Simple Features

in Geodatenbanken

Referent: Niko Kolaxidis
Seminar Geodatenbanken
Universität Heidelberg
19.05.2022

Einstieg in die Thematik

Was sind Simple Features?

1 Was sind Simple Features?

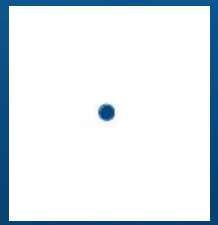
→ OpenGIS SFA

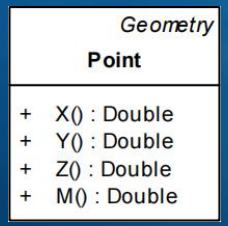
Feature =

Abstraktion von Objekten der realen Welt in digitale (Vektor-)daten

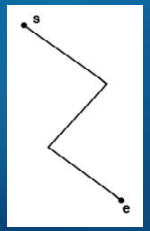
näher beschrieben in ISO 19107 des OpenGIS® Implementation Standard for Geographic information (OGC 2010 & 2011)

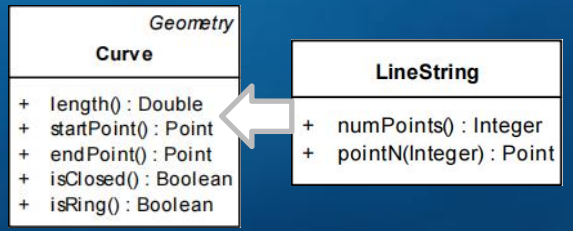
A 0 Dimensionen



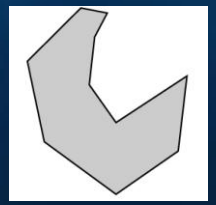


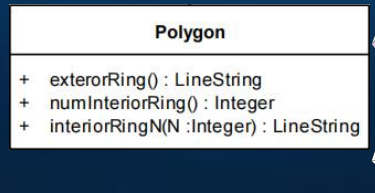
B 1 Dimension

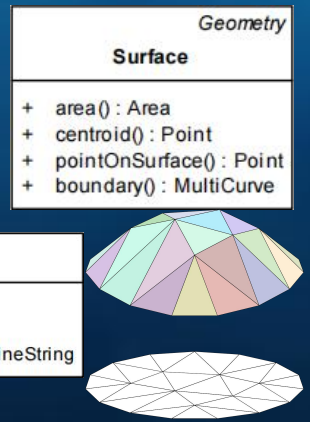


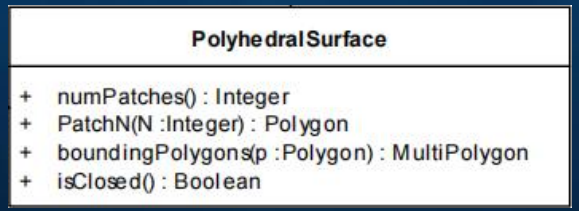


C 2 (bis 3) Dimensionen



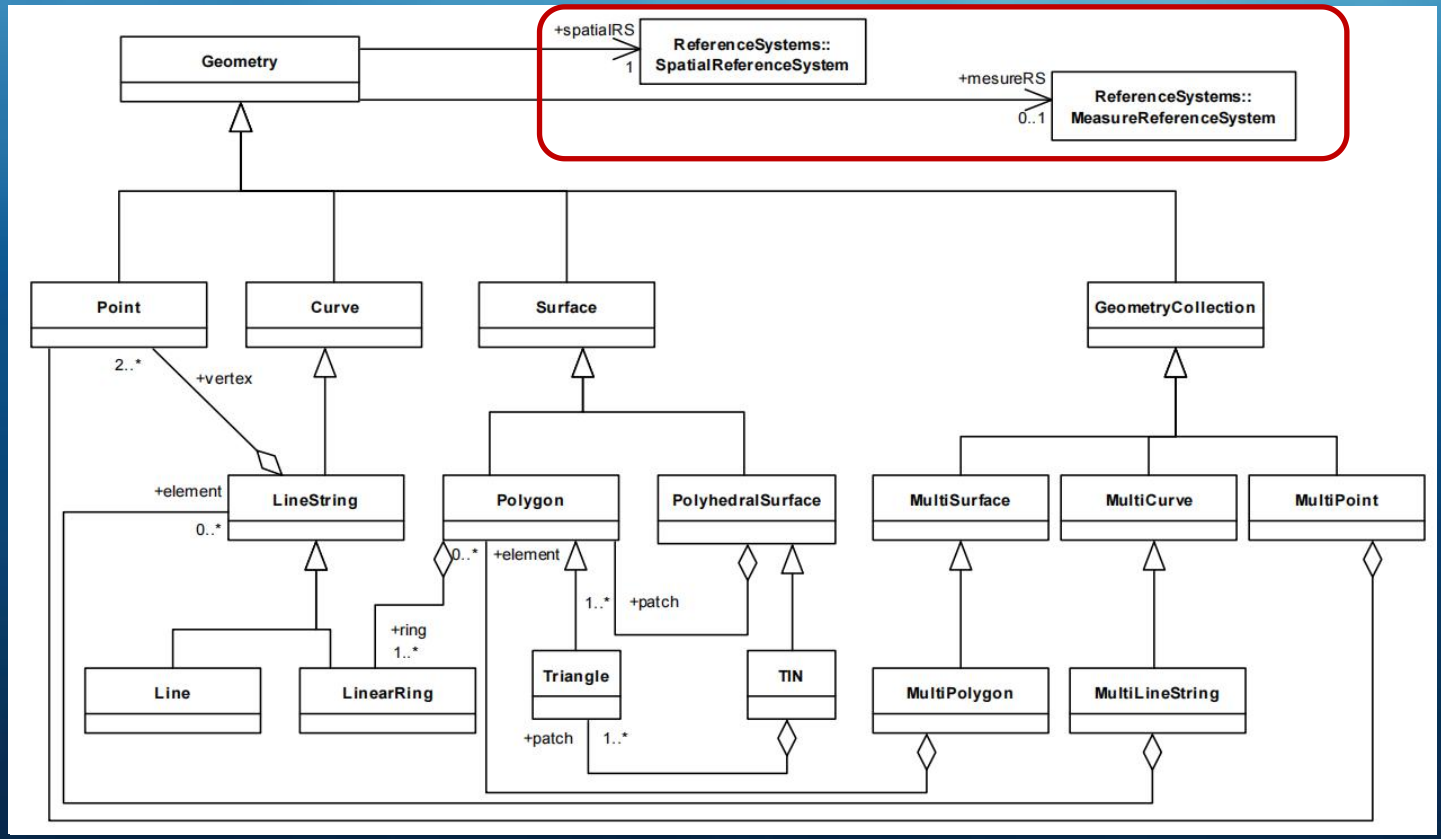






Quelle der Abbildungen: Alexandrov 2007; OGC 2011, S. 21-28; Stolze 2003, S. 251

1 Geometry object model für Simple-Features



Quelle: OGC 2011, S. 14

Integrierung in SQL

WKT vs. WKB

Erweiterungen

2 Repräsentation der Geometrie

Well-known Text (WKT)

```
Point(1,1)
```

Well-known Binary (WKB)

```
01010000000000000000F03F000000000000F03F
```

```
01 : Byte-Reihenfolge (Leserichtung)
01000000 : Geometrietyp (Punkt)
000000000000F03F : X
000000000000F03F : Y
```

```
LineString(2 2, 9 9)
```

```
0102000000020000000000000000000040000000000000004000000000000022400000000000002240
```

WKT:

```
INSERT INTO geotable ( geom, name )
  VALUES ( ST_GeomFromText('POINT(-126.4 45.32)', 312), 'A Place');
```

WKB:

```
INSERT INTO geotable ( geom, name )
  VALUES ( ST_GeomFromWKB('\x01010000000000000000f03f000000000000f03f', 312), 'A Place');
```

2 Geometrietypen in WKT

Geometrische Objekte in PostgreSQL

Name	Description	Representation
point	Point on a plane	(x,y)
line	Infinite line	{A,B,C}
lseg	Finite line segment	((x1,y1),(x2,y2))
box	Rectangular box	((x1,y1),(x2,y2))
path	Closed path (similar to polygon)	((x1,y1),...)
path	Open path	[(x1,y1),...]
polygon	Polygon (similar to closed path)	((x1,y1),...)
circle	Circle	<(x,y),r> (center point and radius)

