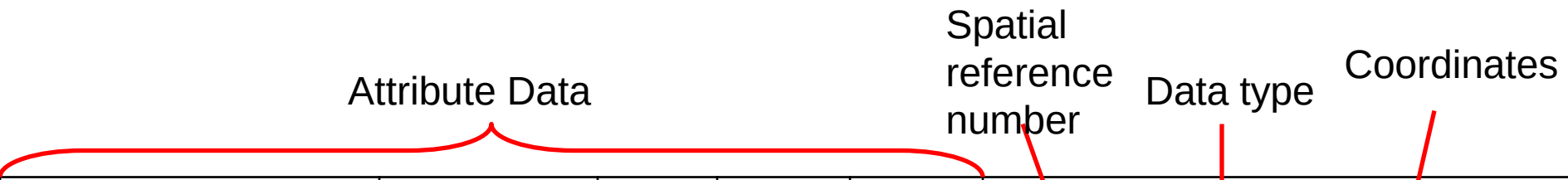
Well Known Text (WKT)

- Geometry types from OGC standard for Simple Features:
- “POINT(50 100)”
- “LINESTRING (10 10, 20 20)”
- “POLYGON ((0 0, 5 5, 5 0, 0 0))”
- “MULTIPOINT ((1 1), (0 0))”
- “MULTILINESTRING ((...), (...))”
- “MULTIPOLYGON ((...), (...))”

- Supports also Curves!

PostgreSQL/PostGIS

- **The data is stored in a relatively simple format with geometry stored binary. It can be viewed as WKT using AsTextgeom), SELECT name, city, hrs, status, AsText(geom) from mytable;**



name	city	hrs	status	st_fed	geom
Brio Refining	Friendswood	50.38	active	Fed	SRID=32140;POINT(968024.87474318 4198600.9516049)
Crystal Chemical	Houston	60.9	active	Fed	SRID=32140;POINT(932279.183664999 4213955.37498466)
North Cavalcade	Houston	37.08	active	Fed	SRID=32140;POINT(952855.717021537 4223859.84524946)
Dixie Oil Processors	Friendswood	34.21	active	Fed	SRID=32140;POINT(967568.655313907 4198112.19404211)
Federated Metals	Houston	21.28	active	State	SRID=32140;POINT(961131.619598681 4220206.32109146)

How does it work?

- Spatial data is stored using the coordinate system of a particular projection
- That projection is referenced with a Spatial Reference Identification Number (`SRID`)
- This number (e.g. 21781, meaning EPSG:21781) relates to another table (`spatial_ref_sys`) which holds all of the spatial reference systems available
- This allows the database to know what projection each table is in, and if need be, re-project from those tables for calculations or joining with other tables

TABLES WITH GEOMETRIES AND SYSTEM TABLES

Creating a spatial table: Basic steps

- **Creating a table with at least an attribute of type geometry**
  - CREATE TABLE my_pois (
    gid serial PRIMARY KEY,
    geom GEOMETRY(POINT, 21781,2),
    name TEXT
    );
- **Beware old style**
  - CREATE TABLE my_pois (
    gid serial PRIMARY KEY,
    name TEXT
    );
  - SELECT AddGeometryColumn('public',‘my_pois',’geom',‘21781','POINT',2);
- **TIPP:**
  - We recommend “geom” or “geometry” as attribute name (sometimes see also “the_geom”)

Creating a spatial table, step 1

- Note a system generated identified (gid) is used as the primary key
- PostgreSQL/PostGIS will respond:
- NOTICE: CREATE TABLE will create implicit sequence "my_pois_gid_seq" for serial column "my_pois.gid"
- NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index “my_pois_pkey" for table “my_pois"

Creating a spatial table, step 1

Examine the table (\d):

```
      Table "public.my_pois"
 Column |   Type    |    Modifiers
--------+-----------+-----------------------
 gid    | integer | not null default
     nextval('my_pois_gid_seq'::regclass)
Reset of data
Indexes:
"my_pois_pkey"PRIMARY KEY,btree (gid)
```

Creating a spatial table, step 2

- Step 2 are PostGIS internal steps…
- As column “geom” of type GEOMETRY was added, PostGIS will automatically generate integrity constraints
- This accessed the `geometry_columns` system table (details later).

