

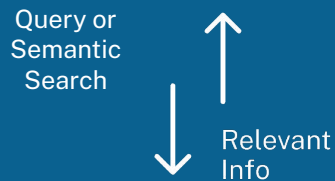
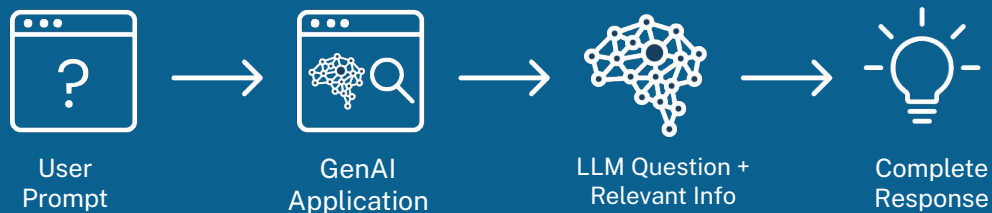


Why Your Agent's Brain Needs a Playbook

Practical Wins from Using Ontologies

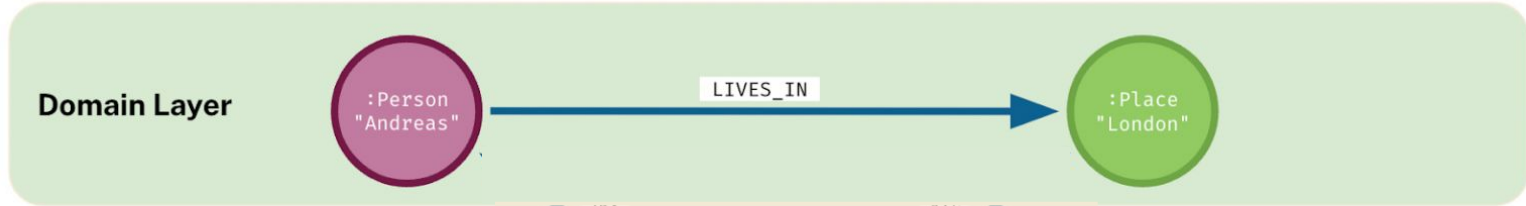
Dr. Jesús Barrasa - AI Field CTO / Neo4j

