Joint Databases for Electronic Health Record Analysis

Teresa Lee, Tino Trangia, Micah Hunter

DSC 202

Background

- As the healthcare industry shifts toward data-driven decision making, there is demand for database systems designed for health records.
- Some challenges
  - Integration: Data often comes from disparate sources (different hospitals, providers, etc) and may have varying format.
  - Future proofing: The healthcare ecosystem is complex and ever-changing. Fixed schemas may struggle to accommodate new types of records and force data migrations.
  - Diverse queries: Some questions will be easily answered by a relational database, while others will require more complex graph analytics.

Data Types

- **Electronic health records (EHRs)** are critical for modern healthcare systems as an efficient, centralized, and interoperable way to manage patient data.
  - Record for each patient, who may have one or more existing (and previous) conditions, medications, claims, etc.
- **Social determinants of health (SDOH)** are non-medical factors within a geographic region such as economic stability, education access, demographics, and built environment characteristics.
  - I.e. “your zipcode is a better predictor of health outcomes than your genetic code.”
  - Often missing from EHRs due to lack of interoperability.

Goals

- Link patients with diseases, other patients, medical insurance companies, etc.
- Similarity queries for patients with similar conditions to improve diagnosis
- Improve data accessibility and interoperability for use by care providers, public health researchers, etc.
- Reveal correlations between social factors and medical outcomes.

Data Sources

Medical records

- Synthetic patient data for the state of Massachusetts from Synthea (SyntheticMass): https://synthea.mitre.org/downloads
- Pre-generated datasets for 1M, 1K, and 100 patients
- Circumvent issues regarding privacy
- Built using models of disease progression and detection (simpler/cleaner than real life)

Social determinants of health

- ZIP code-level data from Agency for Healthcare Research and Quality (AHRQ): https://www.ahrq.gov/sdoh/data-analytics/sdoh-data.html
- Also available at county and census tract granularities
- Not synthetic
- Can combine patient data and social determinants using patient’s address

Methodology

1. Exploratory data analysis
2. Import structured EHR data into PostgreSQL
3. Determine nodes and relationships for graphs
4. Use Python to read data into Neo4j
5. Add SDOH data to both the relational and graph databases to supplement EHRs
6. Build queries to test common use cases

EHR Data

```
patients_csv_data[0]
```


```
{'Id': '30a6452c-4297-a1ac-977a-6a23237c7b46',
 'BIRTHDATE': '1994-02-06',
 'DEATHDATE': '',
 'SSN': '999-52-8591',
 'DRIVERS': 'S99996852',
 'PASSPORT': 'X47758697X',
 'PREFIX': 'Mr.',
 'FIRST': 'Joshua658',
 'MIDDLE': 'Alvin56',
 'LAST': 'Kunde533',
 'SUFFIX': '',
 'MAIDEN': '',
 'MARITAL': 'M',
 'RACE': 'white',
 'ETHNICITY': 'nonhispanic',
 'GENDER': 'M',
 'BIRTHPLACE': 'Boston  Massachusetts  US',
 'ADDRESS': '811 Kihn Viaduct',
 'CITY': 'Braintree',
 'STATE': 'Massachusetts',
 'COUNTY': 'Norfolk County',
 'FIPS': '25021',
 'ZIP': '02184',
 'LAT': '42.21114202874998',
 'LON': '-71.0458021760648',
 'HEALTHCARE_EXPENSES': '56904.96'
```


```
encounters_csv_data[0]
```

✓ 0.0s


```
{'Id': '294d0dab-907e-8fce-7a47-0c0d322a5734',
 'START': '2012-04-01T09:04:48Z',
 'STOP': '2012-04-01T10:02:47Z',
 'PATIENT': '30a6452c-4297-a1ac-977a-6a23237c7b46',
 'ORGANIZATION': 'f2068cee-c75c-321d-9b2c-c33535db89c9',
 'PROVIDER': 'c3d07214-c20f-3f33-ad41-0e55adf5b024',
 'PAYER': 'd31fccc3-1767-390d-966a-22a5156f4219',
 'ENCOUNTERCLASS': 'wellness',
 'CODE': '162673000',
 'DESCRIPTION': 'General examination of patient (procedure)',
 'BASE_ENCOUNTER_COST': '136.80',
 'TOTAL_CLAIM_COST': '1567.00',
 'PAYER_COVERAGE': '87.20',
 'REASONCODE': '',
 'REASONDESCRIPTION': ''}
```


```
medications_csv_data[0]
```