Geometrische Objekte in PostGIS

- POINT (1 2)
- POINT Z (1 2 3)
- POINT ZM (1 2 3 4)
- POINT M (1 2 4)
- LINESTRING (1 2, 3 4, 5 6)
- LINEARRING (0 0 0, 4 0 0, 4 4 0, 0 4 0, 0 0 0)
- POLYGON ((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0))
- TRIANGLE ((0 0, 0 9, 9 0, 0 0))
- TIN Z (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0)))
- POLYHEDRALSURFACE Z (
  ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
  ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
  ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
  ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
  ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
  ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))
- MULTIPOINT ((0 0), (1 2))
- MULTILINESTRING ((0 0,1 1,1 2), (2 3,3 2,5 4))
- MULTIPOLYGON (((1 5, 5 5, 5 1, 1 1, 1 5)), ((6 5, 9 1, 6 1, 6 5)))
- GEOMETRYCOLLECTION (POINT(2 3), LINESTRING(2 3, 3 4))
- CIRCULARSTRING(0 0, 1 1, 1 0)
- CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0)
- COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0 1))
- CURVEPOLYGON(
  CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),(1 1, 3 3, 3 1, 1 1))
- CURVEPOLYGON(
  COMPOUNDCURVE(
  CIRCULARSTRING(0 0,2 0, 2 1, 2 3, 4 3),(4 3, 4 5, 1 4, 0 0)),
  CIRCULARSTRING(1.7 1, 1.4 0.4, 1.6 0.4, 1.6 0.5, 1.7 1))
- MULTICURVE((0 0, 5 5), CIRCULARSTRING(4 0, 4 4, 8 4))
- MULTISURFACE(
  CURVEPOLYGON(
  CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),(1 1, 3 3, 3 1, 1 1)),
  ((10 10, 14 12, 11 10, 10 10), (11 11, 11.5 11, 11 11.5, 11 11)))

Quelle: PostgreSQL Group 2022, S. 166

2 Erweiterungen

- MySQL Spatial Extensions
- MonetDB/GIS extension für MonetDB
- PostGIS extension für PostgreSQL
- SpatiaLite extension für SQLite
- Oracle Spatial
- IBM DB2 Spatial Extender ind IBM Informix Spatial DataBlade
- Microsoft SQL Server
- SAP Sybase IQ
- SAP HANA
- OGR für GDAL
- etc.

Geodatenbanken

- **Geometrische Datentypen**
  - z.B. Punkte, Linien, Polygone
- **Geometrische Operationen / Funktionen**
  - Schnittfläche / Intersection
  - Flächen- und Längenberechnung
- **Räumliche Datenstrukturen**
  - "räumlicher Index" (z.B. R-Baum)
- **Räumliche Abfragen**
  - Punktabfrage
  - Distanzabfrage
  - Regionsabfrage
  - Nächste-Nachbarn-Abfrage
  - ...