GraphRAG

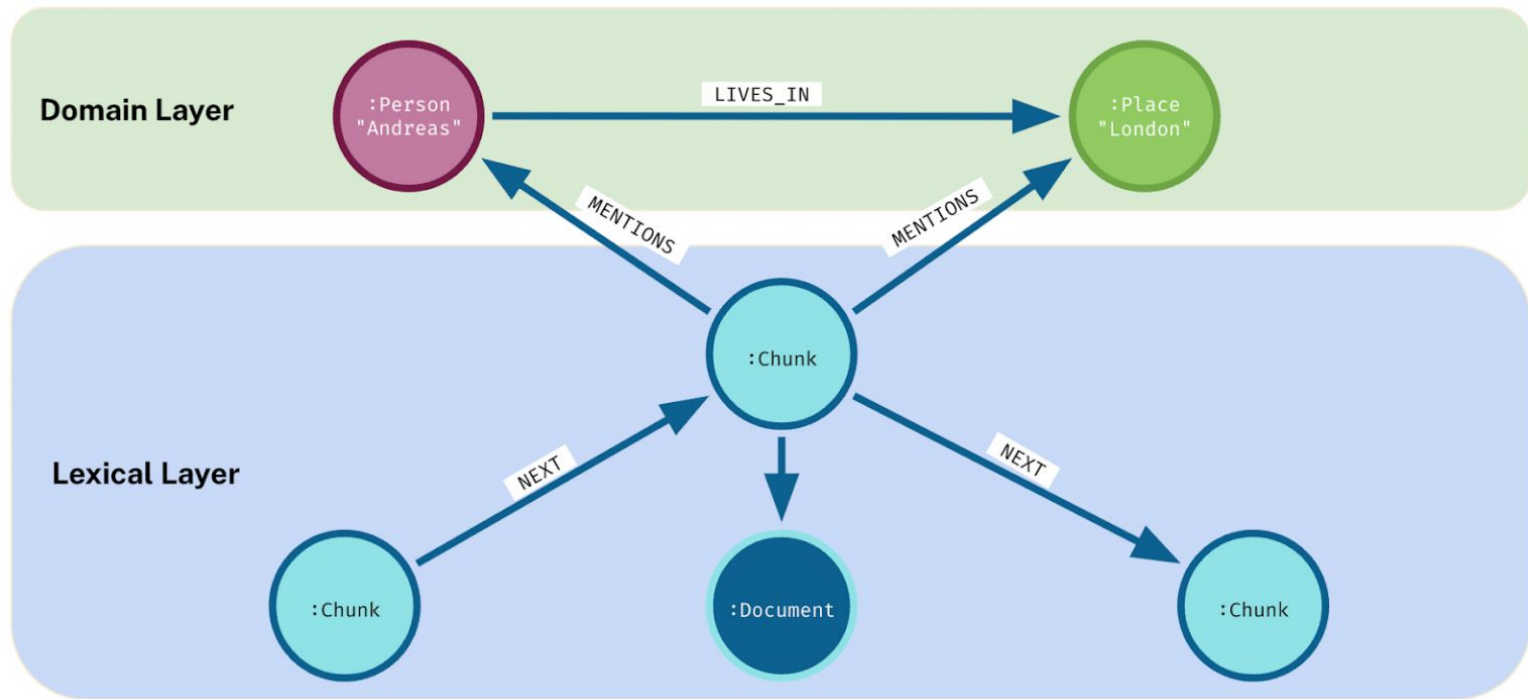


Neo4j's Knowledge Graph

What's in the graph?



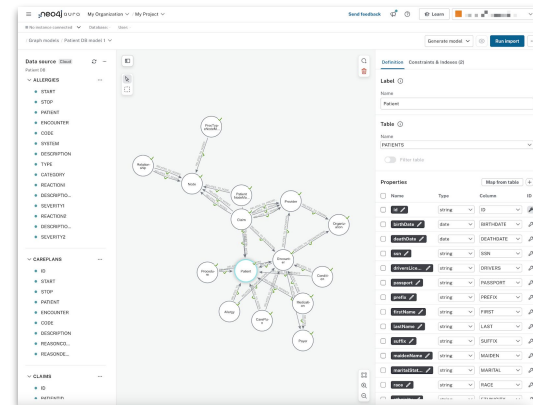
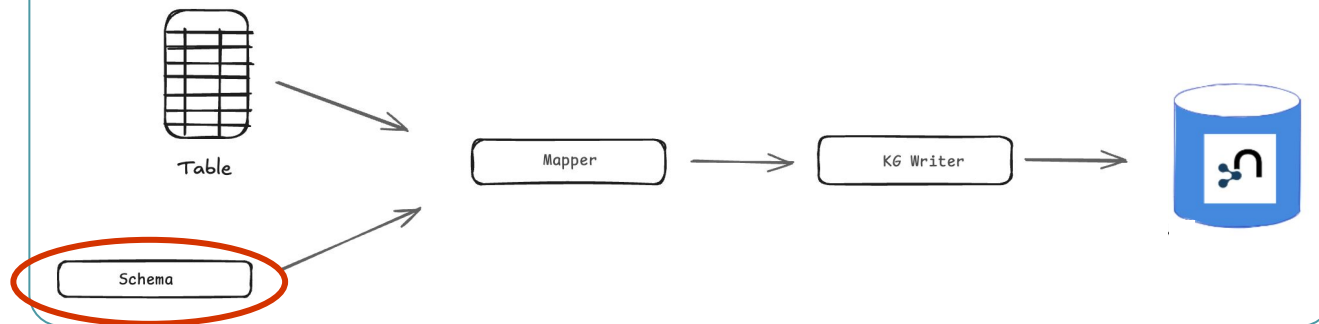
What's in the graph?