✓ 0.0s


```
{'START': '2015-09-28T11:02:48Z',
 'STOP': '2015-10-15T09:04:48Z',
 'PATIENT': '30a6452c-4297-a1ac-977a-6a23237c7b46',
 'PAYER': 'd31fccc3-1767-390d-966a-22a5156f4219',
 'ENCOUNTER': '953c5138-ce17-4084-3432-1ac23f184528',
 'CODE': '857005',
 'DESCRIPTION': 'Acetaminophen 325 MG / HYDROcodone Bitartrate 7.5 MG Oral Tablet',
 'BASE_COST': '2.51',
 'PAYER_COVERAGE': '0.00',
 'DISPENSES': '1',
 'TOTALCOST': '2.51',
 'REASONCODE': '',
 'REASONDESCRIPTION': ''}
```

SDOH Data

```python
# Filter rows for Massachusetts - adjust the column name and value as needed
df_ma = df[df['STATE'] == "Massachusetts"]
df_ma.head(10)
```

	YEAR	STATEFIPS	ZIPCODE	ZCTA	STATE	REGION	TERRITORY	POINT_ZIP	ACS_TOT_POP_WT_ZC	ACS_TOT_POP_US_ABOVE1_ZC	...	CEN_POPDENSITY_ZC	HIFLD_DIST_UC_ZP	PO
227	2020	25	1001	1001.0	Massachusetts	Northeast	0	0	16064.0	15854.0	...	1403.92	5.59	
228	2020	25	1059	1002.0	Massachusetts	Northeast	0	1	30099.0	29954.0	...	546.85	9.25	
229	2020	25	1002	1002.0	Massachusetts	Northeast	0	0	30099.0	29954.0	...	546.85	7.81	
230	2020	25	1004	1002.0	Massachusetts	Northeast	0	1	30099.0	29954.0	...	546.85	7.07	
231	2020	25	1003	1003.0	Massachusetts	Northeast	0	0	11588.0	11588.0	...	16290.28	7.63	
232	2020	25	1005	1005.0	Massachusetts	Northeast	0	0	5166.0	5145.0	...	116.77	18.74	
233	2020	25	1007	1007.0	Massachusetts	Northeast	0	0	15080.0	14972.0	...	286.46	11.03	
234	2020	25	1008	1008.0	Massachusetts	Northeast	0	0	1116.0	1111.0	...	20.75	18.52	
235	2020	25	1009	1009.0	Massachusetts	Northeast	0	0	649.0	649.0	...	814.00	10.60	
236	2020	25	1010	1010.0	Massachusetts	Northeast	0	0	3663.0	3643.0	...	105.43	16.05	

10 rows × 327 columns

Relational Data

- Not much real medical data publicly available → utilizing synthetic data
- Synthetic data from the site SyntheticMass
- Around 1,000 patient entries
- Files in CSV format
- Tables included (and more):
  - Patients
  - Conditions
  - Medications
  - Care Plans
  - Procedures

- Features included (and more):
  - Patients
    - Patient id, birthdate, deathdate, SSN, name, race, ethnicity, gender, address, etc.
  - Conditions
    - Start, stop, patient id, description of condition
  - Medications
    - Start, stop, patient id, payer, description of medication, cost
  - Care Plans
    - Start, stop, patient id, description, reason
  - Procedures
    - Start, stop, patient id, description, cost, coverage, reason