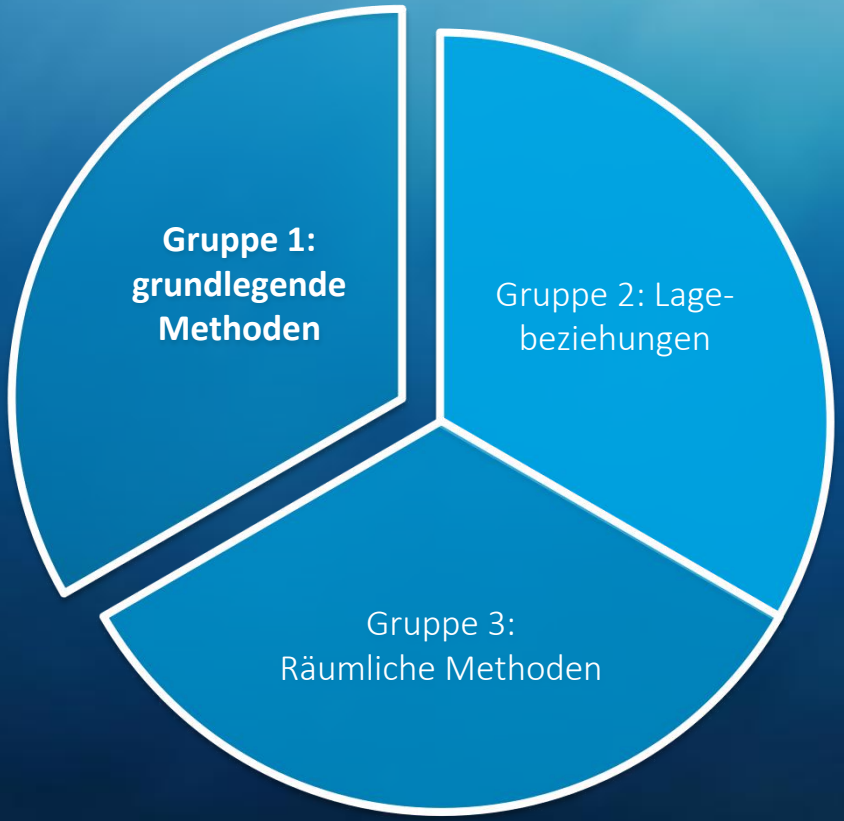
Quelle: Auer 2022, S. 34

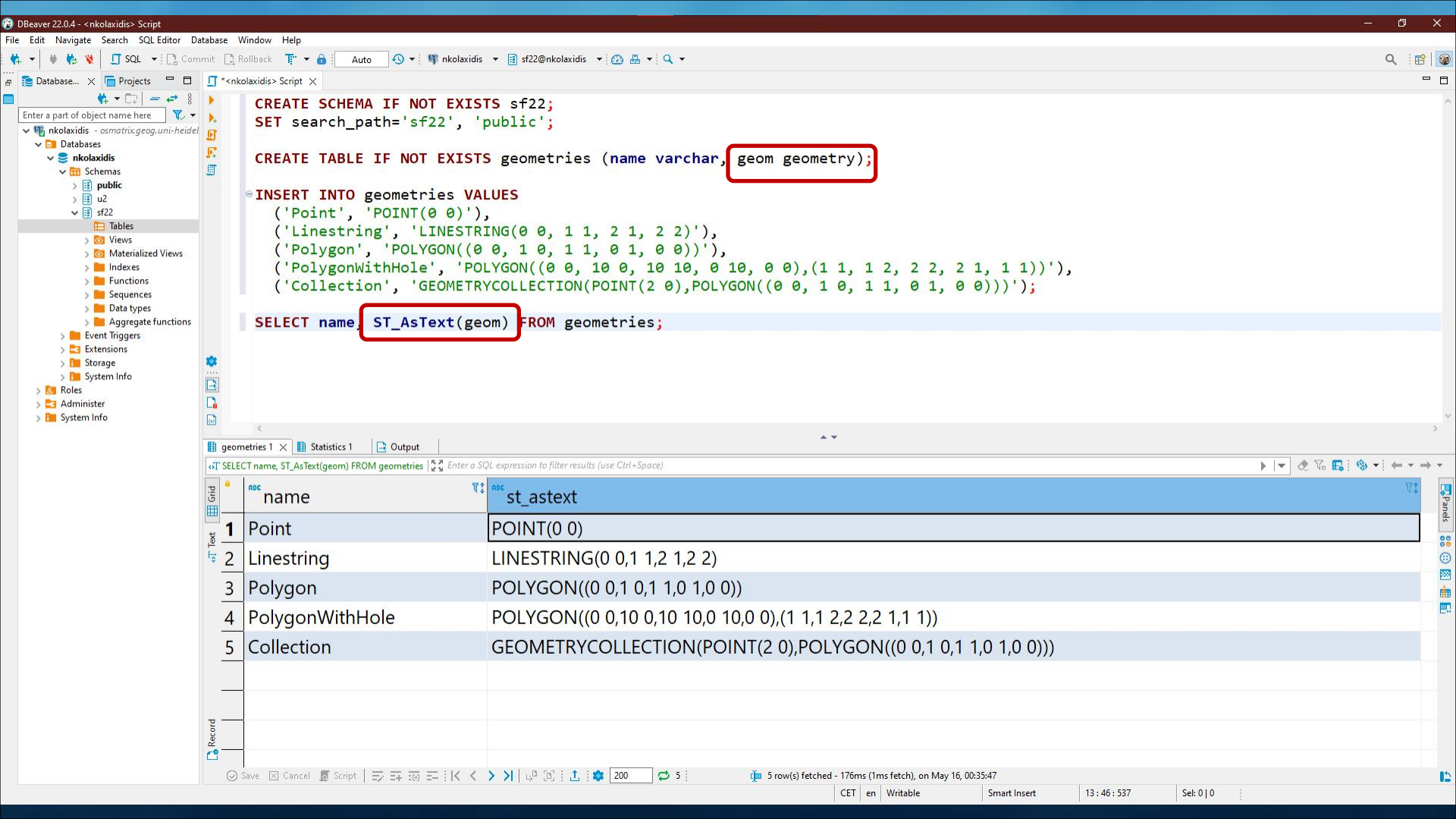
Simple Features in SQL: PostGIS

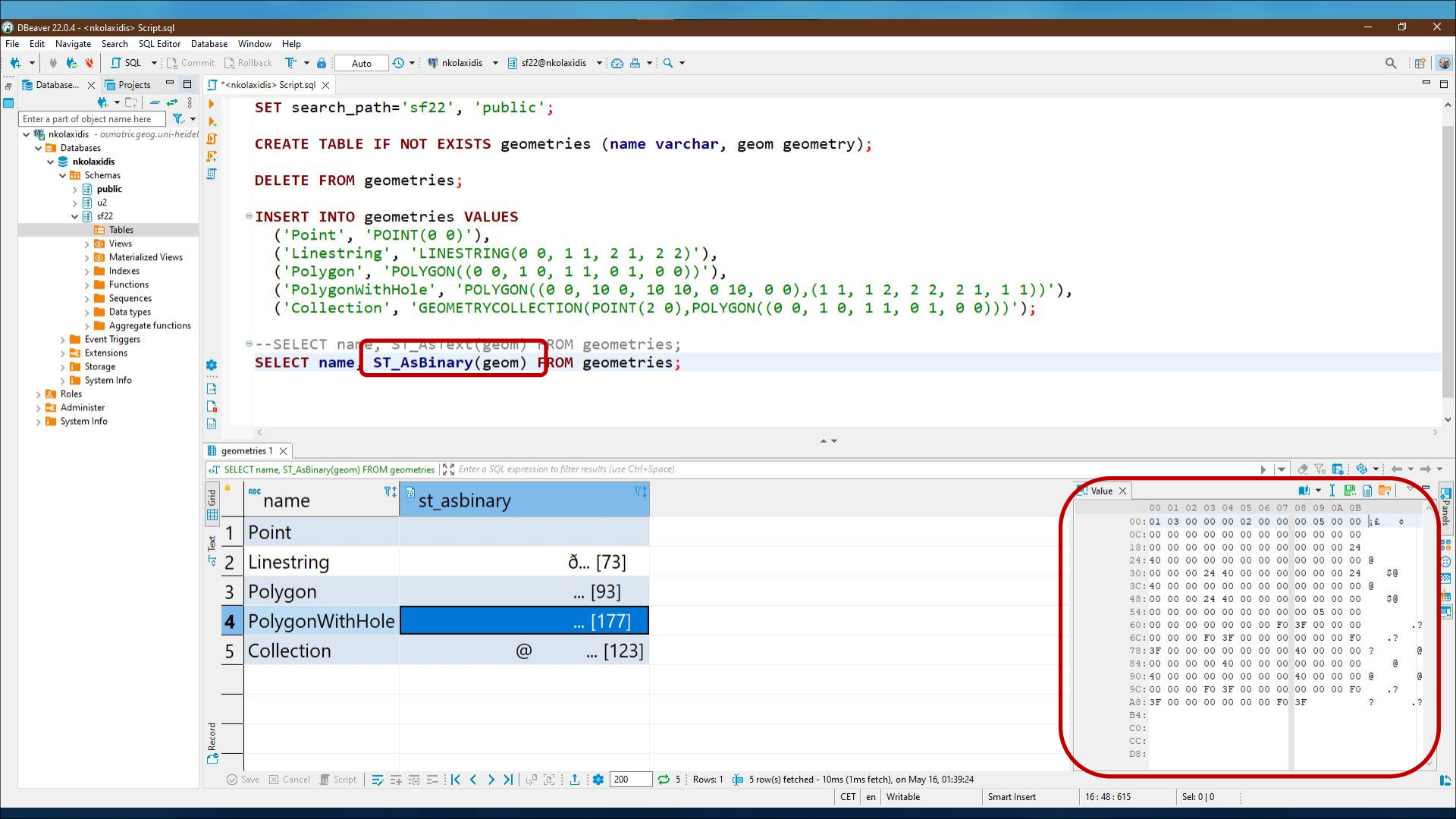
Was ist alles mit Simple Features möglich?

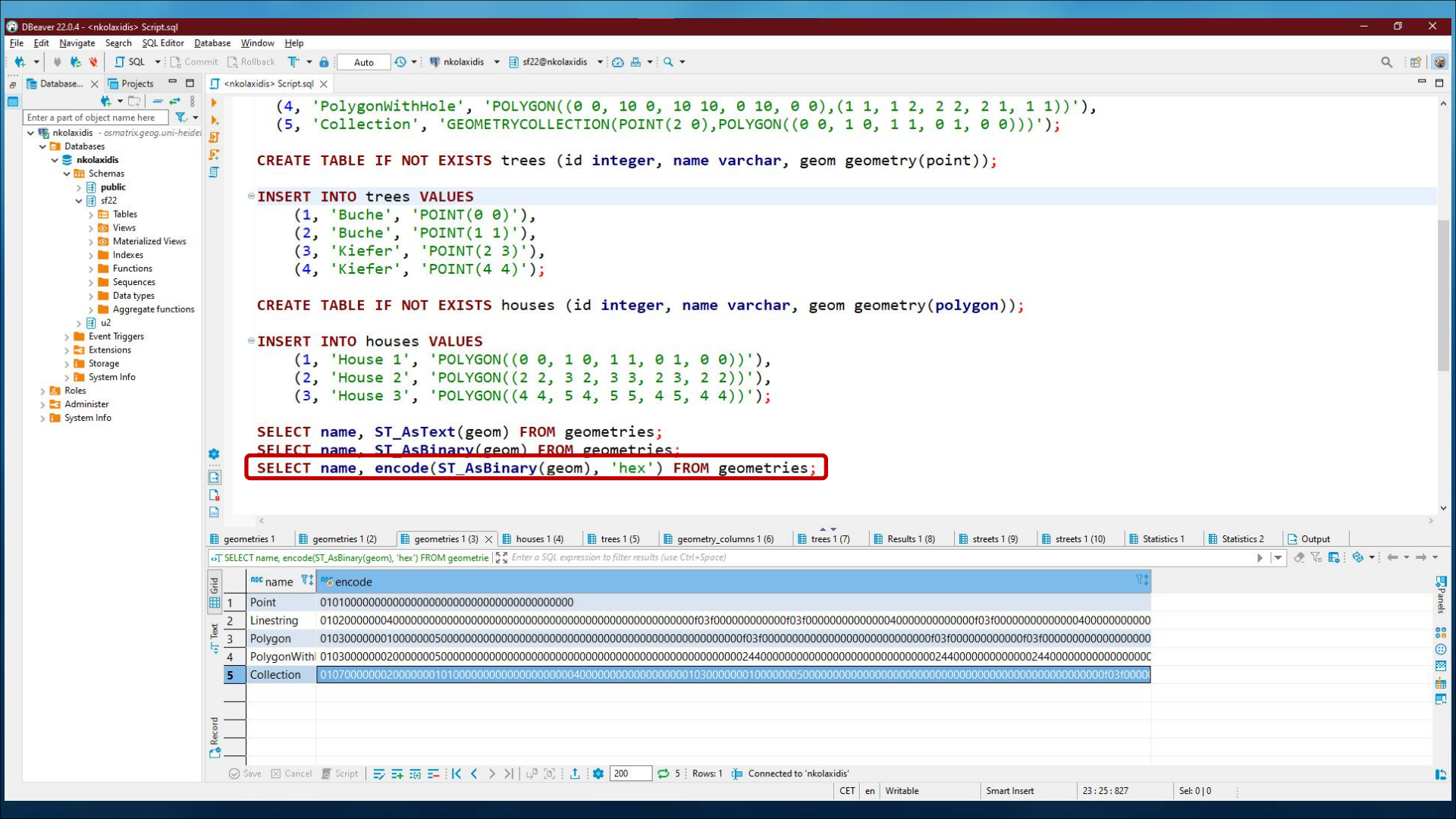
Beispielanwendungen

3 Was mit Simple Features alles möglich ist - Teil 1




DBeaver 22.0.4 - <nkolaxidis> Script
File Edit Navigate Search SQL Editor Database Window Help
Database... Projects
nkolaxidis - osmatrix.geog.uni-heidel
Databases
nkolaxidis
Schemas
public
u2
sf22
Tables
Views
Materialized Views
Indexes
Functions
Sequences
Data types
Aggregate functions
Event Triggers
Extensions
Storage
System Info
Roles
Administer
System Info
*<nkolaxidis> Script
CREATE SCHEMA IF NOT EXISTS sf22;
SET search_path='sf22', 'public';
CREATE TABLE IF NOT EXISTS geometries (name varchar, geom geometry);
INSERT INTO geometries VALUES
('Point', 'POINT(0 0)'),
('Linestring', 'LINESTRING(0 0, 1 1, 2 1, 2 2)'),
('Polygon', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
('PolygonWithHole', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1))'),
('Collection', 'GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))');
SELECT name, ST_AsText(geom) FROM geometries;
geometries 1
Statistics 1
Output
SELECT name, ST_AsText(geom) FROM geometries
Enter a SQL expression to filter results (use Ctrl+Space)
name
st_astext
1 Point POINT(0 0)
2 Linestring LINESTRING(0 0,1 1,2 1,2 2)
3 Polygon POLYGON((0 0,1 0,1 1,0 1,0 0))
4 PolygonWithHole POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 2,2 1,1 1))
5 Collection GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0,1 0,1 1,0 1,0 0)))
5 row(s) fetched - 176ms (1ms fetch), on May 16, 00:35:47
CET en Writable Smart Insert 13 : 46 : 537 Sel: 0 | 0