Creating a spatial table, step 2

First system generated constraint

```
ALTER TABLE my_pois

  ADD CONSTRAINT enforce_dims_geom CHECK
  (ndims(geom) = 2);
```

Creating a spatial table, step 2

Second system generated constraint

```
ALTER TABLE my_pois

  ADD CONSTRAINT enforce_geotype_geom
  CHECK (geometrytype(geom) =
  'POINT'::text OR geom IS NULL);
```

Creating a spatial table, step 2

Third system generated constraint

```
ALTER TABLE my_pois

  ADD CONSTRAINT enforce_srid_geom CHECK
  (srid(geom) = 21781);
```

Creating a spatial table, step 2

The Primary Constraint was created in step1

```
CONSTRAINT my_pois_pkey PRIMARY KEY(gid);
```

Creating a spatial table, step 3

- Given table openstreetmap_points, insert all Zoo's into table my_pois:

```
INSERT INTO my_pois (geom, name)
  SELECT way, name
  FROM openstreetmap_points
  WHERE tags @> hstore('tourism', 'zoo');
```

TIPP: Creation of geometry constructors

- **ST_GeomFromText('POINT(-71.06 42.28)') -- Preferred simplest text form without SRID**
- **ST_GeomFromText('POINT(-71.06 42.28)', 4326) -- Preferred for text form with SRID**
- **ST_MakePoint(-71.06, 42.28, 4326) -- Preferred symbolic form (Hint: returns WKT, not EWKT)**
- **ST_SetSRID(ST_MakePoint(-71.06, 42.28),4326) -- Preferred symbolic form with EWKT**

Additional TIPP: Create Polygon given Bounding Box (BBox)

- **ST_Transform(ST_MakeEnvelope(8.795611, 46.872886, 9.674135, 47.675419, 4326), 3857)**
- **ST_Transform(ST_SetSRID(ST_Envelope('LINESTRING(8.795611 46.872886, 9.674135 47.675419)'::geometry),4326), 3857)**
- **ST_Transform(ST_SetSRID('BOX(8.795611 46.872886, 9.674135 47.675419)'::box2d, 4326), 3857)**
- **ST_Transform(ST_SetSRID('BOX3D(8.795611 46.872886, 9.674135 47.675419)'::box3d, 4326), 3857)**

