Universität Münster

Open Geospatial Consortium

STAC

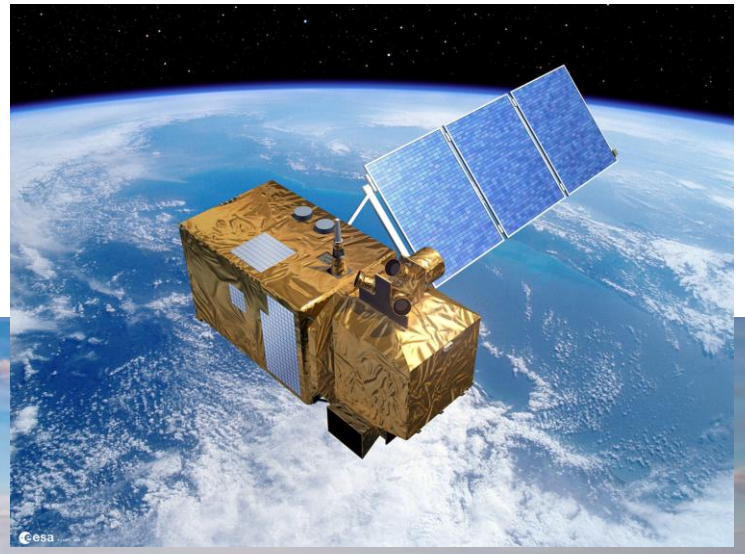
OGC Common Query Language 2

Luis Waldeyer, Nils Große Beikel

Geosoftware II, 15.10.2025

Universität Münster

Einstieg & Motivation



Einstieg & Motivation

- Warum brauchen wir CQL2?
  - Filtern ist eine Grundfunktion in Geodaten-APIs
  - Alte Standards (Filter Encoding, CQL1):
    - XML-basiert, schwer lesbar
    - Kompliziert, schlecht erweiterbar
  - Web-APIs brauchen eine einfache und einheitliche Filtersprache
  - CQL2: kompakt, verständlich und interoperabel – die moderne Weiterentwicklung
  - Ziel: Einheitliche Filterlogik für STAC & OGC API & andere Dienste

Was ist CQL2?

- Ein Filterstandard für die Geodatenwelt
- Offizieller OGC-Standard für Filterausdrücke (Version 1.0, 2024)
- Ermöglicht logische, räumliche & zeitliche Abfragen
- Zwei Schreibweisen: Text vs. JSON
- Standardisiert und kompatibel mit STAC & OGC API

Universität Münster

Syntaxvarianten (Text vs JSON)

```
cloud_cover < 10 AND platform = 'Sentinel-2A'
```

- Text-Encoding
  - Menschlich lesbare Syntax (ähnelt SQL)
  - Kompakt & gut für URL-Parameter geeignet
  - Unterstützt logische, räumliche & zeitliche Filter
  - Wird von OGC API – Features, Records & STAC unterstützt

Universität Münster

Syntaxvarianten (Text vs JSON)

```
{
  "op": "and",
  "args": [
    { "op": "<", "args": [ { "property": "cloud_cover" }, 10 ] },
    { "op": "=", "args": [ { "property": "platform" }, "Sentinel-2A" ] }
  ]
}
```

- JSON-Encoding
  - Maschinenlesbare Variante (JSON-Struktur)
  - Ideal für Web-APIs und zur Programmierung
  - Besser validierbar & einfacher zu parsen
  - Gleiche Logik wie Text-Encoding

Operatoren Überblick

- **Vergleichsoperatoren**
  `=`, `<>`, `<`, `>`, `BETWEEN`, `LIKE`, `IN`
  Vergleich von Attributen & Werten
- **Logische Operatoren**
  `AND`, `OR`, `NOT`
  Kombinieren von Bedingungen
- **Räumliche Operatoren**
  `S_INTERSECTS`, `S_WITHIN`, `S_CONTAINS`, `S_EQUALS`
  Beziehungen zwischen Geometrien
- **Zeitliche Operatoren**
  `T_BEFORE`, `T_AFTER`, `T_DURING`, `T_EQUALS`
  Vergleich zeitlicher Eigenschaften
- jeder Ausdruck entweder `true`, `false`, oder `null`

Beispiel-Filter (Text + JSON)

