Friedrich-Alexander-Universität
Medizinische Fakultät

SQL-on-FHIR v2
Standardkonforme Datenanalyse on FHIR

Christian Gulden

Motivation

Motivation

“FHIR ist nur ein Interoperabilitätsstandard und nicht für die Datenanalysen geeignet”

Wie kommt’s?

- FHIR-Ressourcen sind verschachtelt, während Datenanalysen typischerweise mit Tabellen funktionieren


```
{
  "resourceType" : "Condition",
  "id" : "mii-exa-diagnose-condition-multiple-codings",
  "extension" : [{
    "url" : "http://hl7.org/fhir/StructureDefinition/condition-assertedDate",
    "valueDateTime" : "2020-01-13T14:00:00+01:00"
  }],
  "code" : {
    "coding" : [{
      "system" : "http://fhir.de/CodeSystem/bfarm/icd-10-gm",
      "version" : "2020",
      "code" : "H83.8",
      "display" : "Sonstige näher bezeichnete Krankheiten des Innenohres"
    },
    {
      "system" : "http://fhir.de/CodeSystem/bfarm/alpha-id",
      "version" : "2020",
      "code" : "I125918"
    }]
  },
  "subject" : {
    "reference" : "Patient/mii-exa-person-patient-1"
  },
  "onsetDateTime" : "2020-01-13T14:00:00+01:00",
  "recordedDate" : "2020-01-13T16:00:00+01:00"
}
```


?

Alter	Geschlecht	ICD-Code	Alpha-ID	Feststelldatum	Feststellzeit
90	m	H83.8	I125918	2020-01-13	14:00:00

Wie kommt’s?

- „FHIR“ wird oft gleichgesetzt mit „FHIR-Server“
  - REST API sicher am stärksten ausspezifiziert
  - … aber mit FHIR Search am wenigsten für Datenanalysen geeignet
    - „Wie viele Laborwerte habe ich pro Code?“

```sql
SELECT code, COUNT(*) AS num
FROM observation
GROUP BY code
ORDER BY COUNT(*) DESC
```

```
GET /fhir/Observation
```

1. Alle Observation Ressourcen herunterladen
2. **Daten einlesen in Pandas, Polars, o.ä.**
3. group-by, count, order-by

FHIR Search Data Science

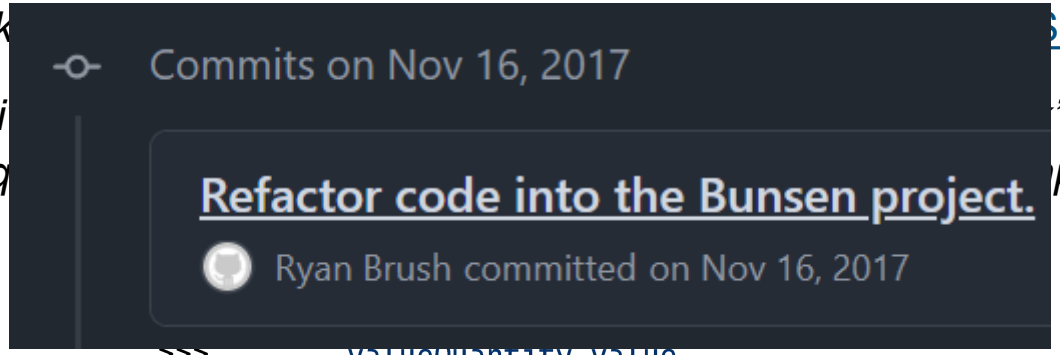
- Existierende Frameworks für FHIR-Search basierte Datenextraktion:
  - FhirExtinguisher (https://github.com/JohannesOehm/FhirExtinguisher)
  - FHIR-PYrate (https://github.com/UMEssen/FHIR-PYrate)
  - fhircrackr (https://github.com/POLAR-fhiR/fhircrackr)
  - fhiry (https://github.com/dermatologist/fhiry)
  - TORCH (https://github.com/medizininformatik-initiative/torch)

- Anekdotische Problem: Performance

Was dann?