DBeaver 22.0.4 - <nkolaxidis> Script.sql
File Edit Navigate Search SQL Editor Database Window Help
SET search_path='sf22', 'public';
CREATE TABLE IF NOT EXISTS geometries (name varchar, geom geometry);
DELETE FROM geometries;
INSERT INTO geometries VALUES
('Point', 'POINT(0 0)'),
('Linestring', 'LINESTRING(0 0, 1 1, 2 1, 2 2)'),
('Polygon', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
('PolygonWithHole', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1))'),
('Collection', 'GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))');
--SELECT name, ST_AsText(geom) FROM geometries;
SELECT name, ST_AsBinary(geom) FROM geometries;
geometries 1
SELECT name, ST_AsBinary(geom) FROM geometries
name
st_asbinary
1 Point
2 Linestring ð... [73]
3 Polygon ... [93]
4 PolygonWithHole ... [177]
5 Collection @ ... [123]
Value
00 01 02 03 04 05 06 07 08 09 0A 0B
00: 01 03 00 00 00 02 00 00 00 05 00 00
0C: 00 00 00 00 00 00 00 00 00 00 00 00
18: 00 00 00 00 00 00 00 00 00 00 00 24
24: 40 00 00 00 00 00 00 00 00 00 00 00
30: 00 00 00 24 40 00 00 00 00 00 00 24
3C: 40 00 00 00 00 00 00 00 00 00 00 00
48: 00 00 00 24 40 00 00 00 00 00 00 00
54: 00 00 00 00 00 00 00 00 00 05 00 00
60: 00 00 00 00 00 00 00 F0 3F 00 00 00
6C: 00 00 00 F0 3F 00 00 00 00 00 00 F0
78: 3F 00 00 00 00 00 00 00 40 00 00 00
84: 00 00 00 00 40 00 00 00 00 00 00 00
90: 40 00 00 00 00 00 00 00 40 00 00 00
9C: 00 00 00 F0 3F 00 00 00 00 00 00 F0
A8: 3F 00 00 00 00 00 00 F0 3F
B4:
C0:
CC:
D8:
Rows: 1
5 row(s) fetched - 10ms (1ms fetch), on May 16, 01:39:24


DBeaver 22.0.4 - <nkolaxidis> Script.sql
File Edit Navigate Search SQL Editor Database Window Help
(4, 'PolygonWithHole', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1))'),
(5, 'Collection', 'GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))');
CREATE TABLE IF NOT EXISTS trees (id integer, name varchar, geom geometry(point));
INSERT INTO trees VALUES
(1, 'Buche', 'POINT(0 0)'),
(2, 'Buche', 'POINT(1 1)'),
(3, 'Kiefer', 'POINT(2 3)'),
(4, 'Kiefer', 'POINT(4 4)');
CREATE TABLE IF NOT EXISTS houses (id integer, name varchar, geom geometry(polygon));
INSERT INTO houses VALUES
(1, 'House 1', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
(2, 'House 2', 'POLYGON((2 2, 3 2, 3 3, 2 3, 2 2))'),
(3, 'House 3', 'POLYGON((4 4, 5 4, 5 5, 4 5, 4 4))');
SELECT name, ST_AsText(geom) FROM geometries;
SELECT name, ST_AsBinary(geom) FROM geometries;
SELECT name, encode(ST_AsBinary(geom), 'hex') FROM geometries;
name
encode
1 Point 01010000000000000000000000000000000000000000
2 Linestring 0102000000040000000000000000000000000000000000000000000000000000000000000000f03f00000000000000f03f0000000000000004000000000000000f03f0000000000000004000000000000
3 Polygon 0103000000010000000500000000000000000000000000000000000000000000000000000000000000f03f0000000000000000000000000000000f03f0000000000000f03f0000000000000000000000000000
4 PolygonWith 0103000000020000000500000000000000000000000000000000000000000000000000000000000000244000000000000000000000000000002440000000000000244000000000000000000
5 Collection 0107000000020000000101000000000000000000004000000000000000000103000000010000000500000000000000000000000000000000000000000000000000000000000000f03f0000

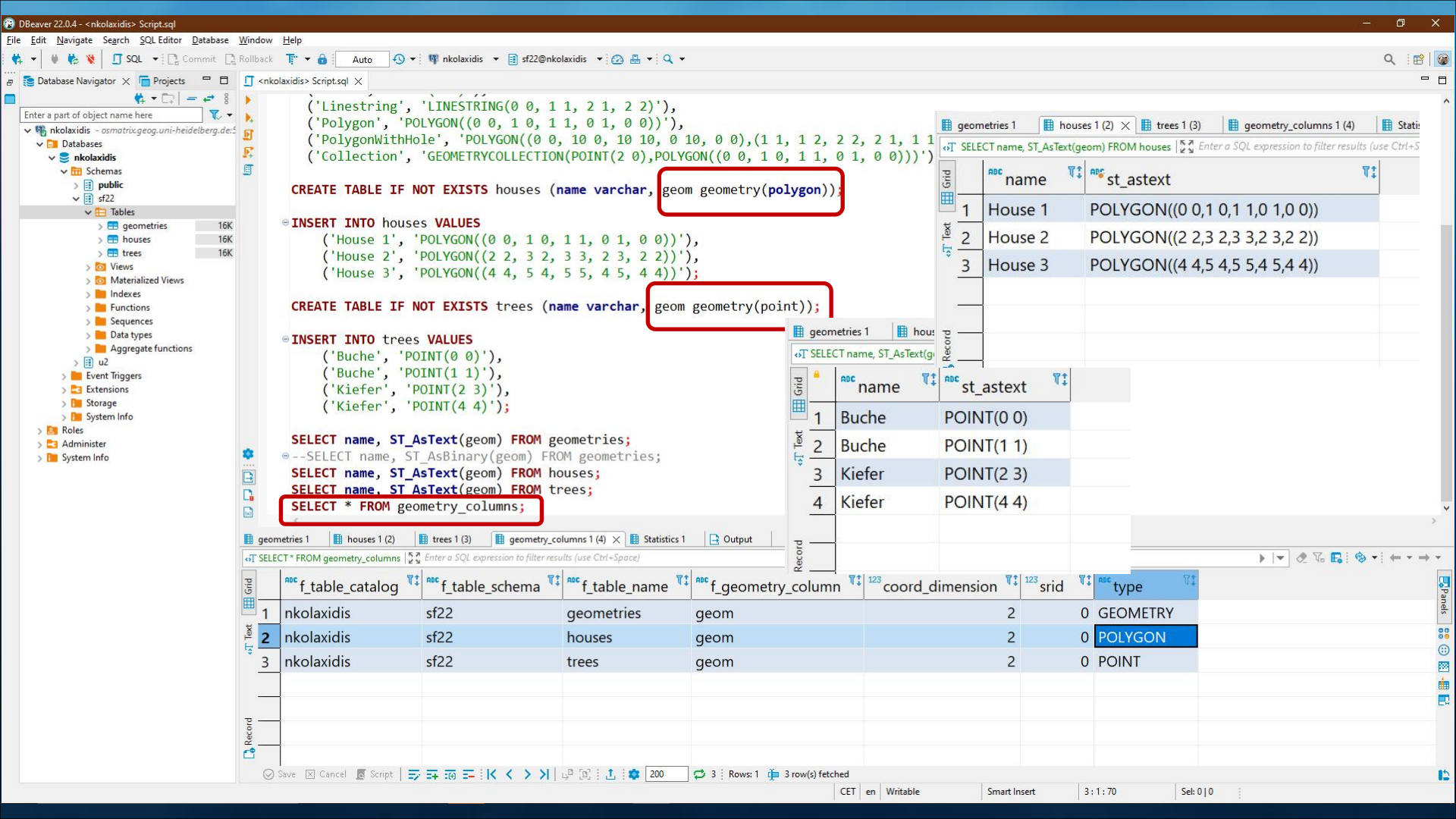
DBeaver 22.0.4 - <nkolaxidis> Script.sql