Graph Data

- 100 synthetic medicare patient records (shrinked dataset but same characteristics as the 1000 record data)
  - Approx. 140,000 nodes from 100 patient records
  - Over 160,000 edges
- Files in CSV format
- Patient demographics
- ICD-10 diagnosis codes
- SNOMED biomedical ontology
- Procedure codes (CPT/HCPCS)
- Provider information
- Insurance claims
- Primary Care Encounters, Emergency Room Encounters, and Symptom-Driven Encounters

Graph Elements

Node types

- Patient
- Condition
- Medication
- Encounter
- Provider
- Organization
- Observation
- Care Plan
- Payer
- Procedure
- SDOH area (zipcode)
- Claim

Relationship types

- HAS_CONDITION
- HAS_MEDICATION
- HAS_ENCOUNTER
- FROM_ENCOUNTER
- PROVIDED_BY
- LOCATED_IN
- WORKS_AT
- WITH_ORGANIZATION
- PAID_BY
- CLAIMED_BY

Relational Database: PostgreSQL

Query 1

- Question: What's the top 10 most common medical condition disorder in our data?

```sql
select description, count(*) as counts
from "Conditions"
group by description
having description like '%(disorder)%'
order by counts desc
limit 10;
```

	description	counts
1	Viral sinusitis (disorder)	1233
2	Acute viral pharyngitis (disorder)	678
3	Acute bronchitis (disorder)	571
4	Anemia (disorder)	324
5	Chronic sinusitis (disorder)	219
6	Streptococcal sore throat (disorder)	162
7	Acute bacterial sinusitis (disorder)	74
8	Hypertriglyceridemia (disorder)	71
9	Metabolic syndrome X (disorder)	68
10	Osteoporosis (disorder)	58

Query 2

- Question: For the top 10 most common medical condition disorder, find the break down of the condition between genders

```sql
-- demo example 2
create table conditionCounts as (
    select description, count(*) as counts
    from "Conditions"
    group by description
    having description like '%(disorder)%'
    order by counts desc
    limit 10
    )

create table countsGender as (
    select c.description, p.gender, count(*) as gender_count
    from "Conditions" c
    left join "Patients" p on c.patient = p.id
    where c.description in (select conditioncounts.description
                            from conditionCounts)
    group by c.description, p.gender
    )

select cc.description,
    COALESCE(SUM(CASE WHEN cg.gender = 'M' THEN cg.gender_count END), 0) AS males,
    COALESCE(SUM(CASE WHEN cg.gender = 'F' THEN cg.gender_count END), 0) AS females
from conditioncounts cc
left join countsGender cg on cc.description=cg.description
group by cc.description
ORDER BY cc.description desc
```

	description	males	females
1	Viral sinusitis (disorder)	563	670
2	Streptococcal sore throat (disorder)	65	97
3	Osteoporosis (disorder)	18	40
4	Metabolic syndrome X (disorder)	33	35
5	Hypertriglyceridemia (disorder)	35	36
6	Chronic sinusitis (disorder)	107	112
7	Anemia (disorder)	173	151
8	Acute viral pharyngitis (disorder)	317	361
9	Acute bronchitis (disorder)	283	288
10	Acute bacterial sinusitis (disorder)	36	38

Query 3

Question: Combining the synthetic data with real demographic data on zip code, find if there is a correlation between percentage of foreign born citizens in a zip code location to the number of medical conditions in that location.

```sql
with combinedTable as (
    select p.id,
           p.zip,
           co.description,
           ci.acs_pct_foreign_born_zc as percentage_foreign_born
    from "Patients" p
    join "Citizenship" ci on p.zip=ci.new_zipcode
    join "Conditions" co on p.id=co.patient
)

select zip, percentage_foreign_born, count(*) as counts
from combinedTable c
group by zip, percentage_foreign_born
order by counts desc
```

	zip	percentage_foreign_born	counts
1	02171	37.72	603
2	02116	24.34	536
3	01020	7.57	425
4	02723	22.41	419
5	01803	23.45	385
6	01970	15.77	368
7	01940	9.23	360
8	02790	10.69	354
9	02169	31.86	339
10	02138	27.85	319

