



DMAV-Check mit CNF

Demo (oder auch nicht)

► 01

02

03

04

05

06

07

08

09

10

11

12

13

14

WTF?!

- 01
- ▶ 02
- 03
- 04
- 05
- 06
- 07
- 08
- 09
- 10
- 11
- 12
- 13
- 14

Nachmittag / <i>Après-midi</i>		
13:45 - 14:15	Cloud-Native Geospatial: Was es kann, was nicht – und wann man es wirklich braucht <i>Cloud-native geospatial» : ce qu'elle peut faire, ce qu'elle ne peut pas faire... et quand on en a vraiment besoin?</i>	R. Straumann
14:15 - 14:30	Demo: DMAV-Check mit CNF <i>Démonstration : vérification DMAV avec CNF</i>	S. Ziegler
14:30 - 15:00	Erfahrungen Kanton BS <i>Expériences du canton BS</i>	L. Hubler



01

02

▶ 03

04

05

06

07

08

09

10

11

12

13

14

Constraints

```
01 !!@ name = GWRC02a
02 !!@ message="(GWRC02a) EGID existiert nicht im GWR";
03 !!@ message_de="(GWRC02a) EGID existiert nicht im GWR";
04 !!@ message_fr="(GWRC02a) L'EGID n'existe pas dans le RegBL";
05 !!@ message_it="(GWRC02a) L'EGID non esiste nel REA";
06 MANDATORY CONSTRAINT GWRC02a :
07     DEFINED(EGID) =>
08         IliValidGwr_V1_0.egidExistsInGWR(EGID,PARAMETER IliVRuntime_V1_0.Scope);
09
10 !!@ name = GWRC03
11 !!@ ilivalid.msg="Kombination EGID/EDID existiert nicht im GWR";
12 !!@ ilivalid.msg_de="Kombination EGID/EDID existiert nicht im GWR";
13 !!@ ilivalid.msg_fr="La combinaison EGID/EDID n'existe pas dans le RegBL";
14 !!@ ilivalid.msg_it="La combinazione EGID/EDID non esiste nel REA";
    MANDATORY CONSTRAINT GWRC03: (
        (DEFINED(EGID) AND DEFINED(EDID)) =>
            IliValidGwr_V1_0.edidExistsInGWR(EGID,EDID,PARAMETER IliVRuntime_V1_0.Scope)
    );
```



01

02

03

04

► 05

06

07

08

09

10

11

12

13

14

Konventionell

```
01 // get CH file from BfS website
02 File gwrFile=null;
03 try {
04     gwrFile = gwr.getLocalGwrDump(); // 900GB. Neuer Download nach 12h.
05 }
06 ...
07
08 try {
09     ret =
10     ch.interlis.ilirepository.IliManager.getLocalCopyOfReposFile(iliManager,fileName,MAX_TTL);
11 } catch (Ili2cException e) {
12     throw new IOException("failed to get GWR dump file",e);
13 }
14
```

```
01 private boolean egidExistsInGWR(int egid,String municipalityId, File gwrFile) throws SQLException
02 {
03     ...
04     try {
05         jdbcConnection = DriverManager.getConnection("jdbc:sqlite:"+gwrFile, null, null);
06         if(municipalityId!=null) {
07             stmt=jdbcConnection.prepareStatement("SELECT EGID FROM building WHERE EGID=? AND
08             GGDENR=?");
09             stmt.setInt(1,egid);
10             stmt.setString(2,municipalityId);
11
12
13
14
```

```
stmt = jdbcConnection.prepareStatement(  
    "SELECT entrance.EGID, entrance.EDID " +  
    "FROM entrance " +  
    "INNER JOIN building ON entrance.EGID = building.EGID " +  
    "WHERE entrance.EGID = ? " +  
    "AND entrance.EDID = ? " +  
    "AND GGDENR = ?"  
);
```



Cloud Native

02

03

04

05

06

07

08

▶ 09

10

11

12

13

14

15

```
03 var buildingUrl = "https://example.com/building-sorted.parquet"
04
05 // DuckDB 🦆 to the rescue
06 PreparedStatement statement =
07     connection.prepareStatement(
08         "SELECT EGID " +
09         "FROM read_parquet(?) " +
10         "WHERE EGID = ? AND GGDENR = ?")) {
11
12
13 statement.setString(1, buildingUrl);
14 statement.setLong(2, testCase.egid);
15 statement.setInt(3, testCase.ggdenr);
16
```

```
var buildingUrl = "https://example.com/building-sorted.parquet"
var entranceUrl = "https://example.com/entrance-sorted.parquet"
```

```
PreparedStatement statement =
    connection.prepareStatement(
        "SELECT e.EGID, e.EDID " +
        "FROM read_parquet(?) AS e " +
        "INNER JOIN read_parquet(?) AS b " +
        "  ON e.EGID = b.EGID " +
        "WHERE e.EGID = ? " +
        "AND e.EDID = ? " +
        "AND b.GGDENR = ?")) {
```

```
statement.setString(1, entranceUrl);
statement.setString(2, buildingUrl);
statement.setLong(3, testCase.egid);
statement.setInt(4, testCase.edid);
statement.setInt(5, testCase.ggdenr);
```



Resultate

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

Testsetup

Ausführung

- › DuckDB 1.5.5.1 mit 4 Threads
- › Parquet-Dateien remote über HTTPS
- › Zstandard-Kompression
- › 100'000 Zeilen pro Row Group
- › Warm-Phase mit vorgängigem Warm-up

Testfälle

- › Je 100 Abfragen
- › 50 Treffer und 50 kontrollierte Nichttreffer
- › single: Abfrage auf building
- › join: Verknüpfung von entrance und building

Messwerte

- › Query gesamt: Summe der 100 Abfragen
- › Median über die ausgeführten Warm-Läufe
- › Requests und MB aus separatem HTTP-Profiling

Layout

- › unsorted: Datensätze zufällig verteilt; kaum Row-Group-Pruning
- › sorted: Datensätze nach den Abfrageschlüsseln sortiert; nicht relevante Row Groups können übersprungen werden

Verbindung

- › per_call: Neue DuckDB-Verbindung und neues Prepared Statement pro Abfrage
- › session_reuse: Eine Verbindung und vorbereitete Statements für alle 100 Abfragen

Cache

- › Der Warm-up füllt Datei- und Metadaten-Cache
- › Bei session_reuse bleibt dieser Cache erhalten

Parquet-Benchmark

Beispiel	Backend	Layout	Verbindung	Query gesamt (s)	Median (ms)	Requests	Bytes (MB)
single	parquet	unsorted	per_call	222	2'376	9'344	1'565
single	parquet	unsorted	session_reuse	<1	9	0	0
single	parquet	sorted	per_call	68	689	584	42
single	parquet	sorted	session_reuse	<1	1	0	0
join	parquet	unsorted	per_call	480	4'758	19'384	2'966
join	parquet	unsorted	session_reuse	2	18	0	0
join	parquet	sorted	per_call	162	1'622	4'382	66
join	parquet	sorted	session_reuse	<1	2	0	0

SQLite-Benchmark

Beispiel	Backend	Verbindung	Query gesamt (ms)	Median (ms)
single	sqlite	per_call	12.571	0.120
single	sqlite	session_reuse	0.659	0.006
join	sqlite	per_call	16.627	0.145
join	sqlite	session_reuse	0.925	0.008



Fazit?

03

04

05

06

07

08

09

10

11

12

13

14

15

► 16