File Edit Navigate Search SQL Editor Database Window Help

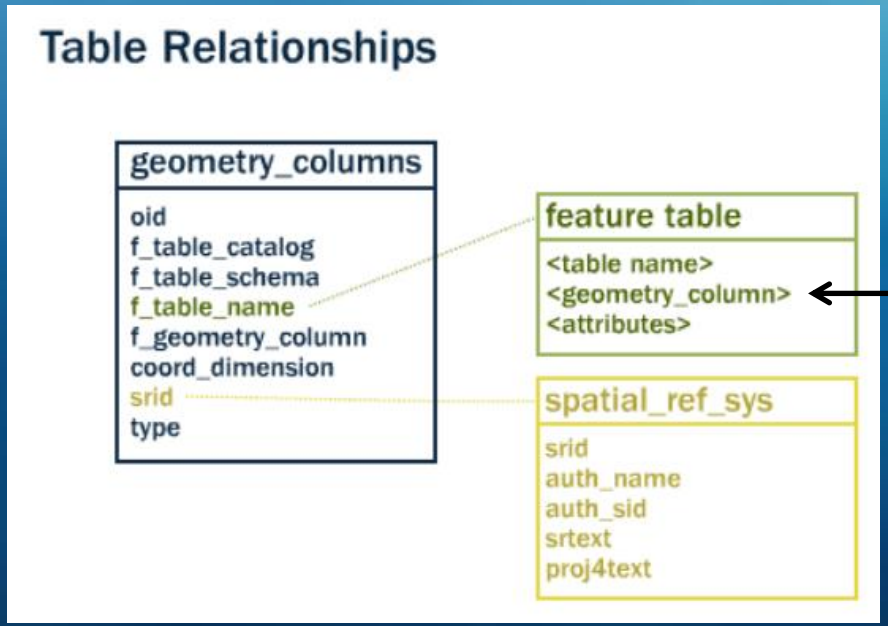
```
CREATE TABLE IF NOT EXISTS trees (id integer, name varchar, geom geometry(point));

INSERT INTO trees VALUES
    (1, 'Buche', 'POINT(0 0)'),
    (2, 'Buche', 'POINT(1 1)'),
    (3, 'Kiefer', 'POINT(2 3)'),
    (4, 'Kiefer', 'POINT(4 4)');

CREATE TABLE IF NOT EXISTS houses (id integer, name varchar, geom geometry(polygon));

INSERT INTO houses VALUES
    (1, 'House 1', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
    (2, 'House 2', 'POLYGON((2 2, 3 2, 3 3, 2 3, 2 2))'),
    (3, 'House 3', 'POLYGON((4 4, 5 4, 5 5, 4 5, 4 4))');

SELECT name, ST_AsText(geom) FROM geometries;
SELECT name, ST_AsBinary(geom) FROM geometries;
SELECT name, encode(ST_AsBinary(geom), 'hex') FROM geometries;
SELECT name, ST_AsText(geom) FROM houses;
SELECT name, ST_AsText(geom) FROM trees;

SELECT name, ST_ASGeoJSON(geom) FROM trees;
```

	name	st_asgeojson
1	Buche	{"type":"Point","coordinates":[0,0]}
2	Buche	{"type":"Point","coordinates":[1,1]}
3	Kiefer	{"type":"Point","coordinates":[2,3]}
4	Kiefer	{"type":"Point","coordinates":[4,4]}

4 row(s) fetched


DBeaver 22.0.4 - <nkolaxidis> Script.sql
File Edit Navigate Search SQL Editor Database Window Help
Database Navigator
Projects
<nkolaxidis> Script.sql
('Linestring', 'LINESTRING(0 0, 1 1, 2 1, 2 2)'),
('Polygon', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
('PolygonWithHole', 'POLYGON((0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1
('Collection', 'GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))')
CREATE TABLE IF NOT EXISTS houses (name varchar, geom geometry(polygon));
INSERT INTO houses VALUES
('House 1', 'POLYGON((0 0, 1 0, 1 1, 0 1, 0 0))'),
('House 2', 'POLYGON((2 2, 3 2, 3 3, 2 3, 2 2))'),
('House 3', 'POLYGON((4 4, 5 4, 5 5, 4 5, 4 4))');
CREATE TABLE IF NOT EXISTS trees (name varchar, geom geometry(point));
INSERT INTO trees VALUES
('Buche', 'POINT(0 0)'),
('Buche', 'POINT(1 1)'),
('Kiefer', 'POINT(2 3)'),
('Kiefer', 'POINT(4 4)');
SELECT name, ST_AsText(geom) FROM geometries;
--SELECT name, ST_AsBinary(geom) FROM geometries;
SELECT name, ST_AsText(geom) FROM houses;
SELECT name, ST_AsText(geom) FROM trees;
SELECT * FROM geometry_columns;
geometries 1
houses 1 (2)
trees 1 (3)
geometry_columns 1 (4)
SELECT name, ST_AsText(geom) FROM houses
name
st_astext
1 House 1 POLYGON((0 0,1 0,1 1,0 1,0 0))
2 House 2 POLYGON((2 2,3 2,3 3,2 3,2 2))
3 House 3 POLYGON((4 4,5 4,5 5,4 5,4 4))
1 Buche POINT(0 0)
2 Buche POINT(1 1)
3 Kiefer POINT(2 3)
4 Kiefer POINT(4 4)
Statistics 1
Output
SELECT * FROM geometry_columns
f_table_catalog
f_table_schema
f_table_name
f_geometry_column
coord_dimension
srid
type
1 nkolaxidis sf22 geometries geom 2 0 GEOMETRY
2 nkolaxidis sf22 houses geom 2 0 POLYGON
3 nkolaxidis sf22 trees geom 2 0 POINT
Rows: 1
3 row(s) fetched
CET en Writable Smart Insert 3 : 1 : 70 Sel: 0 | 0

3 Speicherarchitektur von Geo-Features



Geometry Table (Normalized Schema)		Geometry Table (Binary Schema)
GID ESEQ ETYPE SEQ X1 Y1 ... ... X<MAX PP R> Y<MAX PP R>	or	GID YMIN YMAX XMIN XMAX WKB_GEOMETRY

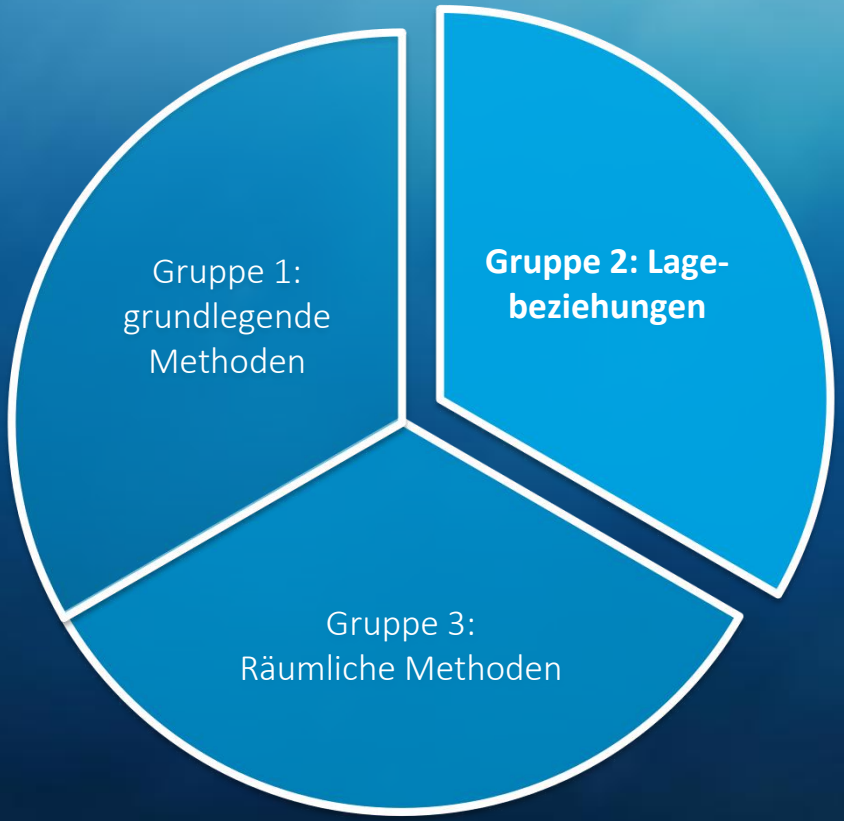
Quelle: PostGIS Group 2022; OGC 2010, S. 6