Top 10 zip codes with most medical conditions

Bottom 10 zip codes with most medical conditions

212	02067	23.56	4
213	02664	12.26	4
214	01566	6.4	4
215	02191	11.62	4
216	02655	13.16	4
217	01129	8.12	3
218	01540	6.5	3
219	02554	17.31	3
220	01030	6.18	2
221	02675	6.05	2

Graph Database: Neo4j

Importing the data

- Use Python to read in CSV data from file
- Write upload function containing queries to create (or merge) nodes of a certain type
- Add relationships where necessary (e.g. a patient can have direct edges to their encounters with care providers and to their conditions)
- Use neo4j-driver to import data to the database (locally or to the cloud)

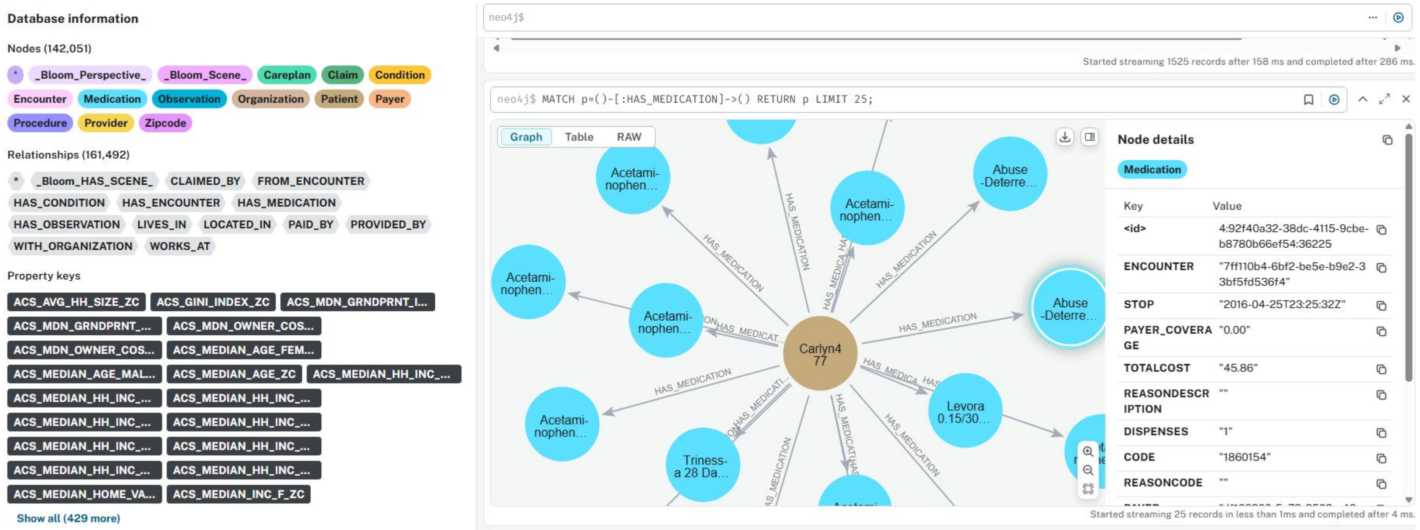
```python
encounters_csv_data = []
with open("csv/encounters.csv", newline='') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        encounters_csv_data.append(row)
```

```python
def upload_encounters(tx, records):
    cypher_query = """
    UNWIND $records as record
    MERGE (e:Encounter {Id: record.Id})
    SET e = record
    WITH e, record
    MATCH (p:Patient {Id: record.PATIENT})
    MERGE (p)-[:HAS_ENCOUNTER]->(e)
    WITH e, record
    MATCH (c:Condition {ENCOUNTER: record.Id})
    MERGE (c)-[:FROM_ENCOUNTER]->(e)
    """
    tx.run(cypher_query, records=records)

with driver.session(database="neo4j") as session:
    session.execute_write(upload_encounters, encounters_csv_data)
```

What does the data look like?