No GraphRAG without a graph: KG Construction

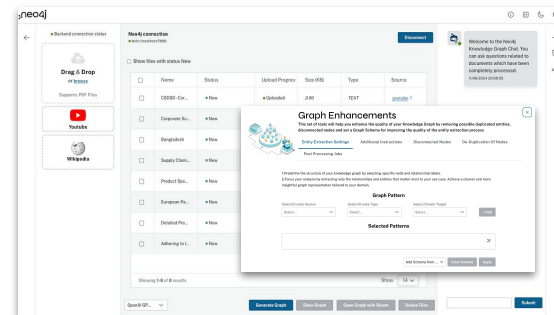
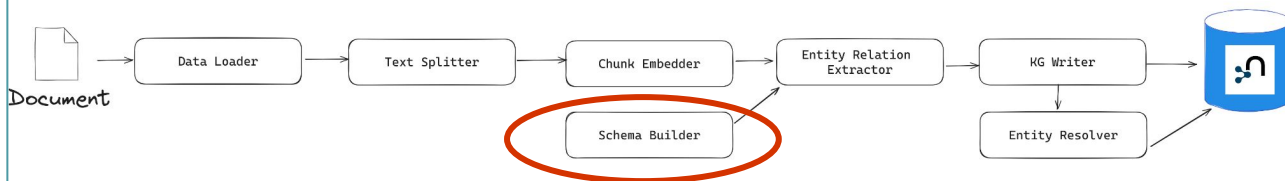


From structured data



<https://console.neo4j.io/>

From unstructured data



<https://llm-graph-builder.neo4j.com>

Ontologies: What? and why?

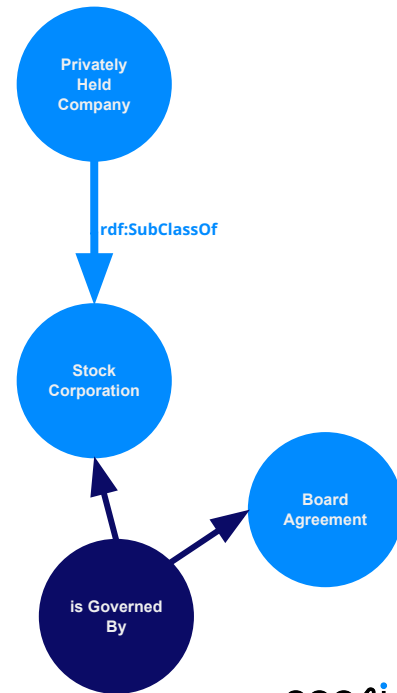
An **ontology** is a shared, explicit, formal domain model

Ontologies: What? and why?

```
<owl:Class rdf:about="&fibo-be-corp-corp;PrivatelyHeldCompany">
  <rdfs:label>privately held company</rdfs:label>
  <fibo-fnd-utl-av:synonym rdf:datatype="&xsd:string">closed corporation</fibo-fnd-utl-av:synonym>
  <fibo-fnd-utl-av:synonym rdf:datatype="&xsd:string">privately held corporation</fibo-fnd-utl-av:synonym>
  <skos:definition rdf:datatype="&xsd:string">A firm whose issued shares are all held by a family or a small
  group of investors and, therefore, cannot be bought by the public.</skos:definition>
  <rdfs:subClassOf rdf:resource="&fibo-be-le-cb;StockCorporation"/>
</owl:Class>
```

```
<owl:Class rdf:about="&fibo-be-le-ch;StockCorporation">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&fibo-fnd-rel-rel;isGovernedBy"/>
      <owl:someValuesFrom rdf:resource="&fibo-be-corp-corp;BoardAgreement"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
```

```
    <owl:Restriction>
      <owl:onProperty rdf:resource="&fibo-fnd-aap-agt;isIdentifiedBy"/>
      <owl:onClass rdf:resource="&fibo-be-corp-corp;RegistrationIdentifier"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&fibo-be-corp-corp;hasDateOfIncorporation"/>
      <owl:onClass rdf:resource="&fibo-fnd-dt-fd;Date"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</rdfs:subClassOf>
```