3 Grundlegende Methoden

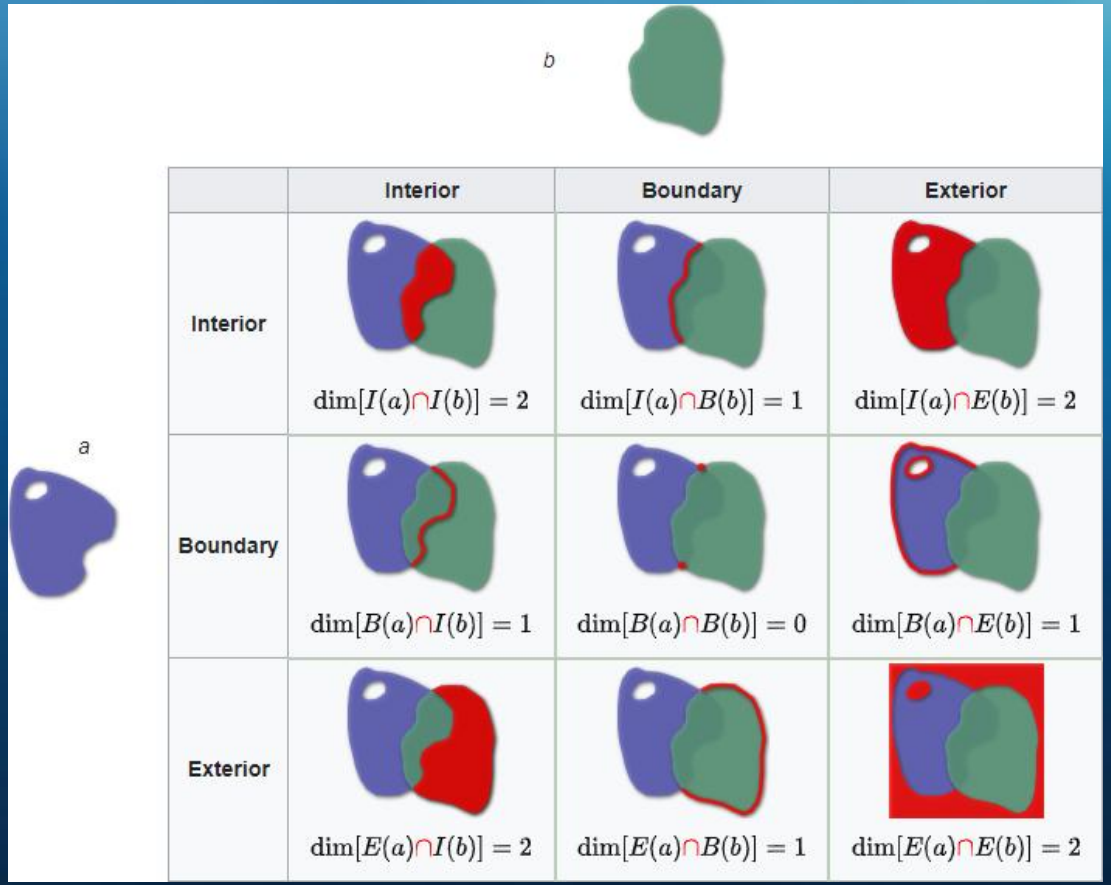
SELECT Methode(Geometriespalte und weitere Attribute der Methode) FROM Tabelle;

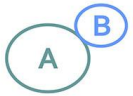
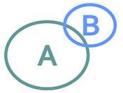
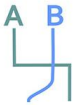
- ST_Area: Returns the area of the surface if it is a polygon or multi-polygon. For “geometry” type area is in SRID units. For “geography” area is in square meters.
- ST_AsText: Returns the Well-Known Text (WKT) representation of the geometry/geography without SRID metadata.
- ST_AsBinary: Returns the Well-Known Binary (WKB) representation of the geometry/geography without SRID meta data.
- ST_EndPoint: Returns the last point of a LINESTRING geometry as a POINT.
- ST_AsEWKB: Returns the Well-Known Binary (WKB) representation of the geometry with SRID meta data.
- ST_AsEWKT: Returns the Well-Known Text (WKT) representation of the geometry with SRID meta data.
- ST_AsGeoJSON: Returns the geometry as a GeoJSON element.
- ST_AsGML: Returns the geometry as a GML version 2 or 3 element.
- ST_AsKML: Returns the geometry as a KML element. Several variants. Default version=2, default precision=15.
- ST_AsSVG: Returns a Geometry in SVG path data given a geometry or geography object.
- ST_ExteriorRing: Returns a line string representing the exterior ring of the POLYGON geometry. Return NULL if the geometry is not a polygon. Will not work with MULTIPOLYGON
- ST_GeometryN: Returns the 1-based Nth geometry if the geometry is a GEOMETRYCOLLECTION, MULTIPOINT, MULTILINESTRING, MULTICURVE or MULTIPOLYGON. Otherwise, return NULL.
- ST_GeomFromGML: Takes as input GML representation of geometry and outputs a PostGIS geometry object.
- ST_GeomFromKML: Takes as input KML representation of geometry and outputs a PostGIS geometry object
- ST_GeomFromText: Returns a specified ST_Geometry value from Well-Known Text representation (WKT).
- ST_GeomFromWKB: Creates a geometry instance from a Well-Known Binary geometry representation (WKB) and optional SRID.
- ST_GeometryType: Returns the geometry type of the ST_Geometry value.
- ST_InteriorRingN: Returns the Nth interior linestring ring of the polygon geometry. Return NULL if the geometry is not a polygon or the given N is out of range.
- ST_Length: Returns the 2d length of the geometry if it is a linestring or multilinestring. geometry are in units of spatial reference and geography are in meters (default spheroid)
- ST_NDims: Returns coordinate dimension of the geometry as a small int. Values are: 2,3 or 4.
- ST_NPoints: Returns the number of points (vertexes) in a geometry.
- ST_NRings: If the geometry is a polygon or multi-polygon returns the number of rings.
- ST_NumGeometries: If geometry is a GEOMETRYCOLLECTION (or MULTI*) returns the number of geometries, otherwise return NULL.
- ST_Perimeter: Returns the length measurement of the boundary of an ST_Surface or ST_MultiSurface value. (Polygon, Multipolygon)
- ST_SRID: Returns the spatial reference identifier for the ST_Geometry as defined in spatial_ref_sys table.
- ST_StartPoint: Returns the first point of a LINESTRING geometry as a POINT.
- ST_X: Returns the X coordinate of the point, or NULL if not available. Input must be a point.
- ST_Y: Returns the Y coordinate of the point, or NULL if not available. Input must be a point.

3 Was mit Simple Features alles möglich ist - Teil 2



3 Lagebeziehungen: Dimensionally Extended 9-Intersection Model (DE-9IM)

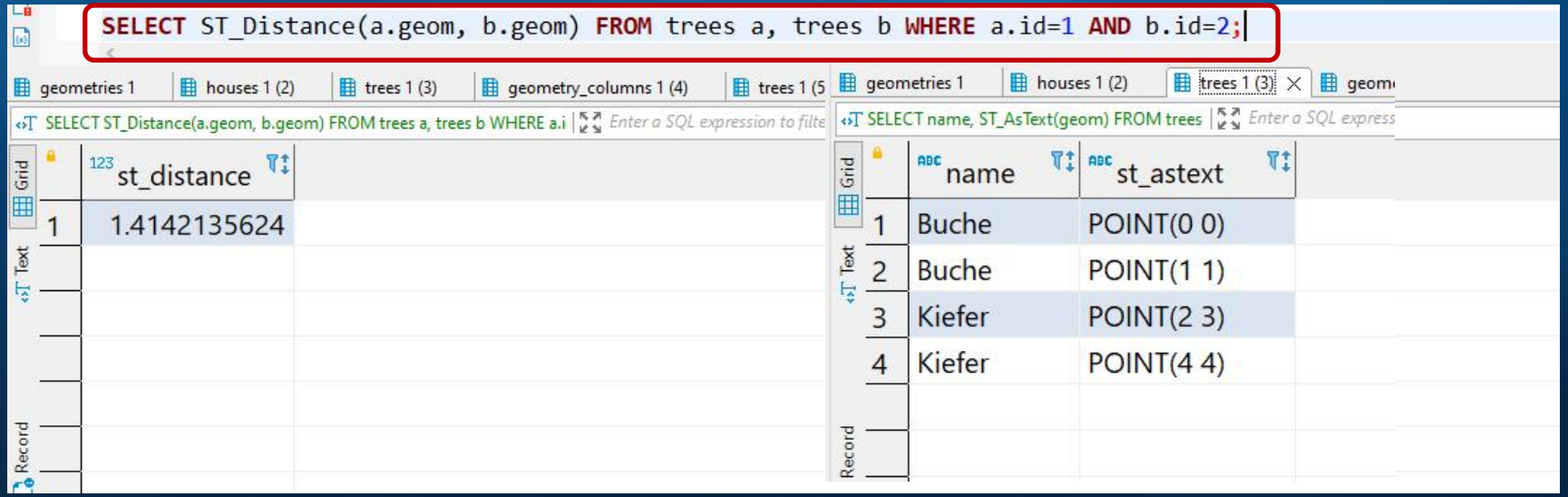


Equals A is the same as B	
Touches A touches B	
Overlaps A and B have multiple points in common	
Contains A contains B	
Disjoint A shares nothing with B	
Covers A covers B (or vice versa)	
Crosses A and B have at least one point in common	