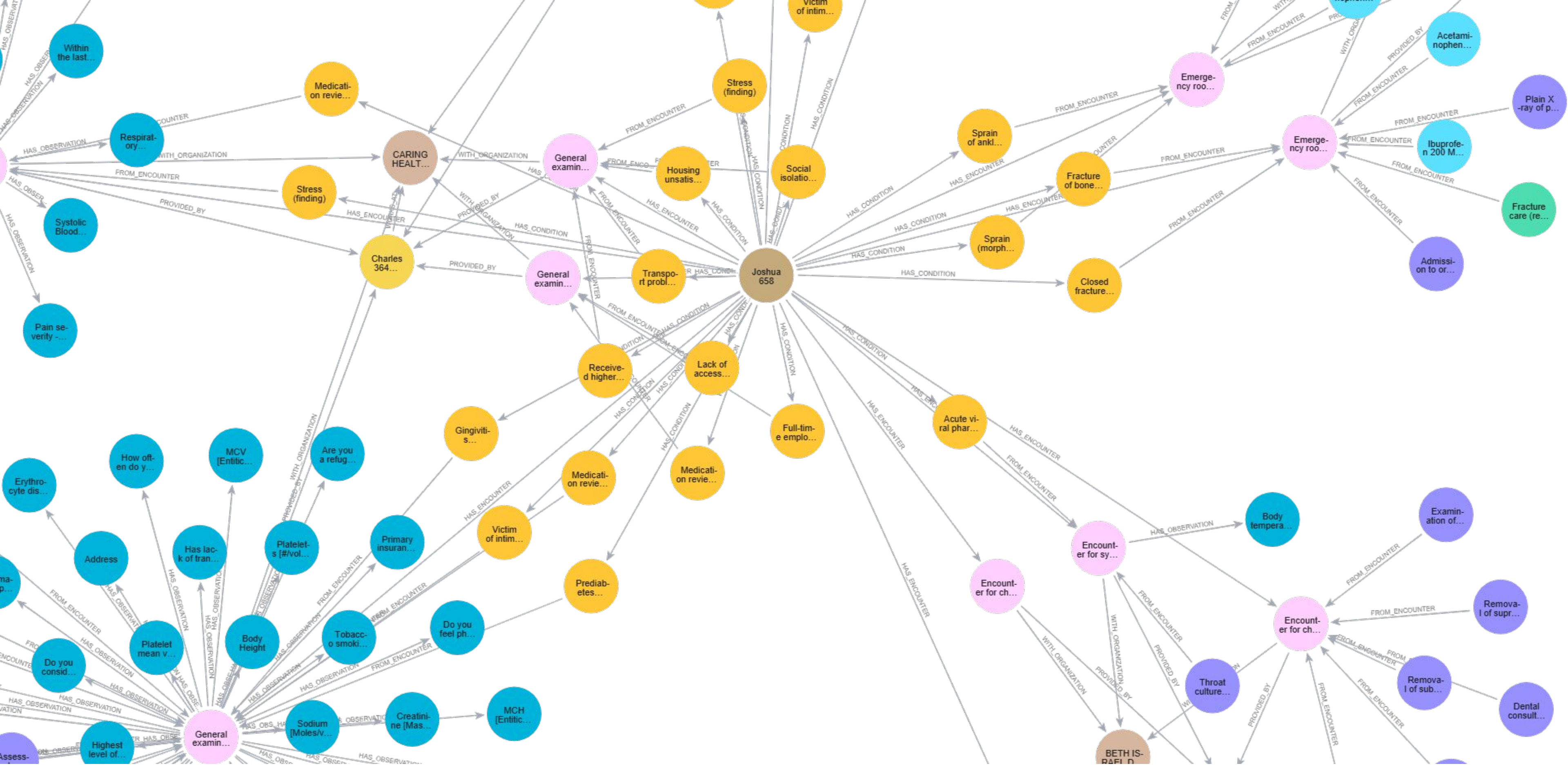
Interact with data using Neo4j browser



Query: show the medical history of a specific patient

```
MATCH path = (p:Patient {Id: "30a6452c-4297-a1ac-977a-6a23237c7b46"})-[*1..3]-(n)
RETURN path
```