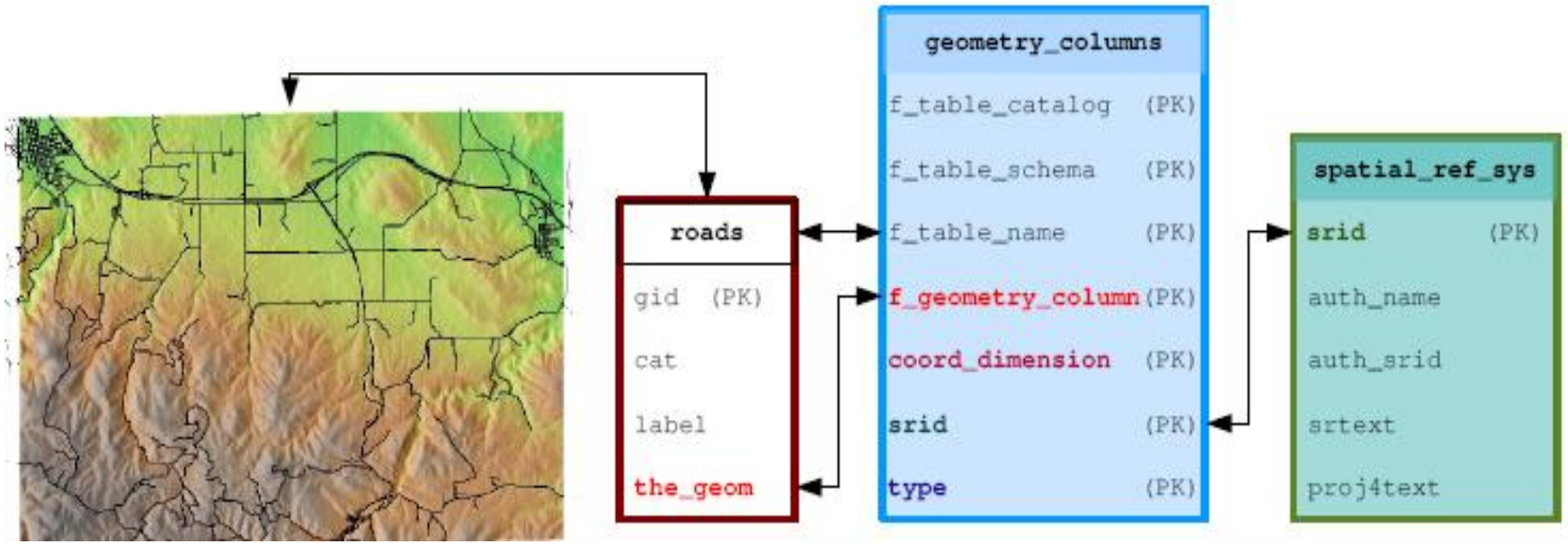
- **See also PostGIS Terminal : http://giswiki.hsr.ch/PostGIS_-_Tipps_und_Tricks#PostGIS-Daten_laden**

POSTGIS SYSTEM TABLES

PostGIS System Tables (OGC – Metadata tables)

To conserve metadata consistency, OGC standard need two tables to colelct information about georeferenced data set.

```
CREATE TABLE roads( gid serial NOT NULL, cat int8, label varchar(80), the_geom geometry,
      CONSTRAINT roads_pkey PRIMARY KEY (gid),
      CONSTRAINT enforce_dims_the_geom CHECK (ndims(the_geom) = 2),
      CONSTRAINT enforce_geotype_the_geom CHECK (geometrytype(the_geom) = 'MULTILINESTRING'::text OR the_geom IS
NULL),
      CONSTRAINT enforce_srid_the_geom CHECK (srid(the_geom) = 26713))
CREATE INDEX roads_the_geom_gist ON roads USING gist(the_geom);
```



geometry_columns table/view

```
Column                        |            Type                  |Modifiers
------------------------------+----------------------------------+----------
 f_table_catalog              | character varying(256) | not null
 f_table_schema               | character varying(256) | not null
 f_table_name                 | character varying(256) | not null
 f_geometry_column            | character varying(256) | not null
 coord_dimension              | integer                | not null
 srid                         | integer                | not null
 type                         | character varying(30)  | not null
Indexes:
    "geometry_columns_pk" PRIMARY KEY, btree (f_table_catalog, f_table_schema, f
_table_name, f_geometry_column)
```

This table/view allows PostgreSQL/PostGIS to keep track of actual user spatial tables.

spatial_ref_sys table

- Displaying a spherical earth on a flat surface requires a projection.
- This table uses a standard numbering, called the EPSG, to describe various projections.
- Examine the details for a particular projection e.g. in psql:

```
select * from spatial_ref_sys where srid=21781;
```

- TIPP: See also http://epsg.io/

spatial_ref_sys table

```
\d spatial_ref_sys
 Column     |            Type             | Modifiers
------------+-----------------------------+-----------
 srid       | integer                     | not null
 auth_name  | character varying(256)      |
 auth_srid  | integer                     |
 srtext     | character varying(2048)     |
 proj4text  | character varying(2048)     |
Indexes:
    "spatial_ref_sys_pkey" PRIMARY KEY, btree (srid)
```

