gtfs2rdf

gtfs2rdf: An Efficient, Customisable GTFS-to-RDF Converter

GTFS

- Fahrplandaten
- ZIP-Archiv von CSV-Dateien
- relational
- strukturell einfach, aber Semantik über Dateien verteilt
- wird fortlaufend weiterentwickelt

```
-- stops.txt --
stop_id,stop_name,stop_lat,stop_lon,location_type
de:08311:100,Bertoldsbrunnen,47.9959,7.8504,0
```

```
-- routes.txt --
route_id,agency_id,route_short_name,route_type
R1,AG1,1,0
```

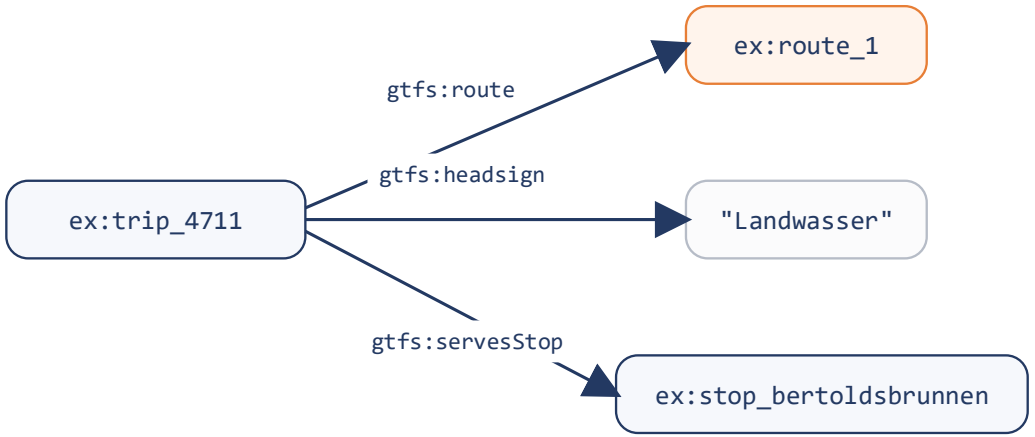
```
-- trips.txt --
route_id,service_id,trip_id,trip_headsign,shape_id
R1,WK,T4711,Landwasser,S17
```

```
-- stop_times.txt --
trip_id,arrival_time,departure_time,stop_id
T4711,08:15:00,08:15:00,de:08311:100
```

RDF

- **Tripel → Wissensgraph**
- **effizient abfragbar**
- **sehr leichte Verknüpfbarkeit**
- **reiches Ökosystem**

```
-- example.ttl --
ex:trip_4711 gtfs:route ex:route_1 .
ex:trip_4711 gtfs:headsign "Landwasser" .
ex:trip_4711 gtfs:servesStop ex:stop_bertoldsbrunnen .
```



DEMO

1. alle Stops (nur zeilenweise)

2. alle Trips mit Stop in Freiburg (Aggregation)

```
-- shapes.txt --
shape_id,shape_pt_lat,shape_pt_lon,shape_pt_sequence
S17,47.9951,7.8510,1
S17,47.9948,7.8532,2
...
```

→

```
-- output.ttl --
LINESTRING(7.8510 47.9951,
           7.8532 47.9948, ...)
```

3. alle Trips mit Stop in Freiburg zu bestimmtem Datum (Fan-out)

```
-- calendar.txt --
service_id,monday,tuesday,...,sunday,start_date,end_date
WK,1,1,...,0,20260401,20260630
SU,0,0,...,1,20260401,20260630
```

★

```
-- calendar_dates.txt --
service_id,date,exception_type
WK,20260413,2
SU,20260413,1
```

Schlüssel-Anforderungen

- **Effizienz**
- **flexible Mappings**
- **GTFS-spezifische Semantik**

Motivation - Zusammenfassung

- **GTFS als spezialisiertes Quellformat**
  grundsätzlich einfach, aber relational → Semantik über Dateien verteilt
- **RDF als allgemeines Zielformat**
  leichte Verknüpfung und schnelle Abfragen
- **passendes RDF-Ökosystem für Fahrplandaten**
  z.B. QLever, QLever Petrimaps, osm2rdf
- **Anforderungen**
  Effizienz, Flexibilität, Unterstützung von GTFS-spezifischer Semantik

Nutzerperspektive

- **Mapping nicht Teil der „Engine“**
- **embedded DSL**

```
-- stops.txt --
stop_id,stop_name,stop_lat,stop_lon
de:08311:100,Bertoldsbrunnen,47.9959,7.8504
```



```
-- stops.cppm ... --
{ SUBJ, {"rdf", "type"}, {IRI("gtfs", "Stop")} },
{ SUBJ, {"gtfs", "name"}, {"{stop_name}", "de"} },
{ SUBJ, {"geo", "asWKT"},
  {"POINT({stop_lon} {stop_lat})", IRI("geo","wktLiteral")} },
```

- **Transformationen (+ Library)**

```
-- ... stops.cppm --
{ SUBJ, {"wgs", "lat"}, {"{stop_lat | isValidDecimal}",
  IRI("xsd", "decimal")} }
```