Analytics-on-FHIR Geschichte

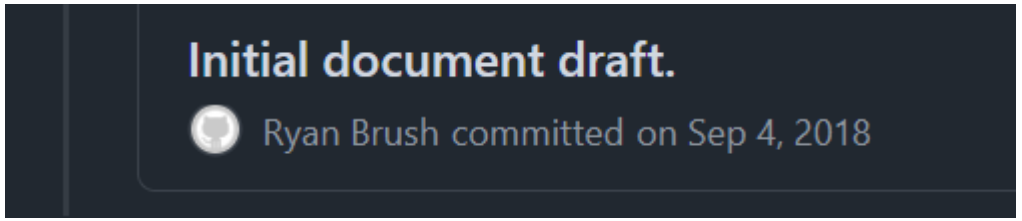
- 2017: Cerner Bunsen
  - *Bunsen lets users load, transform, and analyze FHIR data with Apache Spark. It offers Java and Python APIs to convert FHIR resources into Apache Spark Datasets, which then can be explored with the full power of that platform, including with Spark*
  - *By encoding FHIR in this nati*




```
>>>         valueQuantity.value
>>> from observations
>>> where in_valueset(code, "heart_rate")
>>> limit 5
>>> """).show()
+--------------+-----------------+-------+
|     reference|effectiveDateTime|  value|
+--------------+-----------------+-------+
|Patient/9995679|       2006-12-27|54.0000|
|Patient/9995679|       2007-04-18|60.0000|
|Patient/9995679|       2007-07-18|80.0000|
|Patient/9995679|       2008-01-16|47.0000|
|Patient/9995679|       2008-06-25|47.0000|
+--------------+-----------------+-------+
```

Analytics-on-FHIR Geschichte

- 2018: SQL-on-FHIR v1 (https://github.com/FHIR/sql-on-fhir)



  - *An SQL-based projection of FHIR resources would open up large, portable datasets to a number of analytic tools. […] Importantly, this approach preserves the nested structures and arrays of FHIR resources using ANSI SQL standards. The results of these queries can then be used as arbitrary tables for further analysis in R or other tools.* - https://github.com/FHIR/sql-on-fhir/blob/master/sql-on-fhir.md