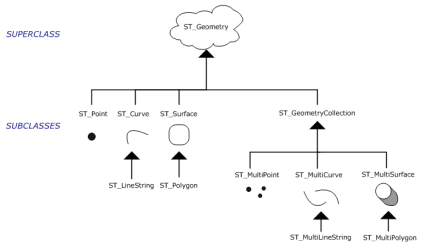
SPATIAL DATA TYPES AND OGC

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Geometry Object Model

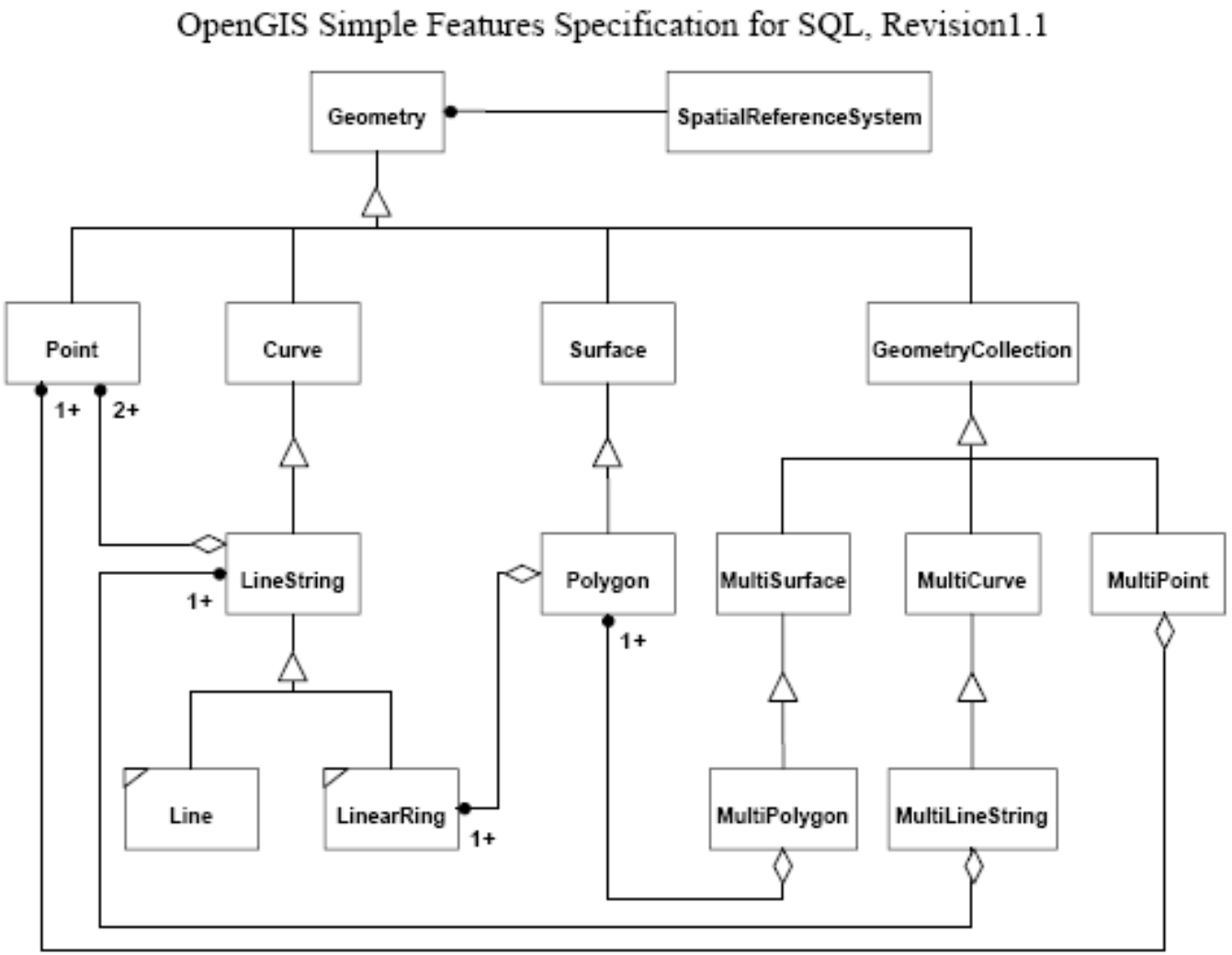
- **is “abstract” (or conceptual) part of the OGC suite of standards**
- **It defines geometries and operations on them.**
- **is conceptual model independent of SQL or any other language**
  - Abstract class: Geometry
  - Instantiable subclasses in include:
  - Points which represent points in 2-dimensional space
  - Lines are linear edges between two points
  - Linestrings are connected lines (end-point is start-point of
  - Linear Rings are ’closed’ Linestrings (last ’end-point’ is first start-point)
  - Polygons Surface within a Linear Ring, potentially excluding inner Linear Rings
  - Uniform Collections of concrete Types