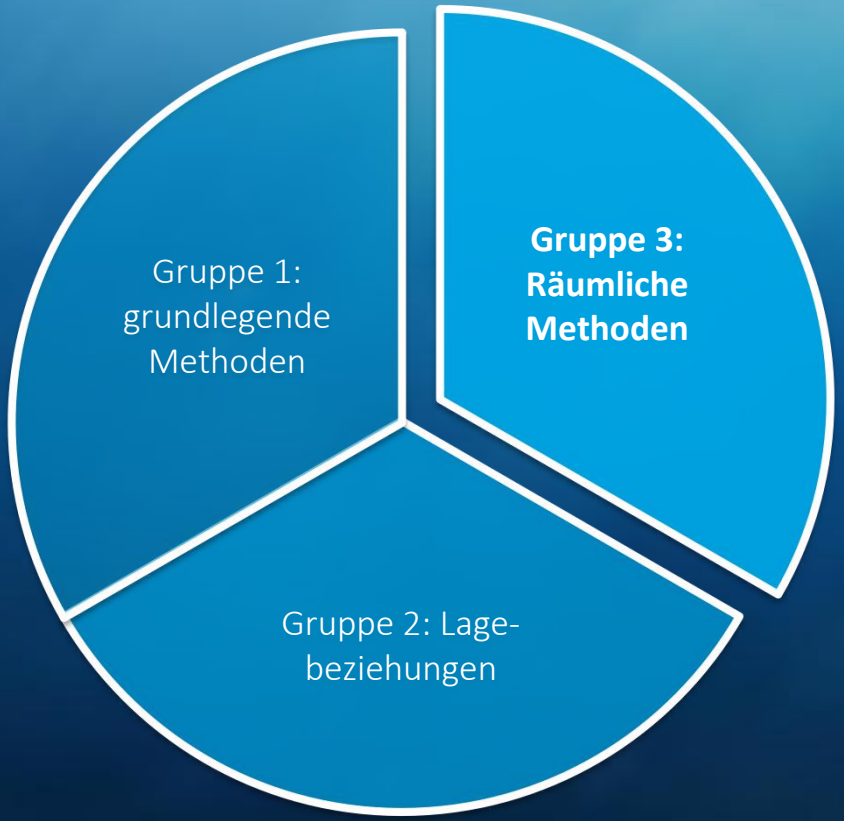
Quelle: PostGIS Group 2022; Pennsylvania State University 2020

3 Lagebeziehungen

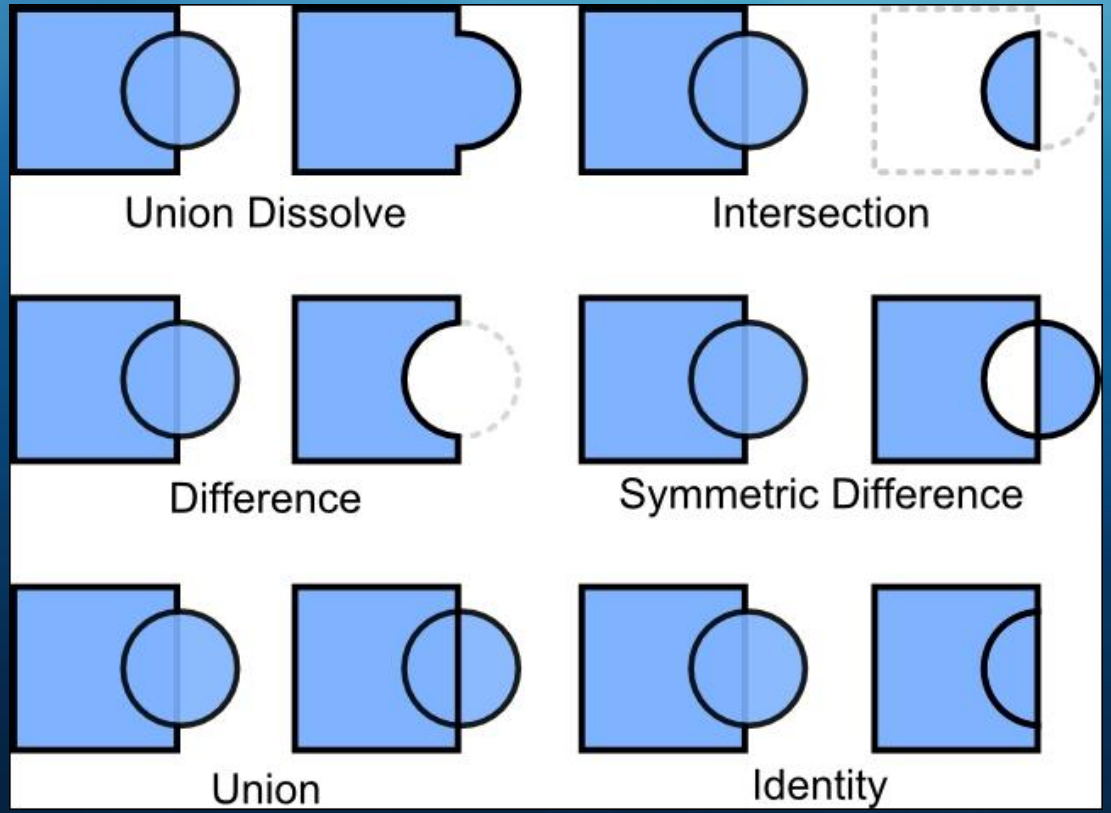
- ST_Contains(geometry A, geometry B): Returns true if and only if no points of B lie in the exterior of A, and at least one point of the interior of B lies in the interior of A.
- ST_Crosses(geometry A, geometry B): Returns TRUE if the supplied geometries have some, but not all, interior points in common.
- ST_Disjoint(geometry A , geometry B): Returns TRUE if the Geometries do not “spatially intersect” - if they do not share any space together.
- ST_Distance(geometry A, geometry B): Returns the 2-dimensional cartesian minimum distance (based on spatial ref) between two geometries in projected units.
- ST_DWithin(geometry A, geometry B, radius): Returns true if the geometries are within the specified distance (radius) of one another.
- ST_Equals(geometry A, geometry B): Returns true if the given geometries represent the same geometry. Directionality is ignored.
- ST_Intersects(geometry A, geometry B): Returns TRUE if the Geometries/Geography “spatially intersect” - (share any portion of space) and FALSE if they don’t (they are Disjoint).
- ST_Overlaps(geometry A, geometry B): Returns TRUE if the Geometries share space, are of the same dimension, but are not completely contained by each other.
- ST_Touches(geometry A, geometry B): Returns TRUE if the geometries have at least one point in common, but their interiors do not intersect.
- ST_Within(geometry A , geometry B): Returns true if the geometry A is completely inside geometry B



3 Was mit Simple Features alles möglich ist - Teil 3



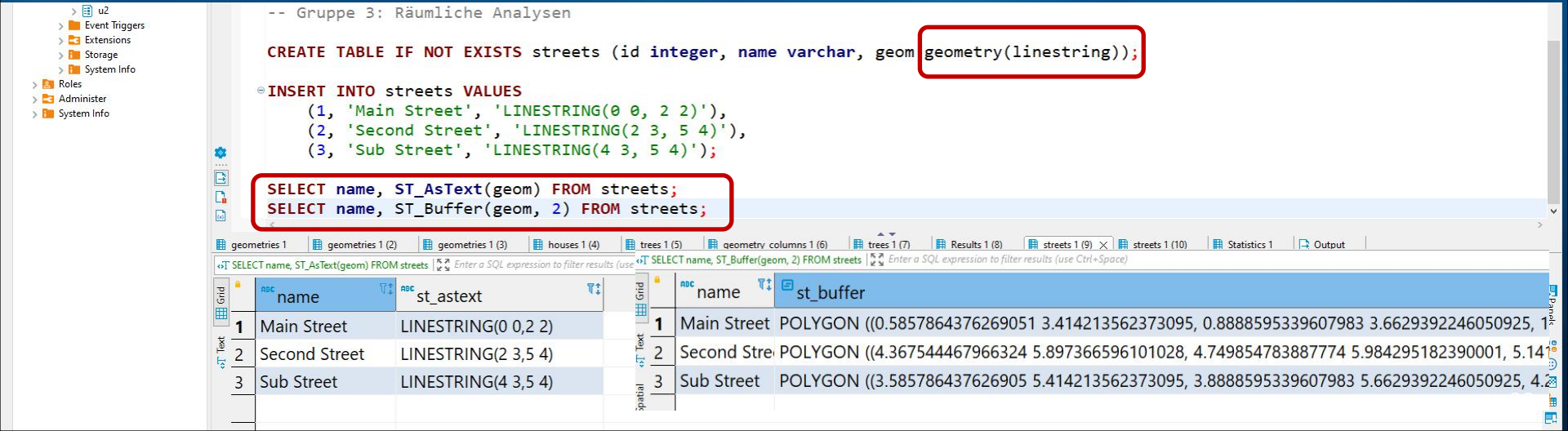
3 Räumliche Methoden



Quelle: Lautenbach/Reinmuth 2022

3 Räumliche Methoden

- ST_Centroid(geometry): Returns a point geometry that represents the center of mass of the input geometry.
- ST_PointOnSurface(geometry): Returns a point geometry that is guaranteed to be in the interior of the input geometry.
- ST_Buffer(geometry, distance): For geometry: Returns a geometry that represents all points whose distance from this Geometry is less than or equal to distance. Calculations are in the Spatial Reference System of this Geometry. For geography: Uses a planar transform wrapper.
- ST_Intersection(geometry A, geometry B): Returns a geometry that represents the shared portion of geomA and geomB. The geography implementation does a transform to geometry to do the intersection and then transform back to WGS84.
- ST_Union(): Returns a geometry that represents the point set union of the Geometries.