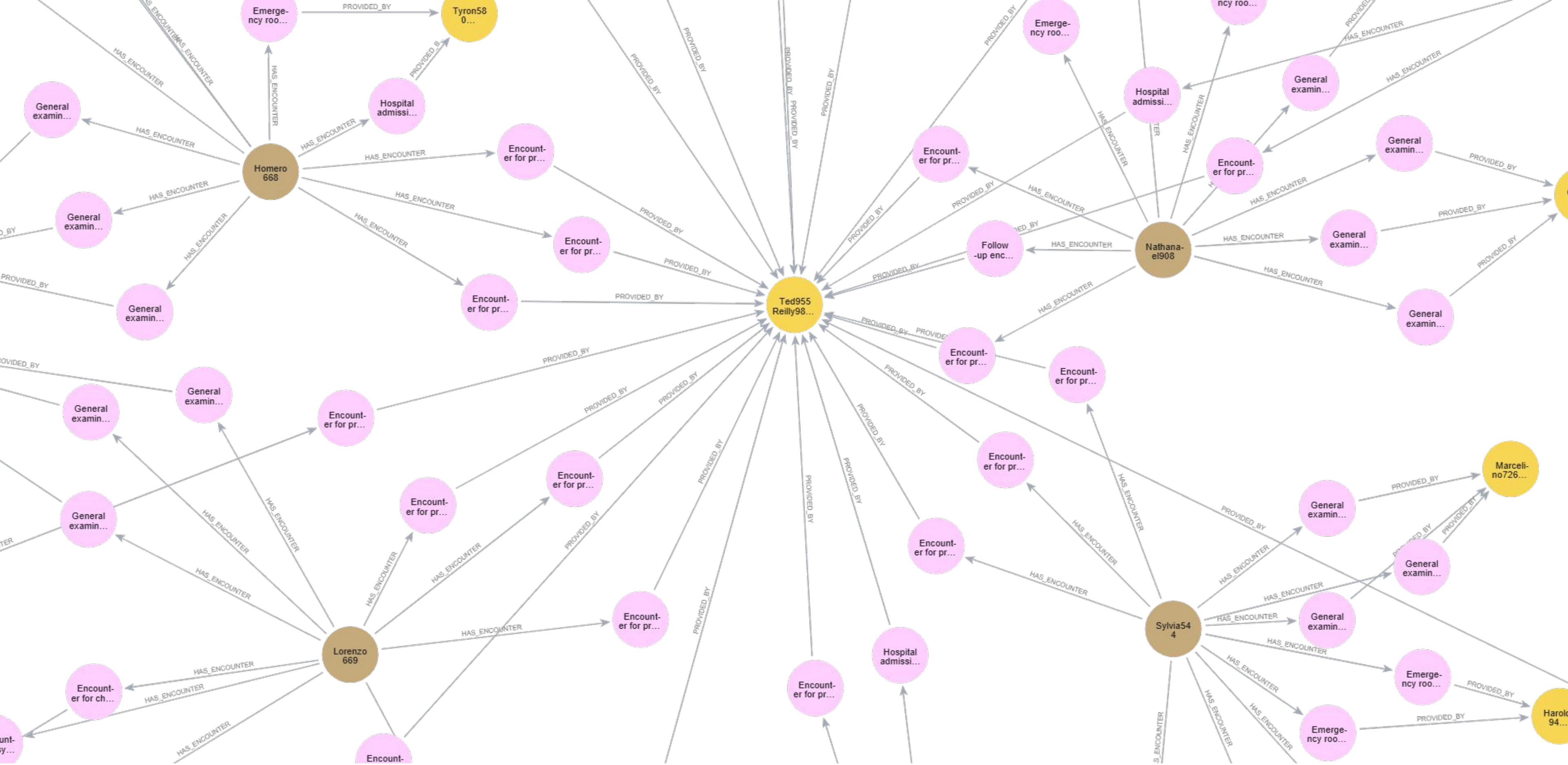
- Essentially retrieving the neighborhood of a patient