ID	platform	cloud_cover (%)	date	region	geom
1	Sentinel-2A	8	2024-03-15	Münster	POLYGON((7.5 51.9, 7.7 51.9, 7.7 52.0, 7.5 52.0, 7.5 51.9))
2	Sentinel-2B	23	2024-04-20	Münster	POLYGON((7.6 51.8, 7.8 51.8, 7.8 51.9, 7.6 51.9, 7.6 51.8))
3	Landsat-8	12	2023-08-10	Münster	POLYGON((7.5 52.0, 7.7 52.0, 7.7 52.2, 7.5 52.2, 7.5 52.0))
4	Sentinel-2A	5	2025-03-08	Münster	POLYGON((7.4 51.9, 7.6 51.9, 7.6 52.0, 7.4 52.0, 7.4 51.9))
5	Landsat-9	40	2025-04-11	Paderborn	POLYGON((8.0 51.7, 8.2 51.7, 8.2 51.9, 8.0 51.9, 8.0 51.7))

Beispiel-Filter (Text + JSON)

ID	platform	cloud_cover (%)	date	region	geom
1	Sentinel-2A	8	2024-03-15	Münster	POLYGON((7.5 51.9, 7.7 51.9, 7.7 52.0, 7.5 52.0, 7.5 51.9))
2	Sentinel-2B	23	2024-04-20	Münster	POLYGON((7.6 51.8, 7.8 51.8, 7.8 51.9, 7.6 51.9, 7.6 51.8))
3	Landsat-8	12	2023-08-10	Münster	POLYGON((7.5 52.0, 7.7 52.0, 7.7 52.2, 7.5 52.2, 7.5 52.0))
4	Sentinel-2A	5	2025-03-08	Münster	POLYGON((7.4 51.9, 7.6 51.9, 7.6 52.0, 7.4 52.0, 7.4 51.9))
5	Landsat-9	40	2025-04-11	Paderborn	POLYGON((8.0 51.7, 8.2 51.7, 8.2 51.9, 8.0 51.9, 8.0 51.7))

Abfrage: Landsat-8 Datensätze

Text	JSON
platform = 'Landsat-8'	`{ "op": "=", "args": [ {"property": "platform"}, "Landsat-8" ] }`

Beispiel-Filter (Text + JSON)

ID	platform	cloud_cover (%)	date	region	geom
1	Sentinel-2A	8	2024-03-15	Münster	POLYGON((7.5 51.9, 7.7 51.9, 7.7 52.0, 7.5 52.0, 7.5 51.9))
2	Sentinel-2B	23	2024-04-20	Münster	POLYGON((7.6 51.8, 7.8 51.8, 7.8 51.9, 7.6 51.9, 7.6 51.8))
3	Landsat-8	12	2023-08-10	Münster	POLYGON((7.5 52.0, 7.7 52.0, 7.7 52.2, 7.5 52.2, 7.5 52.0))
4	Sentinel-2A	5	2025-03-08	Münster	POLYGON((7.4 51.9, 7.6 51.9, 7.6 52.0, 7.4 52.0, 7.4 51.9))
5	Landsat-9	40	2025-04-11	Paderborn	POLYGON((8.0 51.7, 8.2 51.7, 8.2 51.9, 8.0 51.9, 8.0 51.7))

Abfrage: Weniger als 10% Wolken

Text	JSON

cloudcover < 10	{
	"op": "<",
	"args": [
	{"property": "cloudcover"},
	10
	]
	}

Beispiel-Filter (Text + JSON)

ID	platform	cloud_cover (%)	date	region	geom
1	Sentinel-2A	8	2024-03-15	Münster	POLYGON((7.5 51.9, 7.7 51.9, 7.7 52.0, 7.5 52.0, 7.5 51.9))
2	Sentinel-2B	23	2024-04-20	Münster	POLYGON((7.6 51.8, 7.8 51.8, 7.8 51.9, 7.6 51.9, 7.6 51.8))
3	Landsat-8	12	2023-08-10	Münster	POLYGON((7.5 52.0, 7.7 52.0, 7.7 52.2, 7.5 52.2, 7.5 52.0))
4	Sentinel-2A	5	2025-03-08	Münster	POLYGON((7.4 51.9, 7.6 51.9, 7.6 52.0, 7.4 52.0, 7.4 51.9))
5	Landsat-9	40	2025-04-11	Paderborn	POLYGON((8.0 51.7, 8.2 51.7, 8.2 51.9, 8.0 51.9, 8.0 51.7))