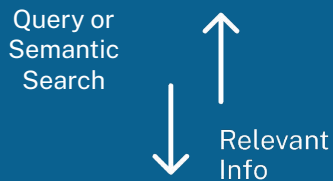
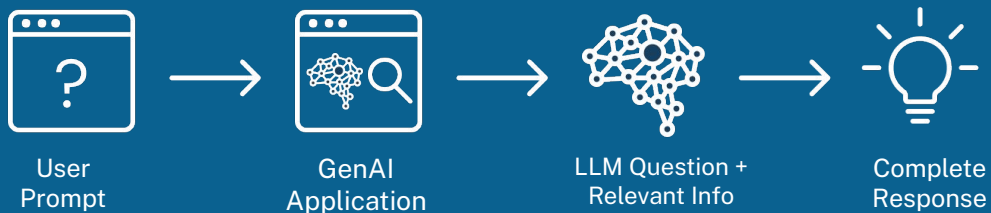
Step by step example



The screenshot displays the Neo4j Developer website. The top navigation bar includes links for Docs, Labs, Get Help, GraphAcademy, and a 'Get Started Free' button. The left sidebar lists various developer guides, with 'GraphRAG Agent for Customer Analytics' highlighted. The main content area features the title 'GraphRAG Agent for Customer & Retail Analytics' and a brief introduction. It lists four steps: 1. Quickly constructing a graph from mixed unstructured and structured data sources, 2. Resolving and linking entities in the graph along the way, 3. Creating diverse graph retrieval tools, and 4. Building an agent with Semantic Kernel. A 'Contents' sidebar on the right lists sections like Prerequisites, Setup, and Schema Generation. A promotional banner for 'NOVES 2025' is also visible. At the bottom, there is a preview of the Neo4j interface showing a graph visualization and a 'Node details' panel.