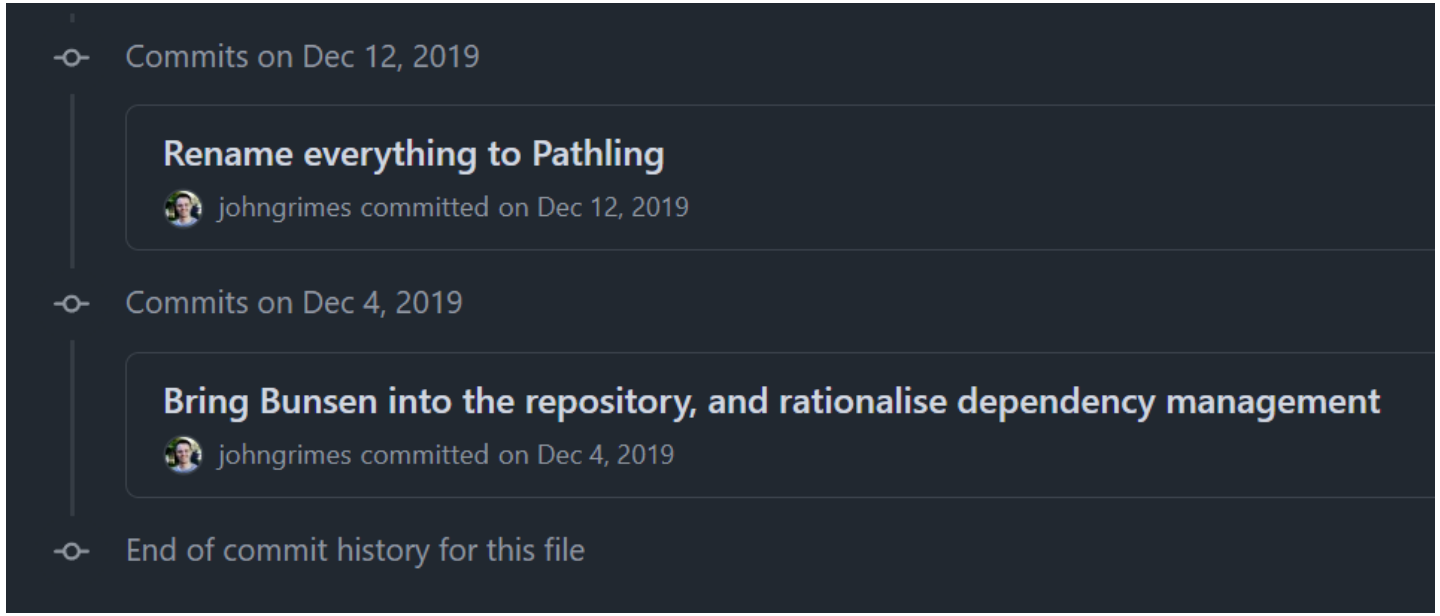
SQL-on-FHIR v1

- https://github.com/FHIR/sql-on-fhir/blob/master/examples/diabetics_with_high_hba1c.sql

```sql
-- Example query that creates multiple views of underlying FHIR resources,
-- then uses them to find all diabetics with a high HbA1c level.

-- A real query would likely use a valueset rather than specified code values,
-- but this query illustrates the pattern of layering views of underlying data
-- to analyze FHIR resources at scale.

-- Create a view of everyone with a diabetes condition
WITH diabetics as (

  SELECT DISTINCT subject.reference person_ref
  FROM condition,
       UNNEST(code.coding) coding
  WHERE verificationstatus = 'confirmed' AND
        coding.system = 'http://snomed.info/sct' AND
        (coding.code = '15777000' OR -- Prediabetes
         coding.code = '44054006') -- Diabetes
),

-- Create a view of all HbA1c values.
hba1c_values as (
  SELECT subject.reference person_ref,
         value.quantity.value,
         value.quantity.unit,
         coding.system,
         coding.code,
         coding.display,
         effectivedatetime
  FROM observation o,
       UNNEST(code.coding) coding
  WHERE coding.system = 'http://loinc.org' AND
        coding.code = '4548-4' AND
        status = 'final'

)
-- Use the above views to find diabetics who have had a
-- high HbA1c value since the start of 2017
SELECT d.person_ref,
       h.value,
       h.display result_display,
       h.effectivedatetime
FROM diabetics d
JOIN hba1c_values h ON d.person_ref = h.person_ref
WHERE h.value > 6.5 AND
      h.effectivedatetime > '2017'
```

Analytics-on-FHIR Geschichte

- 2020: Pathling (https://github.com/aehrc/pathling)
  - *Tools that make it easier to use FHIR and clinical terminology within data analytics, built on Apache Spark.*
  - Pathling-Bibliotheken in Java, R, Python
  - Encoding von FHIR-Ressourcen im Spark-Format
  - Terminologie-Server-Integration



Analytics-on-FHIR Geschichte

- 2020: Pathling (https://github.com/aehrc/pathling)

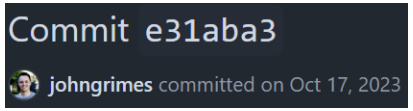
```
from pathling import PathlingContext, Expression as exp

pc = PathlingContext.create()
data = pc.read.ndjson("s3://somebucket/synthea/ndjson")

# For patients that have not received a COVID-19 vaccine, extract the given
# name, family name, phone number and whether the patient has heart disease.
result = data.extract("Patient",
                      columns=[
                          exp("name.first().given.first()", "Given name"),
                          exp("name.first().family", "Family name"),
                          exp("telecom.where(system = 'phone').value",
                              "Phone number"),
                          exp("reverseResolve(Condition.subject).exists("
                              "code.subsumedBy(http://snomed.info/sct|56265001))",
                              "Heart disease")
                      ],
                      filters=[
                          "reverseResolve(Immunization.patient).vaccineCode"
                          ".exists(memberOf('https://aehrc.csiro.au/fhir/ValueSet/covid-19-vaccines'))"
                          ".not()"]
                      )
```

```
{
  "resourceType": "Patient",
  "name": [
    {
      "family": "Smith",
      "given": [
        "John"
      ]
    }
  ],
  "gender": "male",
  "birthDate": "1956-05-27"
}
```