Spatial Types – OGC Simple Features for SQL

- An association represents a family of links.
- Aggregation is a has-a relationship; aggregation is more specific than association.
- Composition is a stronger variant of the "has a" association relationship, it has a strong lifecycle dependency between instances of the container class and instances of the contained class(es).
- The standard does not mention UML composition, but explicitly mentions the “owned by” black dot. Multiplicity in UML allows to specify cardinality - i.e. number of elements - of some collection of elements. In the standards will ill take the open diamond to represent the part-of relation.
- Inheritance represents an is-a relation.

Spatial Types – OGC Simple Features for SQL

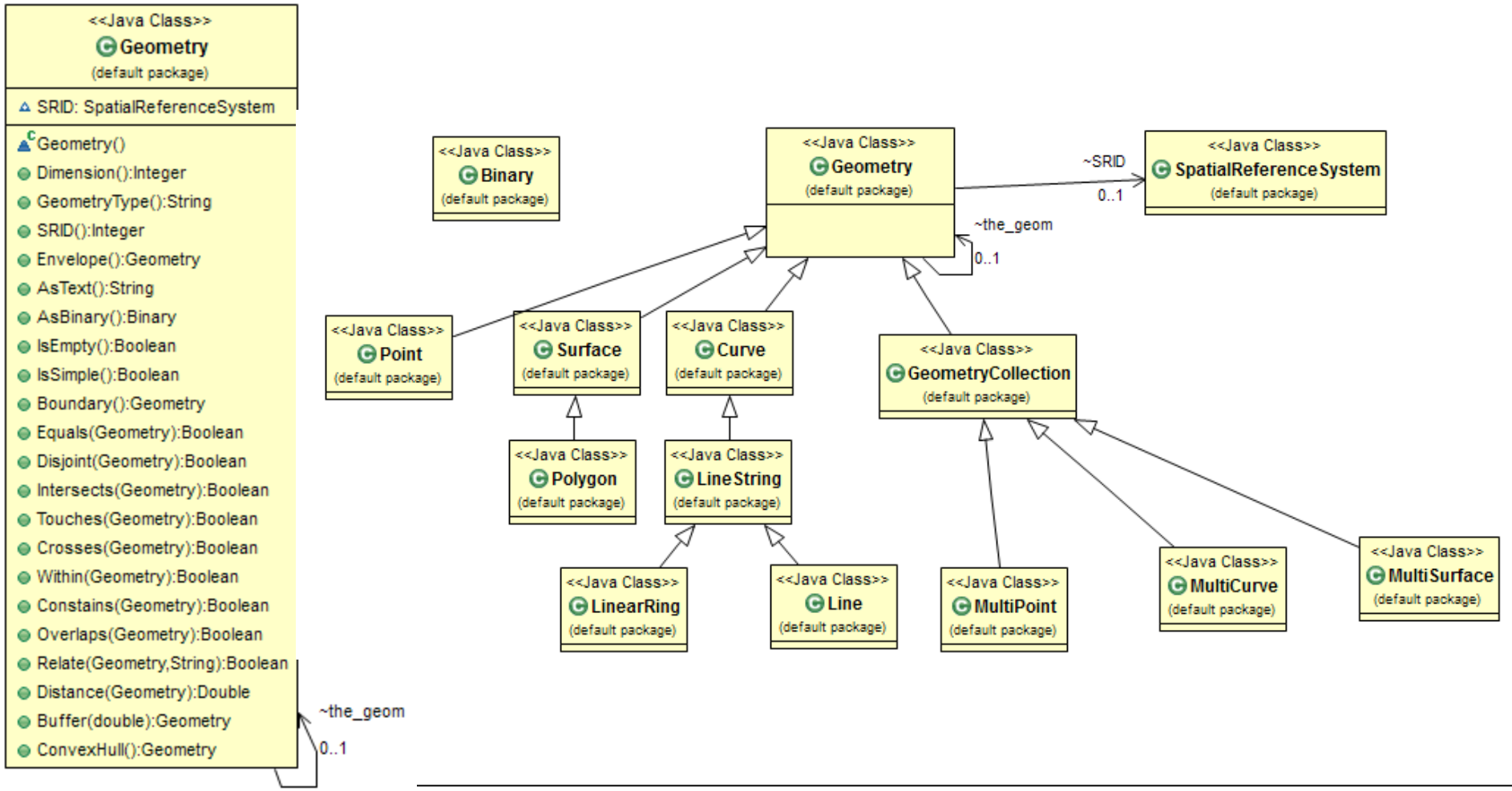


OGC Simple Feature Types: Operators

OGC spatial operators defined on the class geometry

Classes	*Operators*	*Operator Functions*
Basic Operators	Spatial Reference	Returns the reference system of the geometry
	Envelope	Returns the minimum bounding rectangle of the geometry
	Export	Converts the geometry into a different representation
	IsEmpty	Tests if the geometry is the empty set or not
	IsSimple	Returns TRUE if the geometry is simple
	Boundary	Returns the boundary of the geometry
Topological Operators	Equal	Tests if the geometries are spatially equal
	Disjoint	Tests if the geometries are disjoint