Abfrage: Bilder, die Münster überdecken

Text	JSON

S_INTERSECTS(geom, BBOX(7.5, 51.9, 7.7, 52.0))	{
	"op": "s_intersects",
	"args": [
	{"property": "geom"},
	{
	"type": "BBox",
	"value": [7.5, 51.9, 7.7, 52.0]
	}
	]
	}

Beispiel-Filter (Text + JSON)

ID	platform	cloud_cover (%)	date	region	geom
1	Sentinel-2A	8	2024-03-15	Münster	POLYGON((7.5 51.9, 7.7 51.9, 7.7 52.0, 7.5 52.0, 7.5 51.9))
2	Sentinel-2B	23	2024-04-20	Münster	POLYGON((7.6 51.8, 7.8 51.8, 7.8 51.9, 7.6 51.9, 7.6 51.8))
3	Landsat-8	12	2023-08-10	Münster	POLYGON((7.5 52.0, 7.7 52.0, 7.7 52.2, 7.5 52.2, 7.5 52.0))
4	Sentinel-2A	5	2025-03-08	Münster	POLYGON((7.4 51.9, 7.6 51.9, 7.6 52.0, 7.4 52.0, 7.4 51.9))
5	Landsat-9	40	2025-04-11	Paderborn	POLYGON((8.0 51.7, 8.2 51.7, 8.2 51.9, 8.0 51.9, 8.0 51.7))

Abfrage: Sentinel-Bilder mit wenig Wolken

Text

cloudcover < 10 AND
platform LIKE 'Sentinel%'

JSON

```
{
    "op": "and",
    "args": [
        {
            "op": "<",
            "args": [
                {"property": "cloudcover"}, 10
            ]
        },
        {
            "op": "like",
            "args": [
                {"property": "platform"}, "Sentinel%"
            ]
        }
    ]
}
```

Anwendung in **STAC** & OGC API

- STAC = SpatioTemporal Asset Catalog
  - API zum Durchsuchen und Filtern georäumlicher Sammlungen
- Unterstützt CQL2 zur präzisen Suche nach Items
- Formate für Filter:
  - Text-Encoding: einfach in GET-Requests
  - JSON-Encoding: komplexere Filter per POST
- Beispiel:
  - GET /collections/satellite_data/items?filter-lang=cql2-text&filter=cloudcover<10
  - POST /search

```
{
  "filter-lang": "cql2-json",
  "filter": {
    "op": "<",
    "args": [
      {"property": "cloudcover"},
      10
    ]
  }
}
```

Anwendung in STAC & **OGC API**

Open Geospatial Consortium

- OGC API = REST-basierter Standard des Open Geospatial Consortium (OGC)
- Nutzt dieselbe CQL2-Syntax für Filterparamter
- Filter werden bei POST-Requests im JSON-Body übergeben
- Ermöglicht strukturierte Abfragen auf Feature-Daten
- Gleiche Syntax → kompatibel zwischen OGC API & STAC API

Universität Münster

Fazit

- Einheitliche, moderne Filtersprache für alle OGC-konformen Dienste
- Erhöht Interoperabilität: einmal formulieren → überall nutzbar
- Verständlich für Menschen (Text) & Maschinen (JSON)
- Ablösen komplexer XML-Filter älterer Standards
- Grundlage für präzise, standardisierte Geodatenabfragen (z. B. in STAC, OGC API)
- Verweis auf offizielle Dokumentation vom OGC
  - https://docs.ogc.org/is/21-065r2/21-065r2.html

Universität Münster

Fragen?

Quellen

- https://www.ogc.org/standards/cql2
- https://www.ogc.org/announcement/ogc-membership-approves-common-query-language-cql2-as-an-official-ogc-standard
- https://docs.ogc.org/is/21-065r2/21-065r2.html
- Bilder:
  - https://www.esa.int/var/esa/storage/images/esa_multimedia/images/2008/11/sentinel-2/9773238-3-eng-GB/Sentinel-2_pillars.jpg
  - https://www.webbaviation.de/galerie/_data/i/galleries/Nordrhein-Westfalen/Muenster/Muenster_Innenstadt_Luftbild_qd08049-la.jpg
  - https://media.techem.com/is/image/techem/Interoperabilit%C3%A4t?qlt=85&wid=1000&ts=1753807604674&dpr=off
  - https://www.grin.com/wp-content/uploads/2021/12/Fazit_Blog.jpg