Simple Features

Vielen Dank für die Aufmerksamkeit!

Diskussion/Take-Home Messages

Diskussion/Take-Home Messages

1. Was sind die drei grundlegenen Feature-Klassen laut OpenGIS SFA und was zeichnet Simple Features aus?

2. Welche zwei Geometrie-Formate werden im OpenGIS SFA definiert und fällt dir noch ein anderes Geoformat ein?

3. Welche drei Gruppen von Methoden gibt es für Simple Features? Nenne je zwei für jede Gruppe.

4. Wie werden räumliche Methoden & Funktionen aufgerufen (Grundstruktur)?

Literaturnachweise

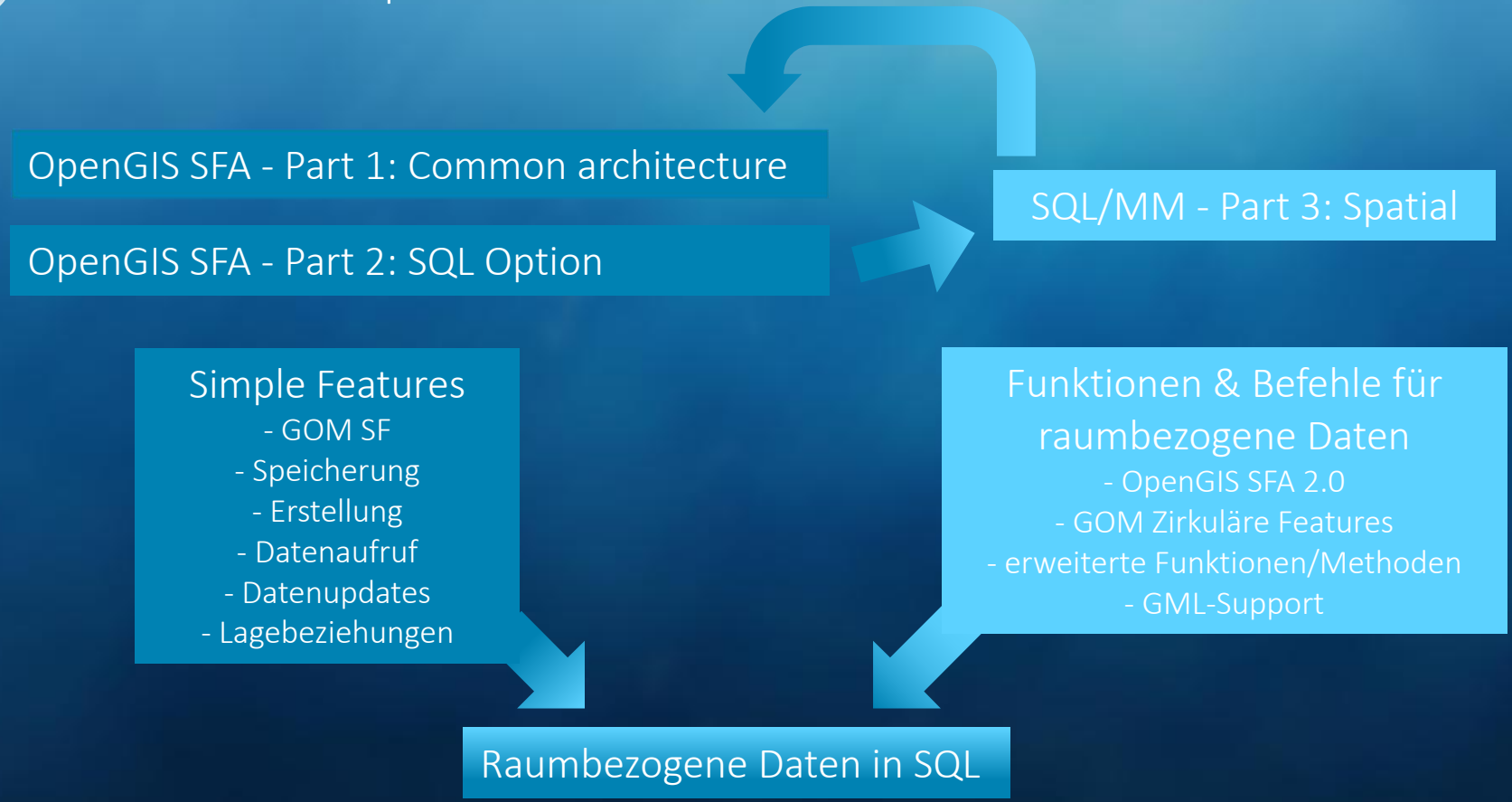
- Alexandrov, O. (2007): A piecewise linear function over two dimensions (top) and the polygonal areas on which it is linear (bottom). - URL: https://en.wikipedia.org/wiki/Polyhedral_terrain#/media/File:Piecewise_linear_function2D.svg [12.05.2022].
- Auer, M. (2022): Seminar Geodatenbanken. Allgemeine Einführung. Foliensatz, Universtät Heidelberg.
- OGC (2010): OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option.
- OGC (2011): OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture.
- OGC (2018): Features and geometry – Part 1: Feature models.
- Piorkowski, A. (2011): MySQL Spatial and PostGIS. Implementations of spatial data standards. In: Electronic Journal of Polish Agricultural Universities, 14(1), #3.
- The PostGIS Development Group (2022): Chapter 4. Data Mangement. PostGIS 3.2.2dev Manual. - URL: https://postgis.net/docs/using_postgis_dbmanagement.html [12.05.2022].
- The PostgreSQL Global Development Group (2022): PostgreSQL 14.3 Documentation.
- QGIS (2020): Simple-Feature-Modell. - URL: https://docs.qgis.org/3.22/de/docs/training_manual/spatial_databases/simple_feature_model.html [12.05.2022].
- Ramsey, P./Leslie, M. (2012): Introduction to PostGIS. - URL: https://postgis.net/workshops/postgis-intro/index.html [12.05.2022].
- Shekhar, S./Xiong, H. (2008): Encyclopedia of GIS. SpringerScience.
- Stolze, K. (o.J.): SQL/MM Spatial. The Standard to Manage Spatial Data in Relational Database Systems. Universität Jena.

Simple Features

Vielen Dank für die Aufmerksamkeit!

Referent: Niko Kolaxidis

1 Standards für Simple Features in SQL



2 Geometrietypen in WKB

Type	Code
Geometry	0
Point	1
LineString	2
Polygon	3
MultiPoint	4
MultiLineString	5
MultiPolygon	6
GeometryCollection	7
CircularString	8
CompoundCurve	9
CurvePolygon	10
MultiCurve	11
MultiSurface	12
Curve	13
Surface	14
PolyhedralSurface	15
TIN	16

Type	Code
Geometry Z	1000
Point Z	1001
LineString Z	1002
Polygon Z	1003
MultiPoint Z	1004
MultiLineString Z	1005
MultiPolygon Z	1006
GeometryCollection Z	1007
CircularString Z	1008
CompoundCurve Z	1009
CurvePolygon Z	1010
MultiCurve Z	1011
MultiSurface Z	1012
Curve Z	1013
Surface Z	1014
PolyhedralSurface Z	1015
TIN Z	1016

Type	Code
Geometry M	2000
Point M	2001
LineString M	2002
Polygon M	2003
MultiPoint M	2004
MultiLineString M	2005
MultiPolygon M	2006
GeometryCollection M	2007
CircularString M	2008
CompoundCurve M	2009
CurvePolygon M	2010
MultiCurve M	2011
MultiSurface M	2012
Curve M	2013
Surface M	2014
PolyhedralSurface M	2015
TIN M	2016

Type	Code
Geometry ZM	3000
Point ZM	3001
LineString ZM	3002
Polygon ZM	3003
MultiPoint ZM	3004
MultiLineString ZM	3005
MultiPolygon ZM	3006
GeometryCollection ZM	3007
CircularString ZM	3008
CompoundCurve ZM	3009
CurvePolygon ZM	3010
MultiCurve ZM	3011
MultiSurface ZM	3012
Curve ZM	3013
Surface ZM	3014
PolyhedralSurface ZM	3015
TIN ZM	3016

Quelle: OGC 2011, S. 63f.

2 Repräsentation der Geometrie

→ Nicht darauf verlassen!

Extended Well-known Text (EWKT)

Für 3D Geometrien wird Z per Default erkannt:

```
OGC: POINT Z (1 2 3)
EWKT: POINT (1 2 3)
```

Für 4D (3D + M) Geometrien werden Z und M per Default erkannt:

```
OGC: POINT ZM (1 2 3 4)
EWKT: POINT (1 2 3 4)
```

Einbezug des CRS (SRID):

```
SRID=32632;POINT(0 0)
```

Extended Well-known Binary (EWKB)

```
01010000020E61000007EAA0A0DC4920DC05AF0A2AF201F4440
```

```
OGC: POINT M (1 2 3)
EWKT: POINTM (1 2 3)
```