	Intersect	Tests if the geometries intersect
	Touch	Tests if the geometries touch each other
	Cross	Tests if the geometries cross each other
	Within	Tests if a geometry is within another geometry
	Contain	Tests if a given geometry contains another geometry
	Overlap	Tests if a given geometry overlaps another given geometry
	Relate	Returns TRUE if the spatial relationship specified by the 9-Intersection matrix holds
Spatial Analysis Operators	Distance	Returns the shortest distance between any two points of two given geometries
	Buffer	Returns a geometry that represents all points whose distance from the given geometry is less than or equal to a specified distance
	ConvexHull	Returns the convex hull of a given geometry
	Intersection	Returns the intersection of two geometries
	Union	Returns the union of two geometries
	Difference	Returns the difference of two geometries
	SymDifference	Returns the symmetric difference (i.e. the logical XOR) of two geometries

OGC Simple Feature Types: Methods and Structures



OGC Simple Features for SQL

- The OGC SF (similar to ISO 19125-1) describes 2-D geometry with linear interpolation between vertices.
- The simple feature model consists of a root class `Geometry` and its specific subclasses `Point`, `Curve`, `Surface`, `GeometryCollection`.
- The class `Geometry` collection has the subclasses `Multipoint`, `Multicurve`, `MultiSurface`.

OGC Simple Features for SQL (*)