Semantikperspektive

- mehr als nur zeilenweise
- Fan-out → Transform2Many
- Persistenz zwischen CSV-Dateien → Speichermodul mit SQLite
- Persistenz + Transform → Shape-Aggregation
- Single-pass → topologische Sortierung

Systemperspektive

- möglichst viel vor Renderschleife vorverarbeiten
- buffered I/O
- streaming execution
- single-pass
- weitere, allgemeingültige Optimierungsprinzipien

$\Longrightarrow$ schmale Renderschleife & kontrollierbarer RAM-Verbrauch

Implementierung - Zusammenfassung

- **Flexibilität**
  - user-erweiterbare Mappings mithilfe von DSL
  - Integration von in C++ definierten Transformationen + kleine Library
- **Effizienz**
  Streaming, buffered I/O, single-pass, schmale Renderschleife
- **GTFS-spezifische Semantik**
  Fan-out, Persistenz zwischen Dateien (SQLite-backed), topologische Sortierung

Setup

- **Metriken**
  - Wall-clock time
  - RSS
  - Input-Größe / Zeilenanzahl
  - Tripelanzahl

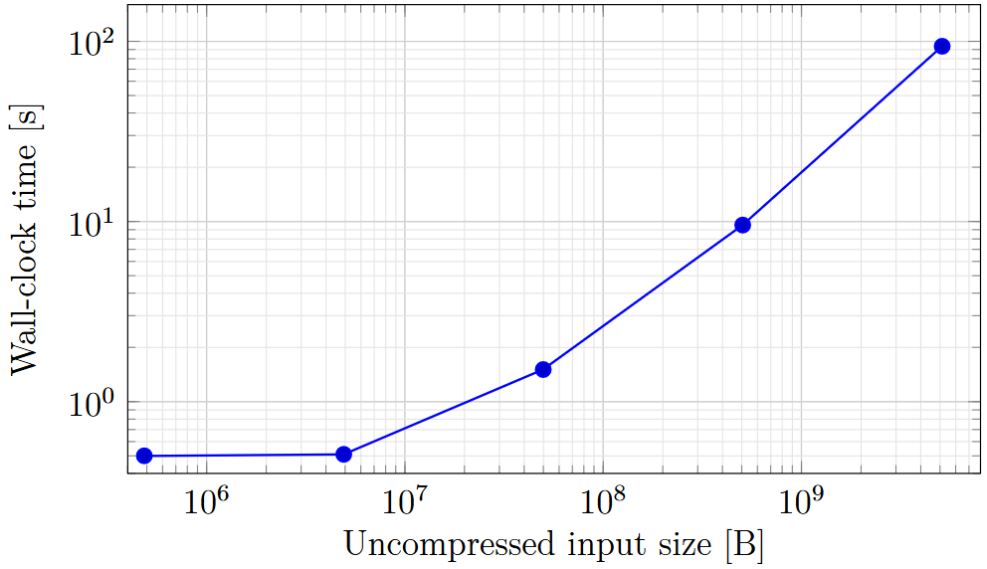
- **Datensätze**
  - Madrid-1 ... Madrid-10000
    - synthetisch
    - zeilenweise + Transformationen
    - keine shapes.txt

    → Skalierbarkeit
  - GTFS.de-real / -shuffled
    - realer Feed für ganz DE
    - vollständiges Mapping
    - insb. mit shapes.txt (ca. 3.3 GB)
    - original / zufällig permutiert

    → Praktikabilität

Ergebnisse - Madrid

Dataset	Input size [B]	Rows	Triples	Time [s]	RSS [MB]
Madrid-1	485 112	3846	38 816	0.50	13
Madrid-10	4 913 037	38 460	389 990	0.51	16
Madrid-100	49 841 601	384 600	3 899 900	1.51	169
Madrid-1000	505 651 366	3 846 000	38 999 000	9.57	169
Madrid-10000	5 129 009 781	38 460 000	389 990 000	94.01	169



Ergebnisse - Deutschland

Dataset	Input size [B]	Rows	Triples	Time [s]	RSS [MB]
GTFS.de-real	5 893 951 972	133 562 578	295 187 794	196.94	623
GTFS.de-shuffled	5 893 951 972	133 562 578	295 187 794	1871.85	623

- realer Feed sehr schnell (ca. 1.5 Mio. Tripel / s)
- Stresstest merklich langsamer (ca. 150 Tsd. Tripel / s)
- Flaschenhals Speicherstruktur wenn shapes.txt nicht optimiert

Takeaway

- Streaming (zeilenweise) skaliert gut
- RAM ist gut regulierbar
- praktikabel auf großem, realem Feed
- aber Stresstest-Feeds höhere Laufzeit, gleiches RSS

Quellen und Datengrundlage

- GTFS-Standard / Referenz: GTFS.org
- Realdatensatz Deutschland: GTFS.de
- Benchmark-Datensätze Madrid-1 bis Madrid-10000: erzeugt mit dem Skript des GTFS-Benchmarks (https://github.com/oeg-upm/gtfs-bench)
- Messungen, Auswertung und Abbildungen: eigene Experimente und eigene Darstellung