<https://neo4j.com/developer/genai-ecosystem/customer-graph-agent/>

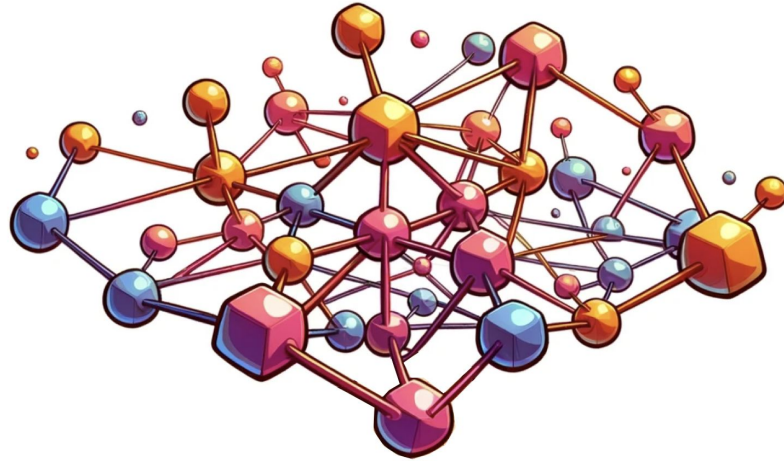
GraphRAG



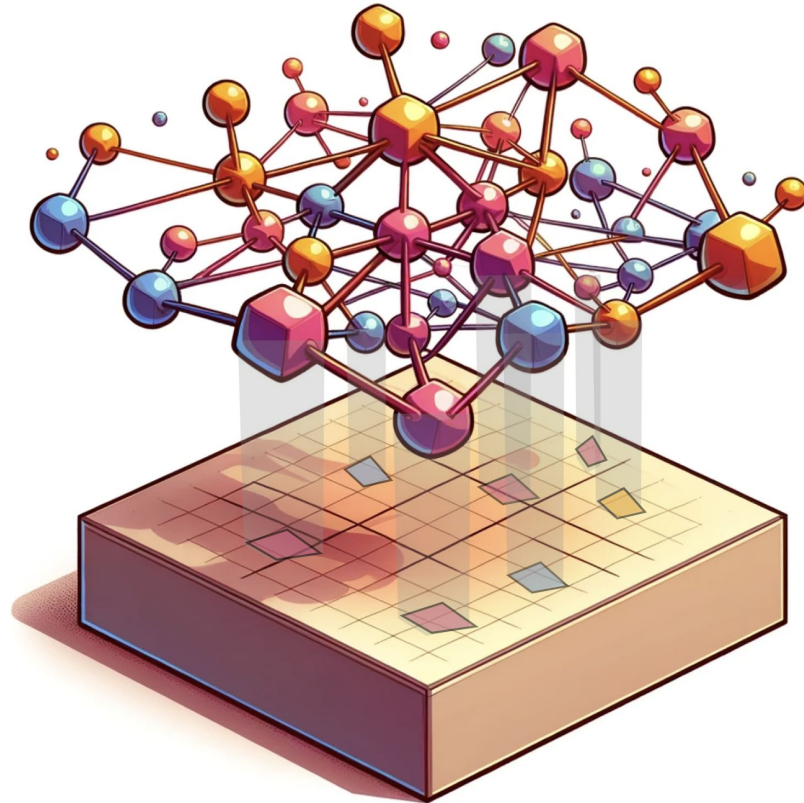
Neo4j's Knowledge Graph

A **Knowledge Graph** captures key enterprise knowledge in the form of **entities and relationships between them**.

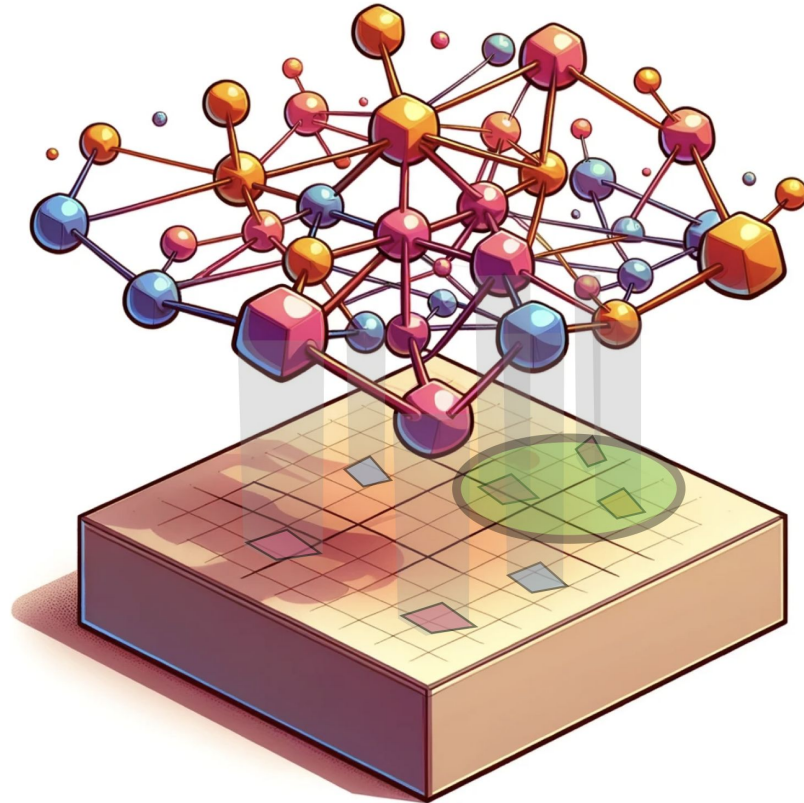
Some nodes in the graph have properties with NL text



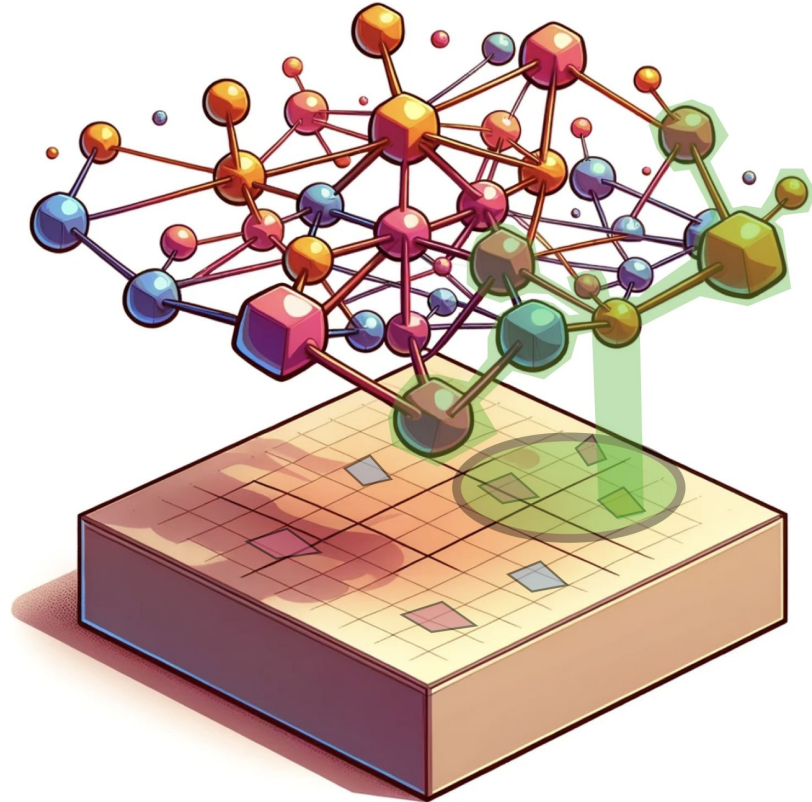
These property values get **embedded** (transformed into a vector representation) and added to a vector index to enable **vector-based semantic search**.