Graph for a single patient

Query: show provider-patient networks

```
MATCH
(pt:Patient)-[r:HAS_ENCOUNTER]->(e:Encounter)-[:PROVIDED_BY]->(p:Provider)
WITH pt, p, e
WITH pt, p, collect(e)[0..3] AS limitedEncounters
UNWIND limitedEncounters AS e
MATCH path = (pt)-[r:HAS_ENCOUNTER]->(e)-[:PROVIDED_BY]->(p)
RETURN pt, p, collect(path) AS paths
```



Provider-patient networks

Query: Provide a table of pre-diabetic patients who have been prescribed with insulin and their most recent encounters with a general practitioner

```
MATCH (p:Patient)-[:HAS_CONDITION]->(c:Condition {CODE:
'714628002'}),
      (p)-[:HAS_MEDICATION]->(m:Medication {CODE:
'106892'}),
      (p)-[:HAS_ENCOUNTER]->(e:Encounter),
      (e)-[:PROVIDED_BY]->(pr:Provider {SPECIALITY:
'GENERAL PRACTICE'})
RETURN DISTINCT p.FIRST AS PatientFirstName,
       p.LAST AS PatientLastName,
       pr.NAME AS ProviderName,
       e.START AS EncounterStartDate
ORDER BY PatientLastName ASC, e.START DESC;
```

PatientLastName	ProviderName	EncounterStartDate
"Balistreri607"	"Erwin847 Stiedemann542"	"2024-09-08T07:48:58Z"
"Balistreri607"	"Laurena366 Anderson154"	"2023-09-17T07:48:58Z"
"Balistreri607"	"Erwin847 Stiedemann542"	"2023-09-03T07:48:58Z"
"Balistreri607"	"Laurena366 Anderson154"	"2023-07-16T07:48:58Z"
"Balistreri607"	"Laurena366 Anderson154"	"2022-09-11T07:48:58Z"
"Balistreri607"	"Erwin847 Stiedemann542"	"2022-08-28T07:48:58Z"
"Balistreri607"	"Laurena366 Anderson154"	"2021-08-29T07:48:58Z"
"Balistreri607"	"Erwin847 Stiedemann542"	"2021-08-22T07:48:58Z"
"Balistreri607"	"Laurena366 Anderson154"	"2021-07-04T07:48:58Z"