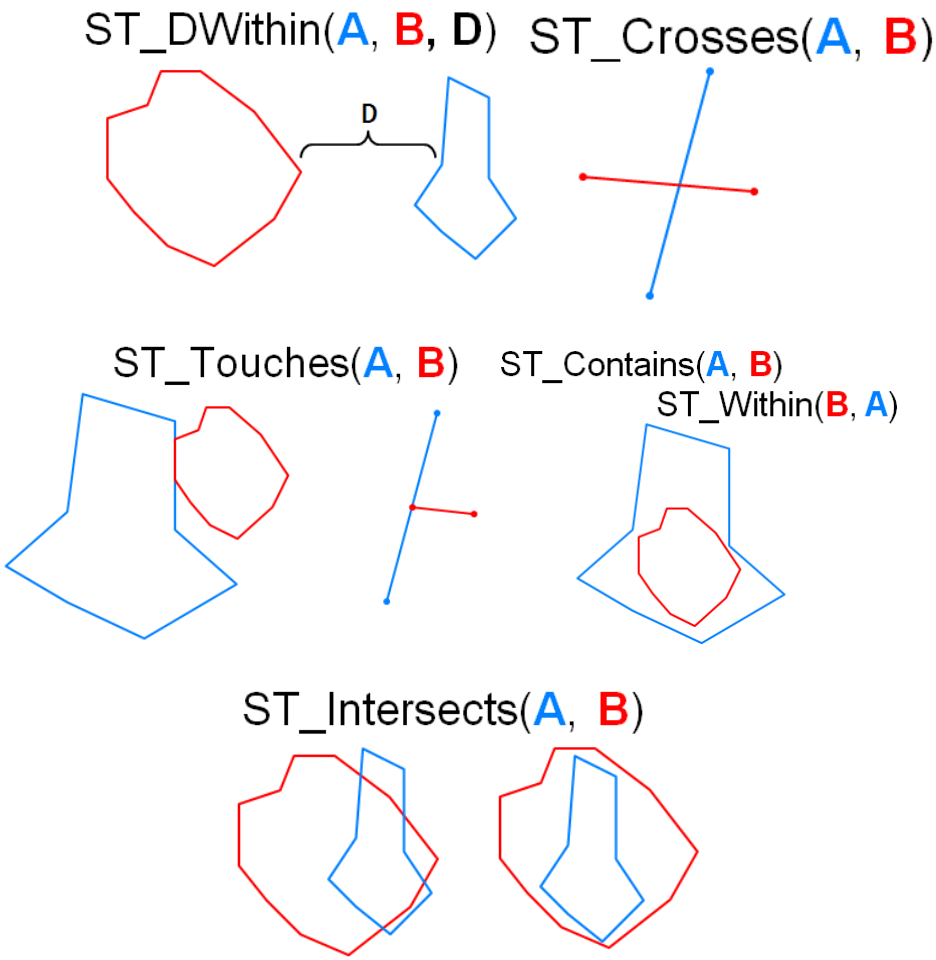
- **Basic Methods on Geometry**
  - Describes the dimensions and reference system (SRID) of the geometry.
  - Operations include Dimension, GeometryType, , conversions AsText, AsBinary, tests on geometry include IsEmpty, IsSimple. Operations that return geometry Boundary, Envelope returns bounding box
- **Methods for testing Spatial Relations between geometric objects**
  - These polymorphic methods check relations on the generic or super class GEOMETRY and usually return a Boolean. Main methods Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps, Relate(testing for intersections between the Interior, Boundary and Exterior of the two geometries)
- **Methods that support Spatial Analysis**
  - A set of geometric and 'metric' methods. Methods calculate distances and areas with respect to the spatial reference system of this Geometry. Methods include Distance, Buffer, ConvexHull, Intersection, Union, Difference, SymDifference.
- **Geometry Collection**
  - A GeometryCollection is a geometry that is a collection of 1 or more geometries. All the elements in a GeometryCollection must be in the same Spatial Reference. Subclasses of GeometryCollection may restrict membership based on dimension and may also place other constraints on the degree of spatial overlap between elements. Methods
  - NumGeometries( ):Integer—Returns the number of geometries in this GeometryCollection.
  - GeometryN(N:integer):Geometry—Returns the Nth

OGC Spatial Relations

- **Equals** – same geometries
- **Disjoint** – geometries share common point
- **Intersects** – geometries intersect
- **Touches** – geometries intersect at common boundary
- **Crosses** – geometries overlap
- **Within**– geometry within
- **Contains** – geometry completely contains
- **Overlaps** – geometries of same dimension overlap
- **Relate** – intersection between interior, boundary or exterior