A semantic search on the vector index returns the k **approximate nearest neighbours** to the search concept (word, question, image, etc)



Each result from the vector search is “dereferenced” to get the corresponding node in the graph and a subsequent **graph exploration** finds **semantically related elements** that enrich and augment the final search result.



Retrieval: Vector Search + Contextualisation

```
retrieval_query = """
RETURN node.title as movieTitle,
       node.plot as moviePlot,
       collect { MATCH (actor:Actor)-[:ACTED_IN]->(node) RETURN actor.name } AS actors
"""

retriever = VectorCypherRetriever(
    driver,
    Index_name = "movieplots",
    embedder = OpenAIEmbeddings(model="text-embedding-3-large"),
    retrieval_query=retrieval_query,
)

query_text = "A computer hacker discovers that the world he knows is a simulated reality
controlled by machines"

retriever_result = retriever.search(query_text=query_text, top_k=3)
```



Retrieval... powerful 💪 but rigid 😞

```
retrieval_query = """
RETURN node.title as movieTitle,
        node.plot as moviePlot,
        collect { MATCH (actor:Actor)-[:ACTED_IN]->(node) RETURN actor.name } AS actors
"""
```

```
retriever = VectorCypherRetriever(
    driver,
    Index_name = "movieplots",
    embedder = OpenAIEmbeddings(model="text-embedding-3-large"),
    retrieval_query=retrieval_query,
)
```

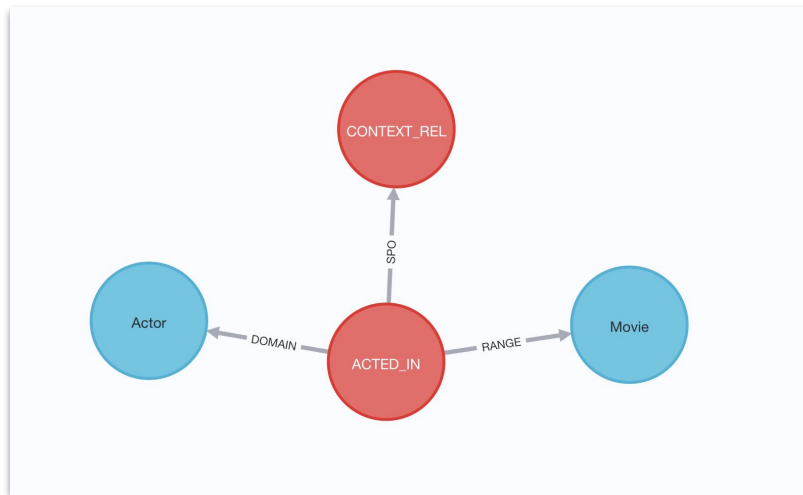
```
query_text = "A computer hacker discovers that the world he knows is a simulated reality  
controlled by machines"
```

```
retriever_result = retriever.search(query_text=query_text, top_k=3)
```



Retrieval... dynamic

```
retrieval_query = """  
MATCH (r)-[:SPO]->(:Relationship { name: "CONTEXT_REL"})  
MATCH (node)-[:$(r.name)]->(related)  
RETURN properties(related) as context  
"""
```



Ontology



Data





Thank you!