Given name	Family name	Phone number	Heart disease
John	Smith	0412345678	false
Jane	Doe	0412345678	true

Analytics-on-FHIR Geschichte

- 2023: SQL-on-FHIR v2 (https://github.com/FHIR/sql-on-fhir-v2)



  - *This specification proposes an interoperable approach to make large-scale analysis of FHIR data accessible to a larger audience and portable between systems. The central goal of this project is to make FHIR data work well with the best available analytic tools regardless of the technology stack. - https://build.fhir.org/ig/FHIR/sql-on-fhir-v2/*

SQL-on-FHIR v2

SQL-on-FHIR v2

- Problem: FHIR für Datenanalyse nutzbar machen
- Lösung: ein Format um tabulare, Use-Case spezifische Views über FHIR-Ressourcen zu definieren. Diese Views können anschließend von Data-Science Tools und Frameworks genutzt werden.

SQL-on-FHIR v2 ViewDefinition

- ViewDefinition Ressourcen

```
{
  "resourceType": "ViewDefinition",
  "resource": "Patient",
  "name": "demo",
  "select": [
    {
      "column": [
        {"name": "id",           "path": "getResourceKey()"},
        {"name": "Geschlecht",   "path": "gender"},
        {"name": "Geburtsdatum", "path": "birthDate"}
      ]
    },
    {
      "forEach": "name.where(use = 'official').first()",
      "column": [
        {"name": "Vorname",  "path": "given.join(' ')"},
        {"name": "Nachname", "path": "family"}
      ]
    }
  ]
}
```

```
{
  "resourceType": "Patient",
  "id": "269…",
  "name": [
    {
      "use": “official",
      "family": "Cole117",
      "given": [
        "Joanie498",
        "Berneice173"
      ]
    }
  ],
  "gender": "female",
  "birthDate": "2012-03-30"
}
```

id	Geschlecht	Geburtsdatum	Vorname	Nachname
269a6a02-37e2-bd1a-e079-fda944434a99	female	2012-03-30	Joanie498 Berneice173	Cole117
71838e94-fb71-a349-4a00-39f41688c86c	male	1999-05-16	Dannie881 Eliseo499	Gerlach374
f1c4c019-1efe-4c98-0001-38a3e28eea8e	female	2015-12-04	Marna905 Cristen212	Heller342
0f41b9d0-e743-9e8a-ea8f-0e2a2707a47f	male	2009-04-07	Nathanael908 Gino587	Metz686
49598903-5ca3-3cbb-e0aa-0e89f74767a2	female	2022-06-06	Logan497 Daniell603	Greenholt190
71be7786-eb5c-b9ce-3fda-f0d55a9c734e	female	1981-12-06	Reanna349 Denisse335	Bode78

Condition ViewDefinition

```
{
  "resourceType": "ViewDefinition",
  "resource": "Condition"
  "name": "diagnoses_vie
  "select": [
    {
      "column": [
        {"name": "condition_id", "path": "getResourceKey()"},
        {"name": "onset", "path": "onset.dateTime"},
        {"name": "abatement", "path": "abatement.dateTime"},
        {"name": "status", "path": "clinicalStatus.coding.code.first()"},
        {"name": "code", "path": "code.coding.where(system='http://fhir.de/CodeSystem/bfarm/icd-10-gm').code.first()"},
        {"name": "display", "path": "code.text"},
        {"name": "patient_id", "path": "subject.getReferenceKey(Patient)"}
      ]
    }
  ]
}
```

Primärschlüssel der Ressource (typischerweise Resource.id)

Fremdschlüssel auf Patient

```
{
  "resourceType" : "Condition",
  "code" : {
    "coding" : [{
      "system" : "http://fhir.de/CodeSystem/bfarm/icd-10-gm",
      "version" : "2020",
      "code" : "H83.8",
      "display" : "Sonstige näher bezeichnete Krankheiten des Innenohres"
    },
    {
      "system" : "http://fhir.de/CodeSystem/bfarm/alpha-id",
      "version" : "2020",
      "code" : "I125918"
    }]
  }
}
```

ViewDefinition

- 1 Resourcentyp pro ViewDefinition
- Keine Implementierung von oder Vorgaben zu Joins zwischen Tabellen/Views (Pathling hatte das noch implementiert via `resolve`)
- Keine Implementierung von Sortierung, Aggregation, Limit
- Agnostisch gegenüber dem Ausgabeformat: CSV-Datei, Tabellen-Datenbank, Parquet-Export, …

SQL-on-FHIR v2 Layers

Analytics Layer

Any analytics tool chosen by the user to consume and work with the views

View Layer

Portable, tabular views of FHIR data generated by and intended for consumption by a wide variety of analytic tools. View Definitions and View Runners are the key components of this layer.

Data Layer

Lossless representations of data as FHIR Resources that collectively enable FHIR to be used with a wide variety of different query technologies. It may optionally be persisted and annotated.

View Runners

- Tools die ViewDefinition auf FHIR-Ressourcen (im Data Layer) ausführen
- z.B. eine Python Bibliothek mit Apache Spark als Query-Engine auf FHIR-Ressourcen im NDJSON-Format
- SQL-on-FHIR v2.1.0-pre
  - `ViewDefinition/[id]/$run` um ViewDefinition via REST auszuführen (synchron, Echtzeit)
  - `ViewDefinition/[id]/$export` um große Datenmenten asynchron als NDJSON, Parquet, CSV zu exportieren

Implementierungen

- https://sql-on-fhir.org/extra/impls.html
  - Aidbox
  - Pathling
  - Google (Open Health Stack)
  - NCQA Fhir Analytics
  - Medplum
  - Safhire
  - FlatQuack
  - FHIR4DS
  - SAS (Python Library)
  - MSSQL on FHIR
  - Helios Software

Pathling

Pathling

- Bibliothek für Java, Python und R um FHIR-Ressourcen via ViewDefinition in Tabellen zu überführen
- Apache Spark als Query-Engine
- Ergebnis der „Verflachung“ ist ein Spark DataFrame
- Dieser DataFrame ermöglicht alle typischen Operationen wie Group-By, Sort, Aggregate, etc.
- Nahtloser Übergang von FHIR ins Spark-Data-Science und ML Ökosystem

Pathling

1. FHIR-Ressourcen müssen in kompatibler Form vorliegen: Bundle, NDJSON (z.B. via FHIR-Server Bulk Export), Parquet, Delta Lake

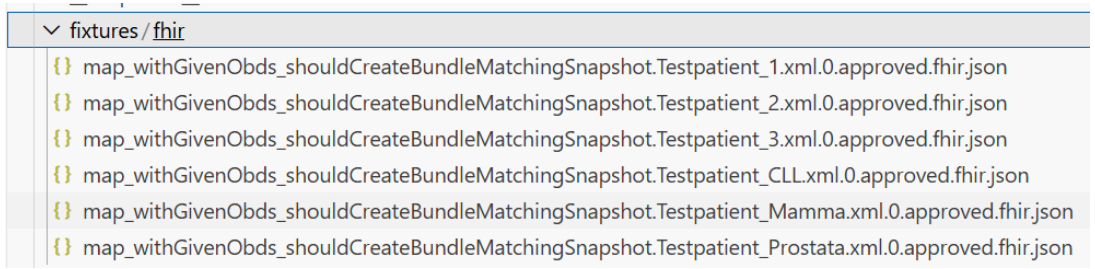
2. Ressourcen einlesen

3. View definieren (FHIRPath)

4. Auf erzeugtem Spark DataFrame arbeiten (filter, join, group-by, etc.)

Pathling Python

1. JSON-Bundles:



2. Einlesen:

```
from pathling import PathlingContext


pc = PathlingContext.create()
data = pc.read.ndjson("./fixtures/fhir")
```

https://github.com/bzkf/onco-analytics-on-fhir/tree/master/analytics_on_fhir

Pathling Python

3. Views definieren

```
patients = data.view(
    "Patient",
    select=[
        {
            "column": [
                {
                    "path": "getResourceKey()",
                    "name": "patient_id",
                },
                {
                    "path": "identifier.where("
                    + f"system='{self.settings.fhir.patient_identifier_system}').value",
                    "name": "patient_mrn",
                },
            ],
        }
    ],
)
```

Pathling Python

3. Views definieren

```
FHIR_CODE_SYSTEM_ICD10 = "http://fhir.de/CodeSystem/bfarm/icd-10-gm"
FHIR_SYSTEMS_CONDITION_ASSERTED_DATE = "http://hl7.org/fhir/StructureDefinition/condition-assertedDate"
conditions = self.data.view(
    "Condition",
    select=[
        {
            "column": [
                {
                    "path": "getResourceKey()",
                    "name": "condition_id",
                },
                {
                    "path": "subject.getReferenceKey()",
                    "name": "condition_patient_reference",
                },
                {
                    "path": f"extension('{FHIR_SYSTEMS_CONDITION_ASSERTED_DATE}')"
                    + ".value.ofType(dateTime)",
                    "name": "diagnosis_assertedDateTime",
                },
                {
                    "path": "recordedDate",
                    "name": "diagnosis_recordedDate",
                },
                {
                    "path": "onset.ofType(dateTime)",
                    "name": "diagnosis_onsetDateTime",
                },
                {
                    "path": f"code.coding.where(system = '{FHIR_CODE_SYSTEM_ICD10}').code",
                    "name": "icd_code",
                },
            ],
        }
    ],
)
```

Pathling Python

4. Filtern auf Liste von ICD-Codes *

```python
icd_codes_aml = self.data.spark.read.csv(
    (HERE / "icd_codes_aml.csv").as_posix(), header=True
)


conditions = conditions.join(
    icd_codes_aml, conditions.icd_code == icd_codes_aml.icd_code, "inner"
)


logger.info(f"Found {conditions.count()} Conditions with matching AML ICD codes")

conditions.show()

aml_patient_references = conditions.select("condition_patient_reference").distinct()
```

```
1   icd_code
2   C92
3   C92.0
4   C92.00
5   C92.01
6   C92.4
7   C92.40
8   C92.41
9   C92.5
10  C92.50
11  C92.51
12  C92.6
13  C92.60
14  C92.61
15  C92.7
16  C92.70
17  C92.71
18  C92.8
19  C92.80
20  C92.81
21  C92.9
22  C92.90
23  C92.91
```

* OK, es ginge auch mit einem Filter-Ausdruck in der ViewDefinition, aber es sind immerhin 50 codes. **

** OK, ein Regex-Filter in Spark wäre für ICD-10 teilweise auch gegangen: `conditions = conditions.filter(col("icd_code").rlike(r"^C92.*"))`

Spark Lazy Evaluation

```
result = cohort.groupBy("medication_atc_code").agg(
    count_distinct(
        when(
            col("medication_start_date") < reference_date,
            col("patient_id"),
        )
    ).alias("num_patients_before"),
    count_distinct(
        when(
            col("medication_start_date") ≥ reference_date,
            col("patient_id"),
        )
    ).alias("num_patients_after"),
)
```

Der gesamte Query-Graph wird von Spark so spät wie möglich ausgeführt, um ihn maximal optimieren zu können: ViewDefinition, Join, filter, etc. erst wenn die Daten dargestellt oder abgespeichert werden sollen.

Pathling als SQL-on-FHIR v2 Implementierung

- Man kommt schnell von verschachtelten FHIR-Ressourcen zu Tabellarischen Daten
- Diese sind direkt in einem skalierbaren Framework auswertbar (Spark)
- Und auch wieder als CSV, XSLX, Parquet und co. ausleitbar
- Signifikant performanter als Ausleitungen via FHIR-Search
- Aber:
  - Spark als Framework ist nicht trivial was die Konfiguration (insb. verteilt) und Tuning (insb. RAM) angeht.
  - FHIRPath ist auch nicht immer intuitiv und nicht alle Features vollständig implementiert in Pathling (hilfreich: https://sql-on-fhir.org/extra/playground.html und https://hl7.github.io/fhirpath.js/)

Ausblick

- SQL-on-FHIR v2.1.0
  - SQLQuery Library Resource

> *The SQLQuery profile on the FHIR Library resource represents a single, logical SQL query within the FHIR ecosystem. It bridges the View Layer and the Analytics Layer: ViewDefinitions produce the flat tables, and SQLQuery joins and aggregates them using native SQL. - https://build.fhir.org/ig/FHIR/sql-on-fhir-v2/index.html#sqlquery*

  - `$sqlquery-run & sqlquery-export`

SQL-on-FHIR v2.1.0


```
{
  "resourceType": "Library",
  "meta": {
    "profile": ["https://sql-on-fhir.org/ig/StructureDefinition/SQLQuery"]
  },
  "type": {
    "coding": [{
      "system": "https://sql-on-fhir.org/ig/CodeSystem/LibraryTypesCodes",
      "code": "sql-query"
    }]
  },
  "name": "DiagnosisByAgeSummary",
  "status": "active",
  "relatedArtifact": [
    {
      "type": "depends-on",
      "resource": "https://example.org/ViewDefinition/patient_demographics",
      "label": "pt"
    },
    {
      "type": "depends-on",
      "resource": "https://example.org/ViewDefinition/diagnoses_view",
      "label": "dg"
    }
  ],
  "content": [{
    "contentType": "application/sql",
    "extension": [{
      "url": "https://sql-on-fhir.org/ig/StructureDefinition/sql-text",
      "valueString": "SELECT DATE_PART('year', AGE(pt.dob::timestamp)) AS age, pt.gender, dg.code, dg.display, count(*) FROM pt JOIN dg USING
(patient_id) GROUP BY 1,2,3,4 ORDER BY 1, 5 DESC"
    }],
    "data":
"U0VMRUNUIERBVEVfUEFSVCgneWVhcicsIEFHRShwdC5kb2I6OnRpbWVzdGFtcCkpIEFTIGFnZSwgcHQuZ2VuZGVyLCBkZy5jb2RlLCBkZy5kaXNwbGF5LCBjb3VudCgqKSBGUk9NIHB0IEpPSU4
gZGcgVVNJTkcgKHBhdGllbnRfaWQpIEdST1VQIEJZIDEsMiwzLDQgT1JERVIgQlkgMSwgNSBERVND"
  }]
}
```


ViewDefinition

SQL Query

Elegante Möglichkeit um beliebig-komplexe (EA-) Kriterien zu formulieren?

Zusammenfassung

- FHIR wird nicht nur für den Datenaustausch sondern auch Datenanalysen eingesetzt
- Der SQL-on-FHIR v2 Standard beschreibt eine möglichst Technologie-Stack-Agnostische Implementierung, um von verschachtelten FHIR-Ressourcen zu Tabellen zu gelangen
- Mehrere Umsetzungen existieren, u.a. Pathling, welches den Standard mit geprägt hat
- Pathling erfolgreich in einem BZKF (Bayerisches Zentrum für Krebsforschung) – Projekt zur onkologischen Datenauswertung im Einsatz (https://github.com/bzkf/onco-analytics-on-fhir/)

Vielen Dank!