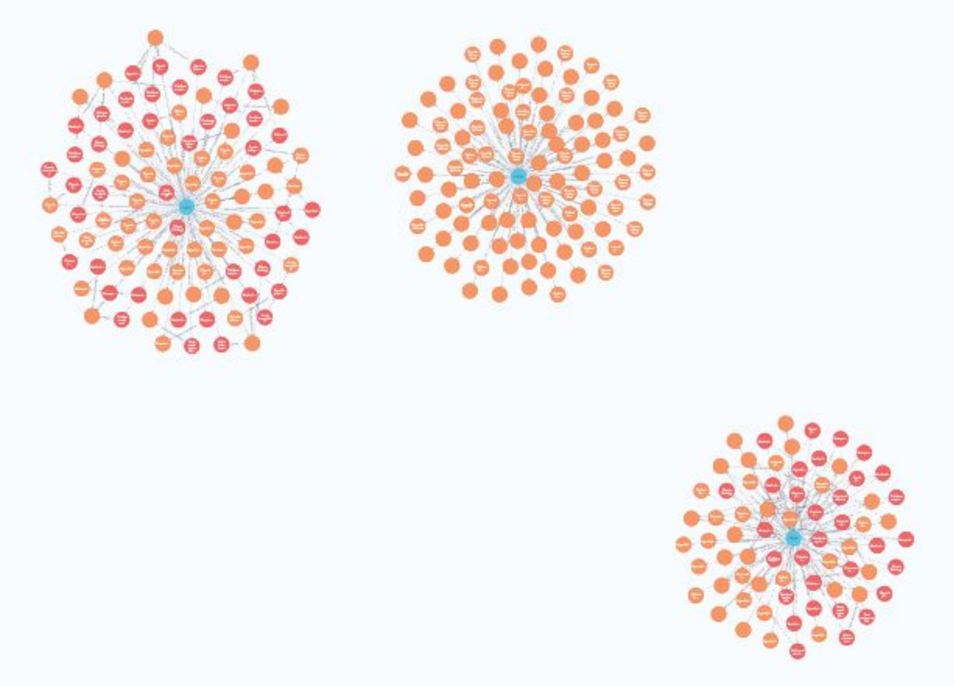
Query: Find the set of providers and total claim cost for each pre-diabetic patient

```
MATCH (p:Patient)-[:HAS_CONDITION]->(c:Condition
{CODE: '714628002'})
WITH p, count(DISTINCT c) as conditionCount
WHERE conditionCount >= 1
MATCH
(p)-[:HAS_ENCOUNTER]->(e:Encounter)-[:PROVIDED_BY]->(
pr:Provider)
WITH p, collect(DISTINCT pr) as providers,
sum(toFloat(e.TOTAL_CLAIM_COST)) as totalCost
RETURN p.FIRST AS PatientFirstName,
       p.LAST AS PatientLastName,
       providers,
       totalCost
ORDER BY totalCost DESC;
```

	PatientFirstName	PatientLastName	providers	totalCost
1	"Elna874"	"Prohaska837"	[(:Provider {ZIP: "021111552", PROCEDURES: "0", STATE: "MA", LON: "-71.0631836", NAME: "Santina680 Dicki44", ORGANIZATION: "0d1570ab-371c-3898-9397-95905d8c5166", CITY: "BOSTON", ADDRESS: "750 WASHINGTON ST", GENDER: "F", Id: "8ba9dc63-e8c2-383a-9031-314602010985", SPECIALITY: "GENERAL PRACTICE", LAT: "42.3499038", ENCOUNTERS: "776"}), (:Provider {ZIP: "021253120", PROCEDURES: "0", STATE: "MA", LON: "-71.04610844302991", NAME: "Magdalena964 Torphy630", ORGANIZATION: "2b97893e-dc50-378b-a266-f089b8450329", CITY: "DORCHESTER", ADDRESS: "250 MOUNT VERNON ST", GENDER: "F", Id: "2d312216-1433-3a77-a29e-f1d766339b2d", SPECIALITY: "GENERAL PRACTICE", LAT: "42.3194571", ENCOUNTERS: "68"}), (:Provider {ZIP: "021272642", PROCEDU	645070.6300000001

Query: Identify patients most likely to have a heart attack

```
MATCH
(p:Patient)-[:HAS_OBSERVATION]->(o:Observation)
WHERE o.DESCRIPTION CONTAINS "Hypertension"
   OR o.DESCRIPTION CONTAINS "High Cholesterol"
   OR o.DESCRIPTION CONTAINS "Obesity"
   OR o.DESCRIPTION CONTAINS "Diabetes"
WITH p, COUNT(o) AS riskFactors
WHERE riskFactors >= 2
RETURN p, riskFactors
ORDER BY riskFactors DESC
```



Conclusions

- Our project combines structured (SQL) and relationship-based (Neo4j) data to enhance healthcare insights.
- Addresses data fragmentation, improves diagnostics, and enhances patient care access.

Future work

- Develop pipelines for JSON data and specifically JSON-based HL7 FHIR (Fast Healthcare Interoperability Resource) to support the latest industry standards
- Extend integration between PostgreSQL, Neo4j, and potentially document databases such as MongoDB

References

- Electronic Health Records | CMS
- Social Determinants of Health - Healthy People 2030 | odphp.health.gov
- Overview - FHIR v5.0.0

Thank you!