OGC Spatial Operations & Relations

- **The most typical spatial relationships (and it's opposites) got own functions, like:**
- **ST_Within != ST_Contains (*)**
- **ST_Covers != ST_CoveredBy**
- **ST_Intersects != ST_Disjoint**
- **(*) Note: Prefer ST_Covers over ST_Contains if lines on boundaries count as „inside“ (Source: Martin Davis: http://lin-ear-th-inking.blogspot.ch/2007/06/subtleties-of-ogc-covers-spatial.html)**



Spatial Methods

- **Distance** – shortest distance
- **Buffer** – geometric buffer
- **ConvexHull** – smallest convex polygon geometry
- **Intersection** – points common to two geometries
- **Union** – all points in geometries
- **Difference** – points different between two geometries
- **SymDifference** – points in either, but not both of input geometries

TIPP: Overlay (1 von 2)

- **Gegeben die beiden Tabellen:**
- **gemeinden_bl: Gemeinden Kt.BL aus Vermessung mit den Attributen gem_id_bfs, name, geom(MultiPolygon,21781)**
- **gemeinden_bl_simpl - Gemeinden Kt.BL aus Raumplanung, von Vermessung digitalisiert und mit zusätzlichen Polygonen**
- **Gesucht: Polygon-Verschnitt (Overlay, Intersection)**

- **Vorbereitungen:**
  - DB mit PostGIS Extension
  - CREATE SEQUENCE my_sequence MINVALUE 0;
  - SELECT setval('my_sequence', 0); -- Reset sequence:

TIPP: Overlay – the robust way (2 von 2)

```
CREATE TABLE gemeinden_bl_intersected_multi AS
SELECT
  nextval('my_sequence') AS id,
  a.id AS aid,
  b.id AS bid,
  a.name AS name,
  a.gem_id_bfs AS gem_id_bfs,
  round( ST_Area( ST_Intersection(a.geom,b.geom) ) )::int
    AS area,
  ST_Intersection(a.geom, b.geom) AS geom
FROM gemeinden_bl AS a
INNER JOIN gemeinden_bl_simpl AS b
ON ST_Intersects(a.geom, b.geom)
WHERE NOT ST_IsEmpty(ST_Buffer(ST_Intersection(a.geom,
b.geom), 0.0))
AND ST_Area(ST_Intersection(a.geom,b.geom))>=50000.0;
-- m2, 5 Hektaren, 223m*223m
```

POSTGIS FEATURE FRENZY
BY PAUL RAMSEY

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

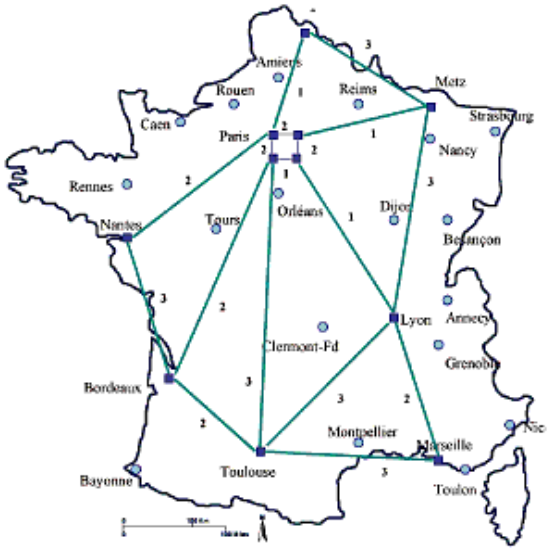
FHO Fachhochschule Ostschweiz

Slides 64 – 109 (Presented at conference FOSS4G NA 2015)

THE END

Not Covered

- **Topology**
- **Routing**
  - Geometry => very slow
  - PostGIS Extension Topology => slow
  - pgRouting
- **Raster Image Data**



DISCUSSION!

Stefan Keller

Geometa Lab at HSR www.hsr.ch/geometalab

Twitter: @geometalab and @sfkeller

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz