sitBCN: Knowledge Graphs with SAP Data

SULZER ESPAÑA

Barcelona, 22.11.2024 – Antonio Leites

sitBCN: Knowledge Graphs with SAP Data

Agenda

- Introduction to Knowledge Graphs
- Building a Knowledge Graph
- SAP and Knowledge Graphs

sitBCN: Knowledge Graphs with SAP Data

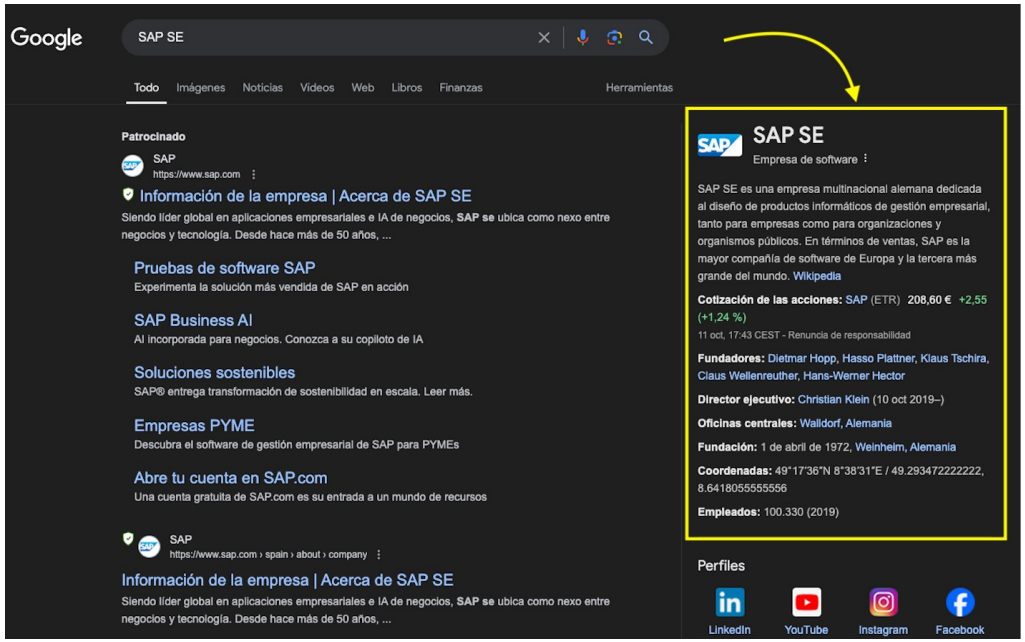
SAP Inside Track BARCELONA

Agenda

- **Introduction to Knowledge Graphs**
- Building a Knowledge Graph
- SAP and Knowledge Graphs

Knowlegde Graphs
Web Search Experience

- The term “**knowledge graph**” was introduced in 2012 by Google
- Today, knowledge graphs are a common part of our daily Web search experience.
- Search results not only include a list of Web pages with the string “SAP SA”, but also provide a **structured** and **interconnected** **network of information** about the *entity* “SAP SA”.
- Key aspect of **knowledge graphs**: they encode “**things, not strings**”, that is: **entities**, their ***properties*** and ***relations*** to **other entities**.



In the knowledge graph, you will see properties of the entity “SAP SE”, for example its description or a link to the SAP website. You will also see relations to other entities in the knowledge graph, for example headquarters of SAP. Clicking e.g. on “Walldorf” will lead you to the knowledge graph entry for Walldorf, including other types of properties and relations, like the population “14,646” and the “region” relation to the entity “Karlsruhe”.

Knowledge Representation with Graphs

FROM DATA TO KNOWLEDGE

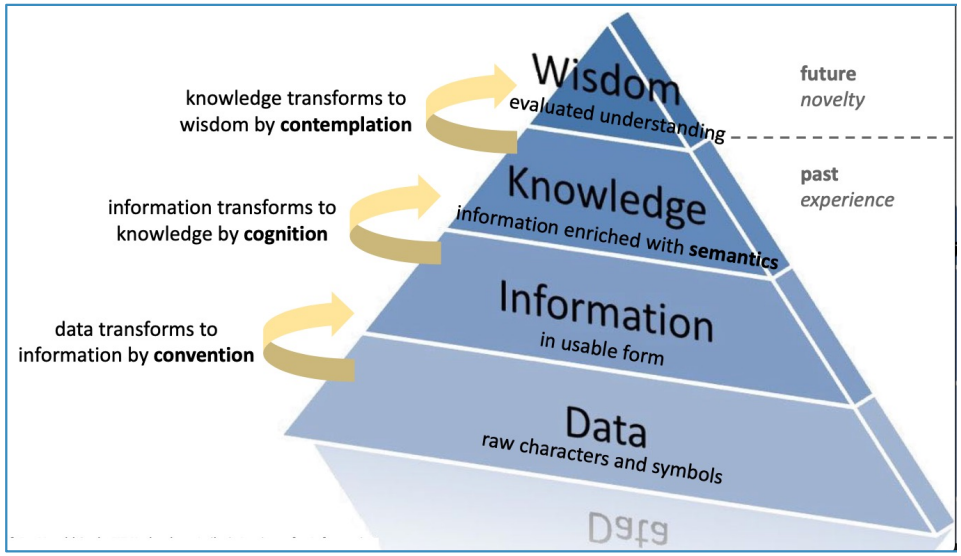
Sulzer

Data:

- Data is raw.
- It simply exists and has no significance beyond its existence (in and of itself).
- it can exist in any form, usable or not.

Information:

- Information is data that has been given meaning by way of relational connection.
- This "meaning" can be useful, but does not have to be.
- Information is contained in descriptions.
- Information answers to questions that begin with such words as **who**, **what**, **when**, **where**, and **how many**.



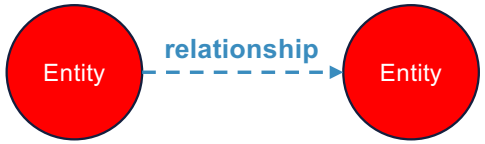
DIKW Pyramid, Ackoff 1989

Knowledge:

- **Knowledge** is the appropriate collection of information, such that its intent is to be useful.
- **Wisdom** is the ability to make sound judgments and decisions.
- **Understanding** is a continuum that leads from **data**, through **information** and **knowledge**, and ultimately to **wisdom**.

SAP Inside Track BARCELONA

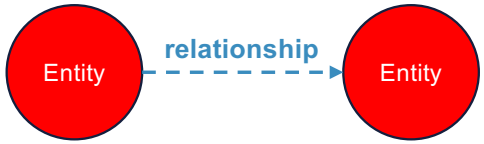
Knowledge Graphs
WHAT IS A KNOWLEDGE GRAPH?



- ✓ **Relationships**: Interconnections between entities. (informations)

Knowledge Graphs
WHAT IS A KNOWLEDGE GRAPH?

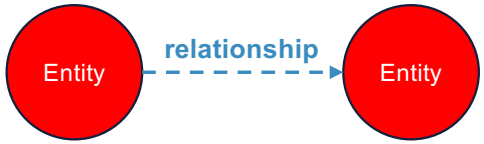
Sulzer

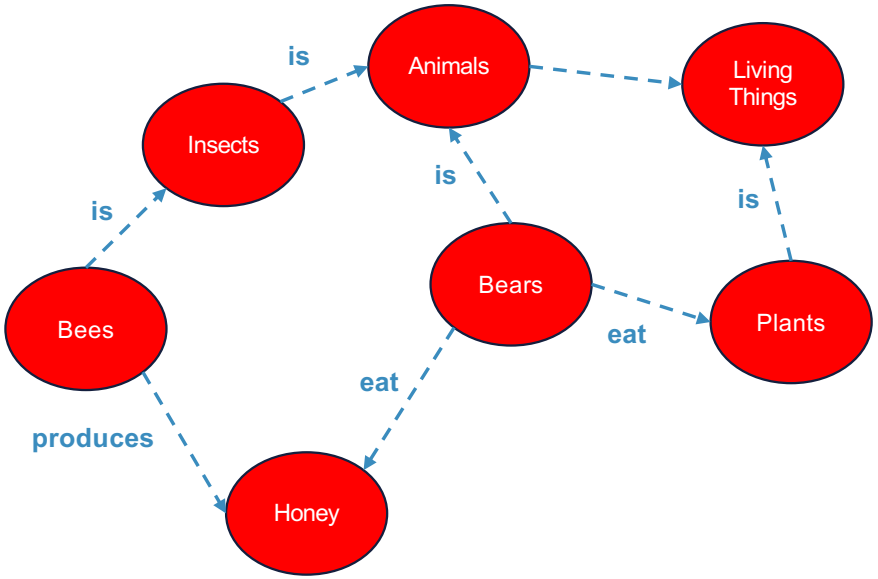


- ✓ **Relationships**: Interconnections between entities. (informations)

Knowledge Graphs

WHAT IS A KNOWLEDGE GRAPH?

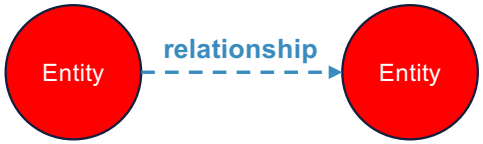


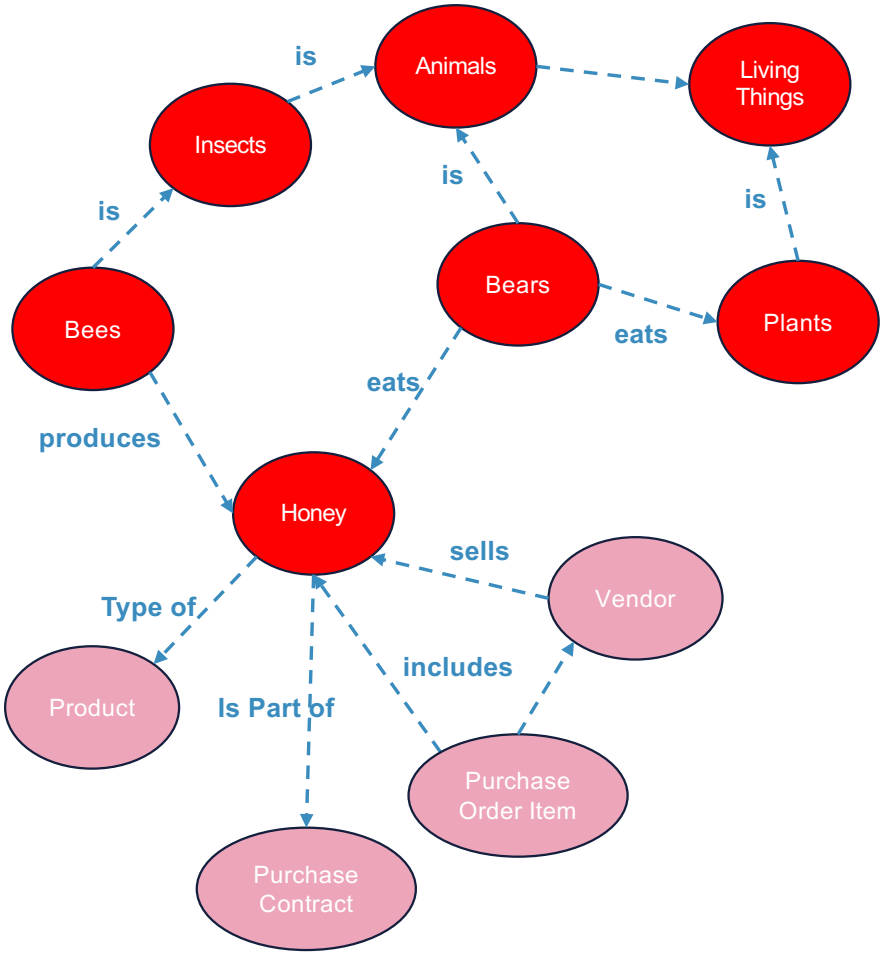


- ✓ **Relationships**: Interconnections between entities. (informations)
- ✓ **Knolewdge**: The whole relationships among entities

Knowledge Graphs

WHAT IS A KNOWLEDGE GRAPH?

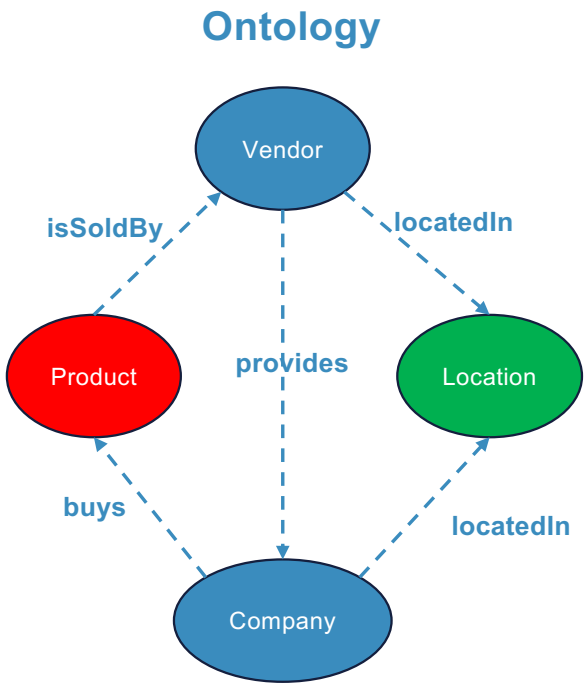




- ✓ **Relationships**: Interconnections between entities. (information)
- ✓ **Knowledge**: The whole relationships between entities

Knowledge Graphs: Ontology

How Knowledge Graphs can be Modelled?

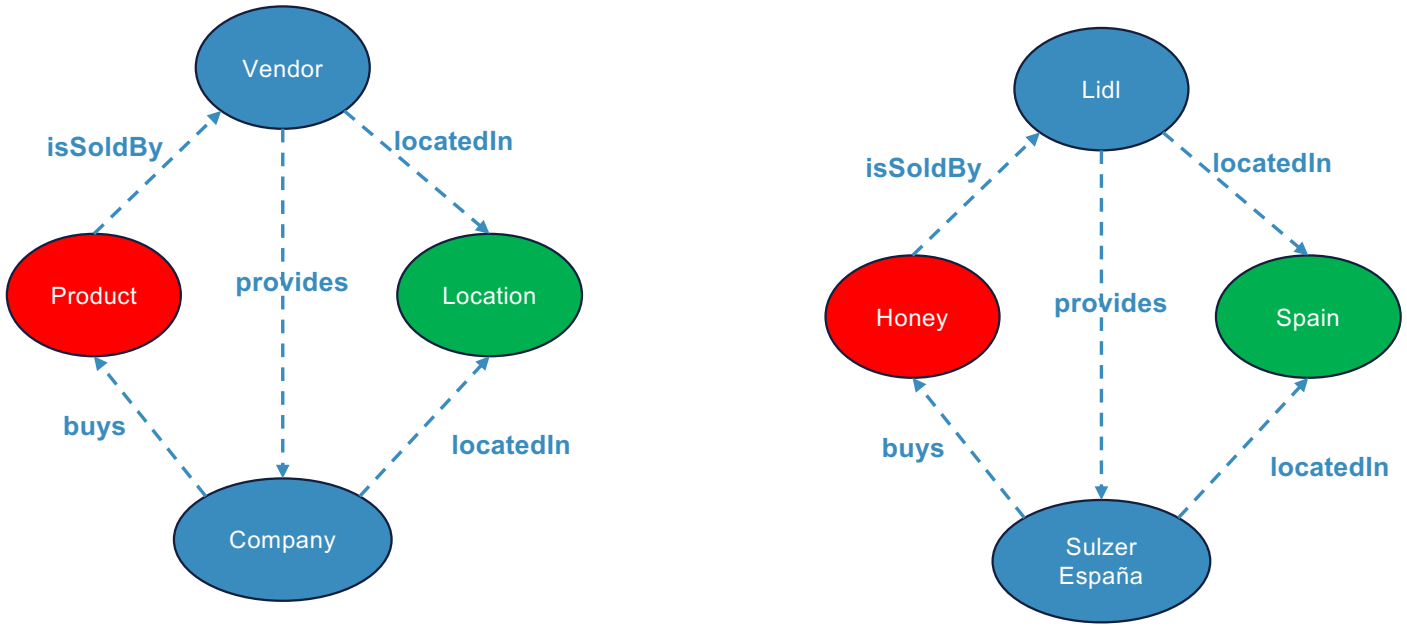


Knowledge Graphs: Ontology

How Knowledge Graphs can be Modelled?

Sulzer

Ontology **+ Data** **= Knowledge Graph**



Triplets: Semantic Facts

<Subject, Predicate, Object>

<Honey , isSoldBy, Lidl>.
<Sulzer España , buys, Honey>.
<Lidl , provides, Sulzer España>.
<Lidl , locatedIn, Spain>.
<Sulzer España , locatedIn, Spain>.

A **knowledge graph** is a **structured representation of data** (adhering to *ontology rules*)

SAP Inside Track BARCELONA

Knowledge Graphs

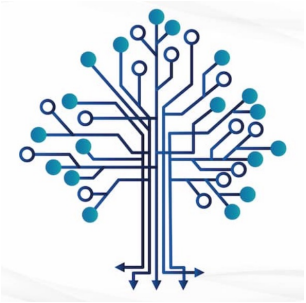
A Different Approach for Data Integration

Sulzer

- **Classic Approach**:
  - ✓ Use of data warehouses with one unified schema.
- **Knowledge Graphs:**
  - ✓ there is no common data model across the data sources.
  - ✓ For various types of entities like companies, persons or cities, the knowledge graph models are created in a data and use case driven manner.
  - ✓ "All models are wrong, but some are useful", what is said about models in statistics can also be said about modelling in knowledge graphs. There is no perfect knowledge graph model of an entity like "person" or "company". There are only models that more or less fit to given use cases.

SAP Inside Track BARCELONA

Knowledge Graphs
WHY USE KNOWLEDGE GRAPHS?



Sulzer

- **Complex Relationships**:
  - ✓ Capture rich semantic connections that are difficult to express in traditional databases.
- **Flexibility and Agility:**
  - ✓ Easily adapt to evolving data and changing business requirements.
  - ✓ Add new entities, relationships and properties without modifying the underlying schema.
- **Efficient Querying:**
  - ✓ Optimized queries involving multiple relationships and entities.

Used in
- Google Websearch
- Amazon Product Search
- Siri
- Wikidata

SAP Inside Track BARCELONA

Sulzer

SAP Inside Track BARCELONA

Agenda

- Introduction to Knowledge Graphs
- **Building a Knowledge Graph**
- SAP and Knowledge Graphs

Knowlegde Graphs on linked business data

KNOWLEDGE GRAPH GENERATION PROCESS

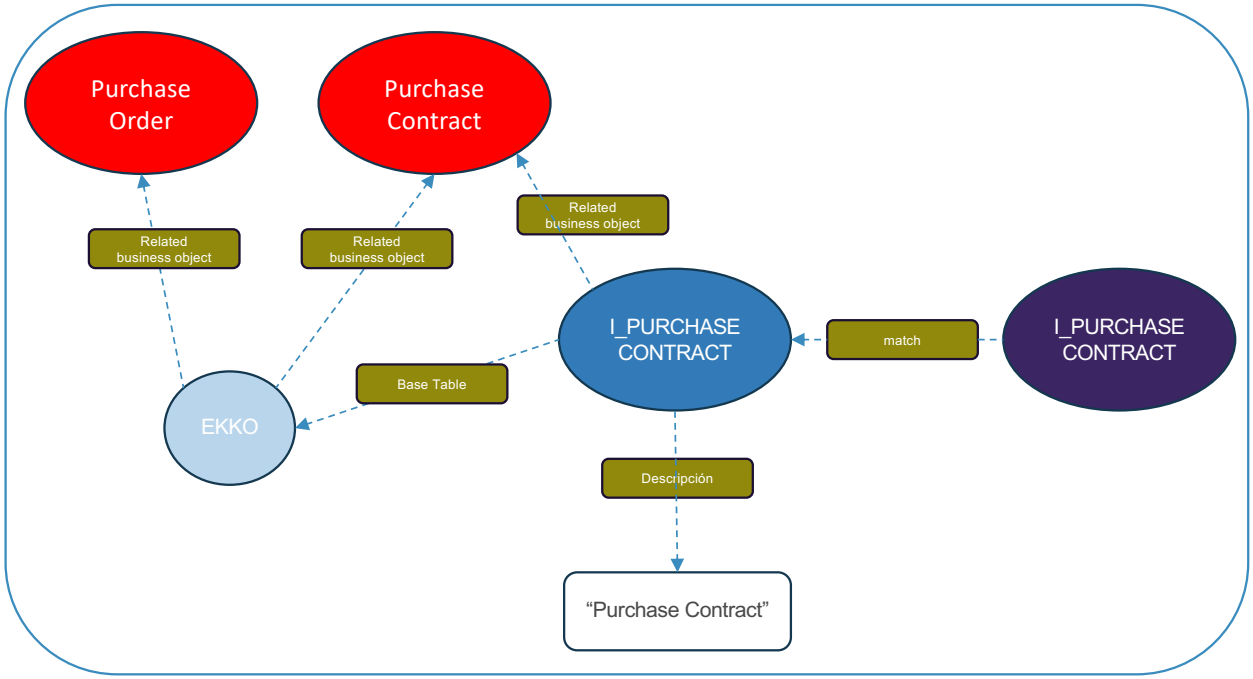
Sulzer

- **Data Extraction**
  - ✓ **Extract metadata** (for ABAP tables, business objects, CDS views etc.) from various SAP internal and external data sources
- **Modelling**
  - ✓ Define **Target Model for the Use Case**. The model captures only types of entities and their required properties for the Use Case **using RDF**
- **Knowledge Graph Generation**
  - ✓ Transform extracted data to the target model and upload the data to a Knowledge Graph DB
- **Knowledge Graph Provision**
  - ✓ Provide the graph to applications via APIs.

SAP Inside Track BARCELONA

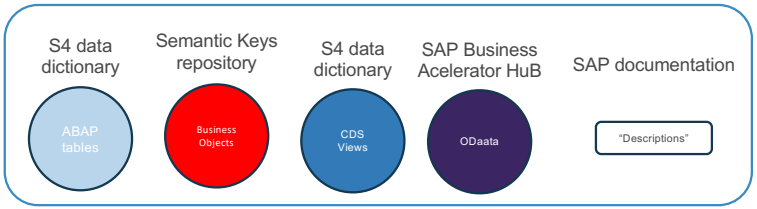
Knowlegde Graphs on linked business data

EXAMPLE KNOWLEDGE GRAPH IMPLEMENTING AND ENABLING SEMANTIC ACCESS TO THE TABLE “EKKO”.



A knowledge graph contains entities, properties and relations.

Underlying metadata sources from different systems



The business objects, like the entities and relations have global and unique identifier, which enable the flexible integration of new (meta) data sources.

Key value of a knowledge graph: it stores entities in the graph.
The “**related business object**” relation, constitutes an extensible semantic access layer across the domain specific metadata items.

Knowlegde Graphs on linked business data

RDF AS THE TECHNICAL BASIS FOR KNOWLEDGE GRAPHS

Sulzer

Technical basis for generating, storing and querying knowledge graphs is the graph data model **RDF** (Resource Description Framework), a set of standards developed by the W3C

RDF Triple

```
1   # Prefixes for classes
2   @prefix abaptable: <http://schema.sap.com/ABAPTable/> .
3   @prefix cdsview: <http://schema.sap.com/CDSView/>.
4   @prefix bo: <http://schema.sap.com/BusinessObject/> .
5   @prefix entitySet: <http://schema.sap.com/ODATAEntitySet/>.
6
7   # Prefixes for properties
8   @prefix s4: <http://schema.sap.com/ns/s4#> .
9   @prefix cds: <http://schema.sap.com/ns/cds#>.
10
11  # Prefixes for general RDF vocabularies, here only SKOS
12  @prefix skos: <http://www.w3.org/2004/02/skos/core#> .
13
14  abaptable:EKKO s4:relatedBusinessObject bo:PurchaseOrder .
15  abaptable:EKKO s4:relatedBusinessObject bo:PurchaseContract .
16
17  cdsview:I_PURCHASECONTRACT s4:relatedBusinessObject bo:PurchaseContract.
18
19  cdsview:I_PURCHASECONTRACT s4:baseTable abaptable:EKKO.
20
21  cdsview:I_PURCHASECONTRACT cds:description "Purchase Contract".
22
23  entitySet:I_PURCHASECONTRACT skos:exactMatch cdsview:I_PURCHASECONTRACT.
```

Triples

- The actual RDF data in the example starts after the prefix definition in line 14
  - **Subject** (in line 14) **abatable:EKKO**
  - **Predicate** (in line 14) **s4:relatedBusinessObject**
  - **Object** (in line 14) **bo:PurchaseOrder**

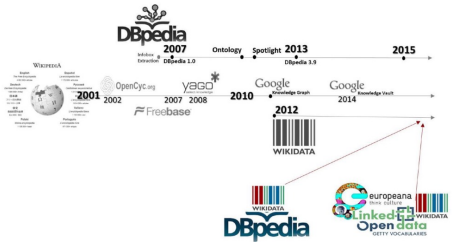
RDF Vocabularies

- **Vocabularies** are a key means to realize the knowledge graph capability of distinguishing “things from strings”.
- In the example, we have two different vocabularies: **cdsview:** and **entitySet:** and we can distinguish two different entities which have the same name: **I_PURCHASECONTRACT**
- RDF also allows to create formal models for vocabularies via technologies like OWL or SHACL.)
- Based on use case needs, we can create and extend our use our own, SAP specific vocabularies and combine them with public vocabularies like SKOS

SAP Inside Track BARCELONA

Knowlegde Graphs on linked business data

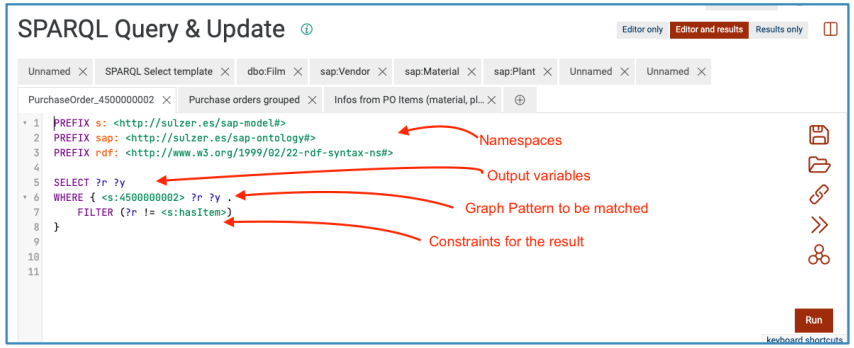
SPARQL RDF Query Language for Knowledge Graphs

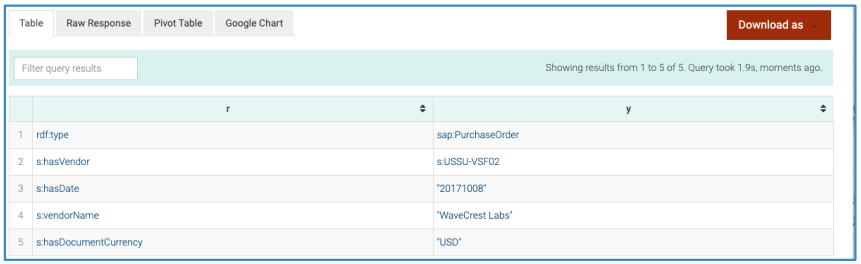


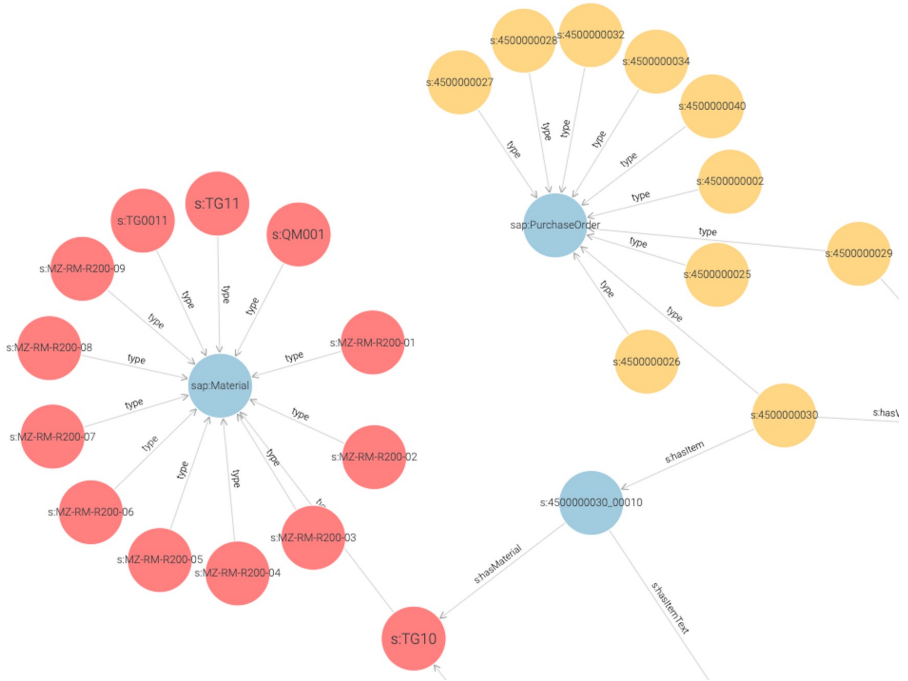
SPARQL is based on RDF Turtle serialization and basic graph pattern matching.

- A Graph Pattern (Triple pattern) is a RDF Triple that contains variables at any arbitrary place (Subject, Property, Object)

Graph Pattern = Turtle + Variables



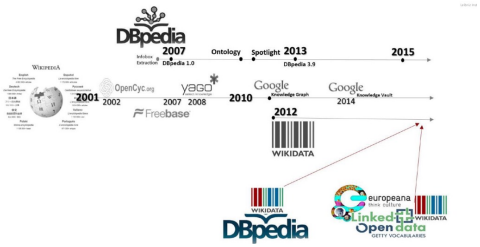




SAP Inside Track BARCELONA

Knowlegde Graphs on linked business data

SPARQL RDF Query Language for Knowledge Graphs



SPARQL is based on RDF Turtle serialization and basic graph pattern matching.

- A Graph Pattern (Triple pattern) is a RDF Triple that contains variables at any arbitrary place (Subject, Property, Object)

Graph Pattern = Turtle + Variables

SPARQL Query & Update

Editor only | Editor and results | Results only

Unnamed | SPARQL Select template | dbo:Film | sap:Vendor | sap:Material | sap:Plant | Unnamed | Unnamed

Purchase orders grouped | Infos from PO Items (material, pl... | sap:PurchaseOrder | sap:PurchaseOrder | sap:Plant

```
PREFIX s: <http://sulzer.es/sap-model#>
PREFIX sap: <http://sulzer.es/sap-ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?deliveryCity ?vendorCity ?vendorName ?purchaseOrder ?material ?materialDescription ?quantity ?unit
WHERE {
  ?purchaseOrder a <sap:PurchaseOrder> .
  ?purchaseOrder <s:hasItem> ?poItem .
  ?poItem <s:receivingPlant> ?plant .
  ?plant <s:hasAddress> ?address .
  ?address <s:city> ?deliveryCity .

  # Link to the vendor and retrieve the vendor's city
  ?purchaseOrder <s:hasVendor> ?vendor .
  ?vendor <s:city> ?vendorCity .
  ?vendor <s:vendorName> ?vendorName .

  # Link to the material details
  ?poItem <s:hasMaterial> ?material .
  ?material <s:hasDescription> ?materialDescription .

  # Retrieve the quantity and unit of measure for the item
  ?poItem <s:hasQuantity> ?quantity .
  ?material <s:hasBaseUnit> ?unit .

}

GROUP BY ?purchaseOrder ?deliveryCity ?vendorName ?vendorCity ?material ?materialDescription ?quantity ?unit
ORDER BY ?deliveryCity ?vendorName
```

Run

	deliveryCity	vendorCity	vendorName	purchaseOrder	material	materialDescription	quantity	unit
1	"Palo Alto"	"Blacksburg"	"Domestic US Subcontractor A"	s:4500000032	s:TG11	"Trad.Good 11,PD,Reg.Trading"	"3.000"	"ST"
2	"Palo Alto"	"Muncie"	"Domestic US Supplier 1"	s:4500000030	s:TG10	"Trad.Good 10,PD,Third Party"	"3.000"	"ST"
3	"Palo Alto"	"Bismarck"	"Domestic US Supplier 2"	s:4500000028	s:TG0011	"Trading Good 0011,PD,Regular Proc."	"3.000"	"ST"
4	"Palo Alto"	"Bismarck"	"Domestic US Supplier 2"	s:4500000034	s:TG10	"Trad.Good 10,PD,Third Party"	"3.000"	"ST"
5	"Palo Alto"	"Wichita"	"Domestic US Supplier 6 (Returns)"	s:4500000027	s:QM001	"QM Regular"	"3.000"	"ST"
6	"Palo Alto"	"Wichita"	"Domestic US Supplier 6 (Returns)"	s:4500000040	s:QM001	"QM Regular"	"3.000"	"ST"
7	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-06	"BKR-200 Brakes"	"3.000"	"ST"
8	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-08	"BKR-200 Pedal Kit"	"3.000"	"ST"
9	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-07	"BKR-200 Derailleur Gears"	"3.000"	"ST"
10	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-02	"BKR-200 Handle Bars"	"3.000"	"ST"
11	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-03	"BKR-200 Seat"	"3.000"	"ST"
12	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-09	"BKR-200 Drive Train"	"3.000"	"ST"
13	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-01	"BKR-200 Frame"	"3.000"	"ST"
14	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-04	"BKR-200 Wheels"	"3.000"	"ST"
15	"Palo Alto"	"Norwalk"	"WaveCrest Labs"	s:4500000002	s:MZ-RM-R200-05	"BKR-200 Forks"	"3.000"	"ST"

Knowlegde Graphs: simplified example

ABAP REPORT EXTRACTING RDF-TRIPLES

Sulzer

Z_CREATE_RDF_TRIPLE

```
*&---------------------------------------------------------------------*
*& Report z_create_rdf_triple
*&---------------------------------------------------------------------*
REPORT z_create_rdf_triple line-size 180.

DATA: lt_purchase_order      TYPE TABLE OF i_purchaseorder,
      lt_purchase_order_item TYPE TABLE OF i_purchaseorderitem,
      lt_material            TYPE TABLE OF i_material,
      ls_material_text       TYPE i_materialtext,
      lt_supplier            TYPE TABLE OF i_supplier,
      ls_supplier            TYPE i_supplier,
      lv_subject             TYPE string,
      lv_predicate           TYPE string,
      lv_object              TYPE string,
      lv_triple              TYPE string,
      lv_filename            TYPE string VALUE '/tmp/rdf_triples.ttl',
      lv_count               TYPE i VALUE 0,
      lv_quantity            TYPE string,
      lv_netamout            type string.

DATA: lt_summary TYPE TABLE OF string,
      lv_summary TYPE string.

" Open the dataset (file) for writing
OPEN DATASET lv_filename FOR OUTPUT IN TEXT MODE ENCODING DEFAULT.

" Write the prefixes at the beginning of the file
TRANSFER '@prefix s: <http://sulzer.es/sap-model#> .' TO lv_filename.              "Instances
TRANSFER '@prefix sap: <http://sulzer.es/sap-ontology#> .' TO lv_filename.          "Ontology
TRANSFER '@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .' TO lv_filename.
TRANSFER '@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .' TO lv_filename.
TRANSFER '@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .' TO lv_filename.
TRANSFER '@prefix owl: <http://www.w3.org/2002/07/owl#> .' TO lv_filename.
TRANSFER '@prefix ex:  <http://example.org/sap-model#> .' TO lv_filename.

TRANSFER '' TO lv_filename. " Blank line for readability

" Define RDF Classes and subclass relationships
TRANSFER 'sap:Document rdf:type owl:Class .' TO lv_filename.
TRANSFER 'sap:PurchaseDocument rdf:type owl:Class ; rdfs:subClassOf sap:Document .' TO lv_filename.
TRANSFER 'sap:MasterData rdf:type owl:Class .' TO lv_filename.
TRANSFER 'sap:Vendor rdf:type owl:Class ; rdfs:subClassOf sap:MasterData .' TO lv_filename.
TRANSFER 'sap:Material rdf:type owl:Class ; rdfs:subClassOf sap:MasterData .' TO lv_filename.
TRANSFER 'sap:PurchaseOrder rdf:type owl:Class ; rdfs:subClassOf sap:PurchaseDocument .' TO lv_filename.
TRANSFER 'sap:PurchaseOrderItem rdf:type owl:Class ; rdfs:label "Purchase Order Item" ; rdfs:subClassOf sap:PurchaseOrder . ' TO lv_filename.

TRANSFER '' TO lv_filename. " Blank line for readability

" Define RDF Properties
TRANSFER 's:hasVendor a rdf:Property ; rdfs:label "hasVendor" .' TO lv_filename.
TRANSFER 's:hasVendor rdfs:domain sap:PurchaseOrder ; rdfs:range sap:Vendor .' TO lv_filename.
TRANSFER 's:hasDate a rdf:Property ;rdfs:label "has Date" ; rdfs:comment "Represents the date associated with a PurchaseOrder".' TO lv_filename.
TRANSFER 's:hasDate rdfs:domain sap:PurchaseOrder ; rdfs:range xsd:date.' TO lv_filename.
TRANSFER 's:vendorName a rdf:Property ; rdfs:label "vendor name" ;rdfs:comment "Represents the name of the Vendor associated with a PurchaseOrder or Vendor entity".' TO lv_filename.
TRANSFER 's:vendorName rdfs:domain sap:Vendor ; rdfs:range xsd:string .' TO lv_filename.
TRANSFER 's:country a rdf:Property ; rdfs:label "country" ; rdfs:comment "Represents the country associated with a Vendor entity".' TO lv_filename.
TRANSFER 's:country rdfs:domain sap:Vendor ; rdfs:range xsd:string .' TO lv_filename.
TRANSFER 's:city a rdf:Property ; rdfs:label "city" ; rdfs:comment "Represents the country associated with a Vendor entity".' TO lv_filename.
TRANSFER 's:city rdfs:domain sap:Vendor ; rdfs:range xsd:string .' TO lv_filename.
TRANSFER 's:hasItem a rdf:Property ;rdfs:label "hasItem" ;rdfs:comment "Associates a Purchase Order with one of its items".' TO lv_filename.
TRANSFER 's:hasItem rdfs:domain sap:PurchaseOrder ; rdfs:range rdfs:Resource ; owl:propertyChainAxiom ( sap:PurchaseOrder sap:hasItem ) .' TO lv_filename.
TRANSFER 's:hasMaterial a rdf:Property ; rdfs:label "hasMaterial" .' TO lv_filename.
TRANSFER 's:hasMaterial rdfs:domain sap:PurchaseOrderItem ; rdfs:range sap:Material .' TO lv_filename.
TRANSFER 's:hasItemText a rdf:Property ; rdfs:label "Item Text" ; rdfs:comment "Text associated to the PO Item".' TO lv_filename.
TRANSFER 's:hasItemText rdfs:domain sap:PurchaseOrderItem ; rdfs:range xsd:string .' TO lv_filename.
TRANSFER 's:hasNetAmount a rdf:Property ;rdfs:label "hasNetAmount" ; rdfs:comment "Represents the net amount for a Purchase Order Item" .' TO lv_filename.
TRANSFER 's:hasNetAmount rdfs:domain sap:PurchaseOrderItem ; rdfs:range xsd:decimal .' TO lv_filename.
TRANSFER 's:hasCurrency a rdf:Property ; rdfs:label "hasCurrency" ; rdfs:comment "Represents the currency of the net amount for a Purchase Order Item" .' TO lv_filename.
TRANSFER 's:hasCurrency rdfs:domain sap:PurchaseOrderItem ; rdfs:range xsd:string .' TO lv_filename.
TRANSFER 's:hasDescription a rdf:Property ; rdfs:label "hasDescription" ; rdfs:comment "Represents a textual description or name for a material" .' TO lv_filename.
TRANSFER 's:hasDescription rdfs:domain sap:Material ; rdfs:range xsd:string .' TO lv_filename.
TRANSFER 's:hasBaseUnit a rdf:Property ; rdfs:label "hasBaseUnit" ; rdfs:comment "Represents the base unit of measure for a material" .' TO lv_filename.
TRANSFER 's:hasBaseUnit rdfs:domain sap:Material ; rdfs:range xsd:string .' TO lv_filename.
TRANSFER 's:hasQuantity a rdf:Property ; rdfs:label "hasQuantity" ; rdfs:comment "Represents the quantity of an item" .' TO lv_filename.
TRANSFER 's:hasQuantity rdfs:domain sap:PurchaseOrderItem ; rdfs:range xsd:decimal . ' TO lv_filename.

TRANSFER '' TO lv_filename. " Blank line for readability
```

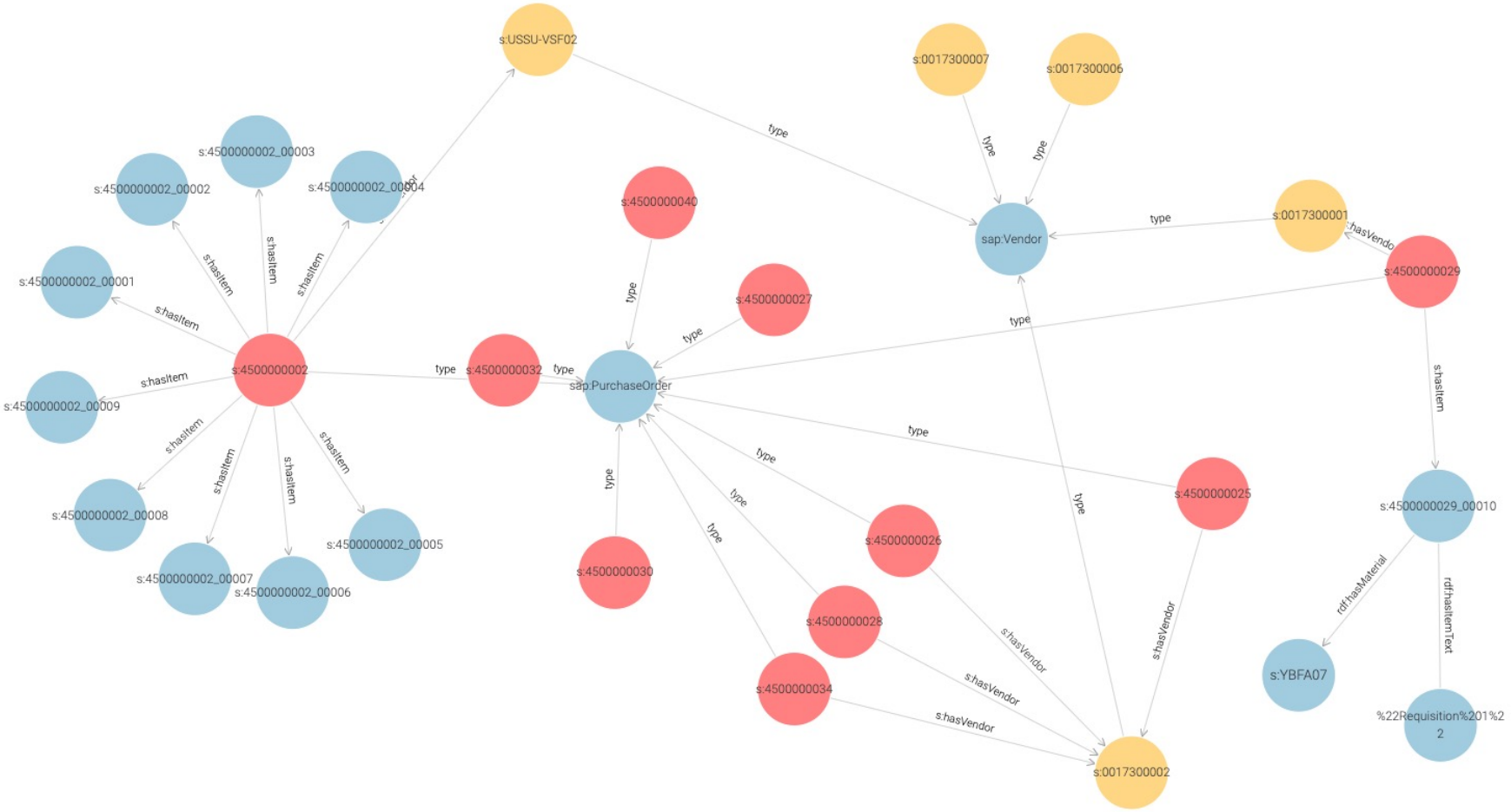
Z_CREATE_RDF_TRIPLE

```
" Select first 10 Purchase Orders from I_PURCHASEORDER
SELECT * FROM i_purchaseorder INTO TABLE @lt_purchase_order UP TO 10 ROWS.

" Fetch vendor details using the I_Supplier CDS view
SELECT * FROM i_supplier INTO TABLE @lt_supplier
  FOR ALL ENTRIES IN @lt_purchase_order
  WHERE supplier = @lt_purchase_order-supplier.

" Loop through Purchase Orders to create RDF triples
LOOP AT lt_purchase_order INTO DATA(ls_purchase_order).

  " Triple to define the PurchaseOrder as an instance of PurchaseOrder
  lv_subject = |s:{ ls_purchase_order-purchaseorder }|.
  lv_predicate = 'rdf:type'.
  lv_object = 'sap:PurchaseOrder'.
  lv_triple = |<{ lv_subject }>{ lv_predicate }<{ lv_object }> .|.
  TRANSFER lv_triple TO lv_filename.
  ADD 1 TO lv_count.

  " Triple for PurchaseOrder
  lv_subject = |s:{ ls_purchase_order-purchaseorder }|.
  lv_predicate = 's:hasDate'.
  lv_object = |"{ ls_purchase_order-PurchaseOrderDate }"|.
  lv_triple = |<{ lv_subject }>{ lv_predicate }>{ lv_object } .|.
  TRANSFER lv_triple TO lv_filename.
  ADD 1 TO lv_count.

  " Triple for Vendor
  lv_predicate = 's:hasVendor'.
  lv_object = |s:{ ls_purchase_order-supplier }|.
  lv_triple = |<{ lv_subject }>{ lv_predicate }><{ lv_object }> .|.
  TRANSFER lv_triple TO lv_filename.
  ADD 1 TO lv_count.

  READ TABLE lt_supplier INTO ls_supplier WITH KEY supplier = ls_purchase_order-supplier.
  IF sy-subrc = 0.

    " Vendor Name triple
    lv_predicate = 's:vendorName'.
    lv_object = |"{ ls_supplier-SupplierName }"|.
    lv_triple = |<{ lv_subject }>{ lv_predicate }>{ lv_object } .|.
    TRANSFER lv_triple TO lv_filename.
    ADD 1 TO lv_count.

    " Vendor Name triple
    lv_subject = |s:{ ls_purchase_order-supplier }|.
    lv_predicate = 's:vendorName'.
    lv_object = |"{ ls_supplier-SupplierName }"|.
    lv_triple = |<{ lv_subject }>{ lv_predicate }>{ lv_object } .|.
    TRANSFER lv_triple TO lv_filename.
    ADD 1 TO lv_count.

    " Vendor Country triple
    lv_predicate = 's:country'.
    lv_object = |"{ ls_supplier-Country }"|.
    lv_triple = |<{ lv_subject }>{ lv_predicate }>{ lv_object } .|.
    TRANSFER lv_triple TO lv_filename.
    ADD 1 TO lv_count.

    " Vendor City triple
    lv_predicate = 's:city'.
    lv_object = |"{ ls_supplier-CityName }"|.
    lv_triple = |<{ lv_subject }>{ lv_predicate }>{ lv_object } .|.
    TRANSFER lv_triple TO lv_filename.
    ADD 1 TO lv_count.

    " Triple to define the Vendor as an instance of Vendors
    lv_predicate = 'rdf:type'.
    lv_object = 'sap:Vendor'.
    lv_triple = |<{ lv_subject }>{ lv_predicate }>{ lv_object }> .|.
    TRANSFER lv_triple TO lv_filename.
    ADD 1 TO lv_count.

  ENDIF.

  " Summary information for the Purchase Order
  CONCATENATE 'Purchase Order:' ls_purchase_order-purchaseorder 'Date:' ls_purchase_order-PurchaseOrderDate 'Vendor:' ls_purchase_order-supplier ls_s
  APPEND lv_summary TO lt_summary.
```

SAP Inside Track BARCELONA

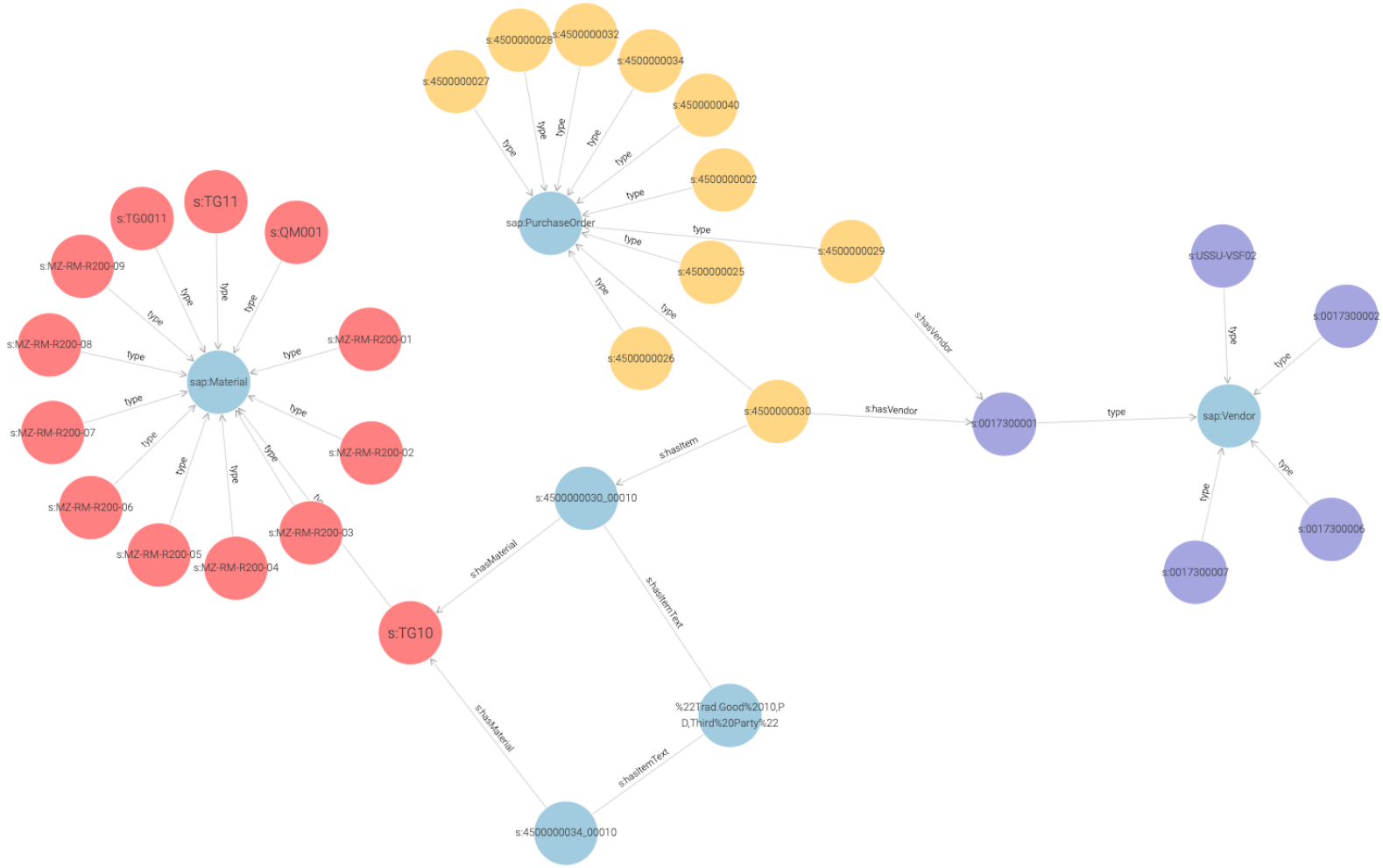
Knowlegde Graphs: simplified example

Representation of the Knowledge Graph (GraphDB)



Knowlegde Graphs: simplified example

REPRESENTATION OF THE KNOWLEDGE GRAPH (GRAPHDB)



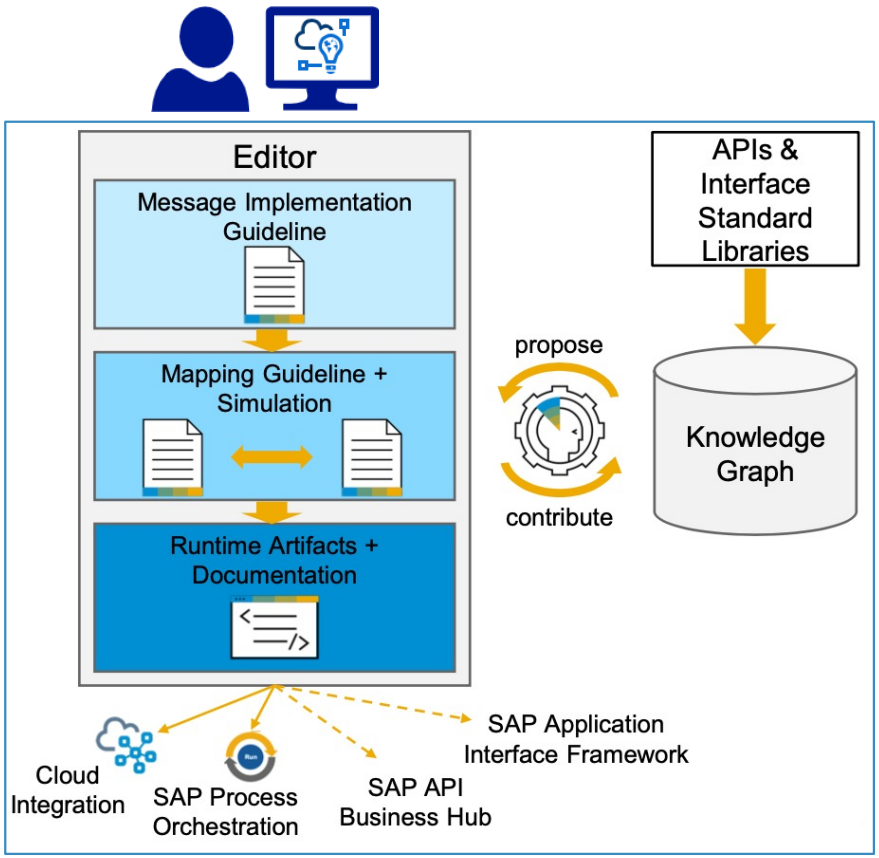
Sulzer

SAP Inside Track BARCELONA

Agenda

- Introduction to Knowledge Graphs
- Building a Knowledge Graph
- **SAP and Knowledge Graphs**

Knowlegde Graphs in SAP
SAP BTP: Integration Advisor



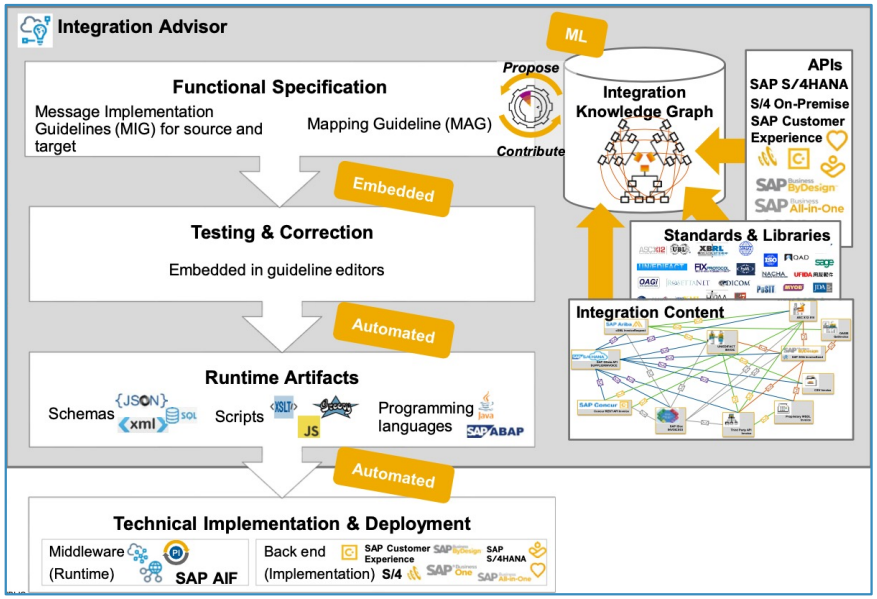
SAP Cloud Platform Integration Advisor drives a new and effective approach in realizing Integrations.

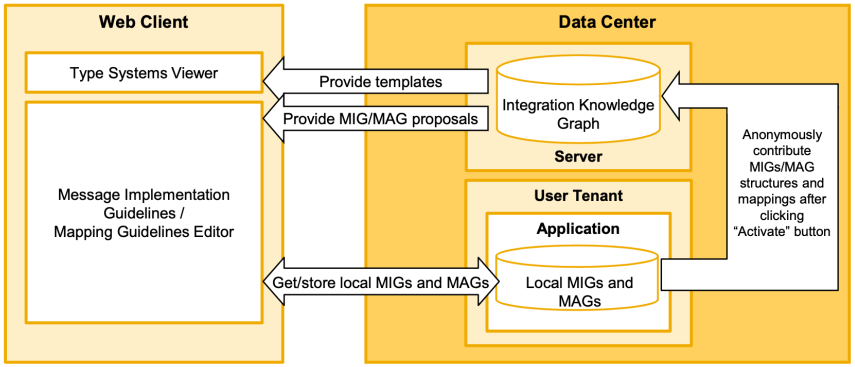
- Simplifies and reduces the effort to a greater extent.
- No need to have add-on tools to document the specifications, no worry about mismatch of specifications document versus artifacts,
- No separate tools for mapping and built-in **ever growing AI based knowledge graph**.

B2B implementation on SAP Cloud Platform Integration and Process Integration is empowered with Integration Advisor. The whole process is simplified and **SAP Cloud Platform Integration Advisor** is definitely an ***Integration game changer!***

Knowledge Graphs in SAP

SAP BTP: Integration Advisor

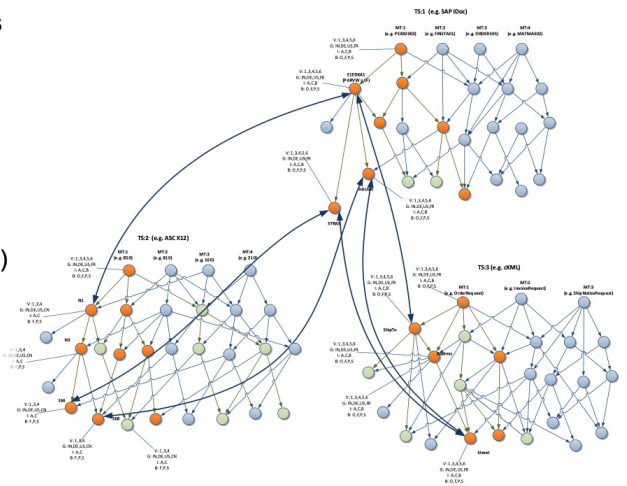




Machine learning using crowd-sourced learnings

Key Capabilities

- **Complete libraries** in form of type systems and all their details with their precise semantics
- Based on **actual, productive** integration artifacts
- Collected by **contributions**
  - Cloud-based MIG/MAG editor
  - Importing existing integration content (mappings)
  - Importing payloads (instance data)
- Centrally provides the **real use of interfaces** of different type systems and how they have been mapped in different contexts
  - Focuses on semantics
  - Using transitivity closure and where/how often used across all libraries and contributions



Knowledge Graphs in SAP

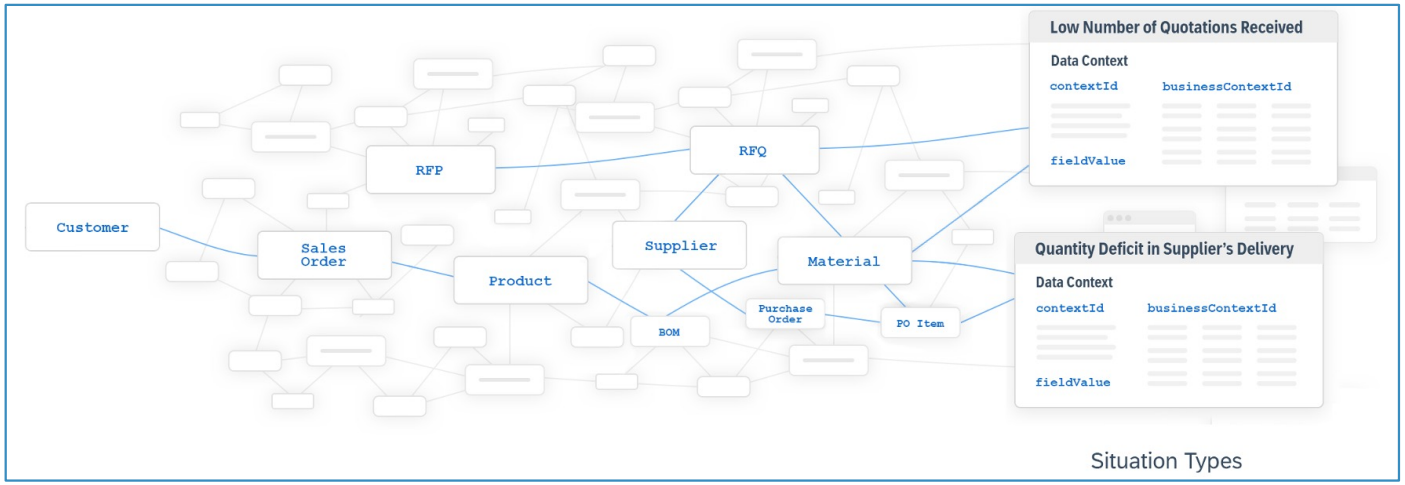
S/4HANA: Situation Handling / Intelligent Situation Automation

Sulzer

Situation Knowledge Graph templates.

- Pending Supplier Confirmation.
- Low number of Quotations Received
- Stock Transport Order Overdue

The knowledge graph is built by extracting, analyzing, and transforming a wide range of metadata, which is available in the various systems at SAP:

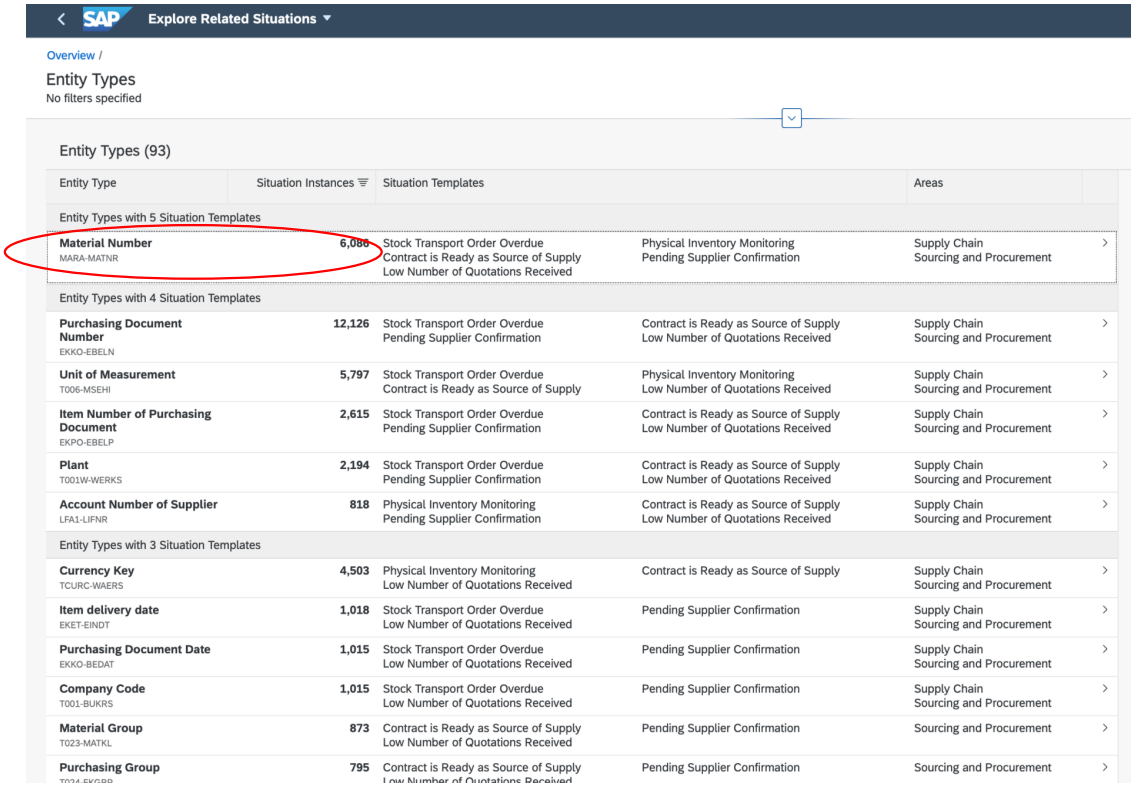


- **S/4HANA** for the virtual data model,
- **Business Situation Type API** for situation master data,
- **FIORI Apps Reference Library** about applications,
- **API Business Hub** about publicly available APIs, CDS views, and Events,
- **Extensibility Cockpit** about business contexts,

SAP Inside Track BARCELONA

Knowledge Graphs in SAP

S/4HANA: Explore Related Situations (New App since 2022)



Explore Related Situations app using the **Situation Knowledge Graph**

New Analytical application, to provide insights that go beyond the scope of a single identified problem.

Explore the relationships between situations, business entities, and other solution-related information.

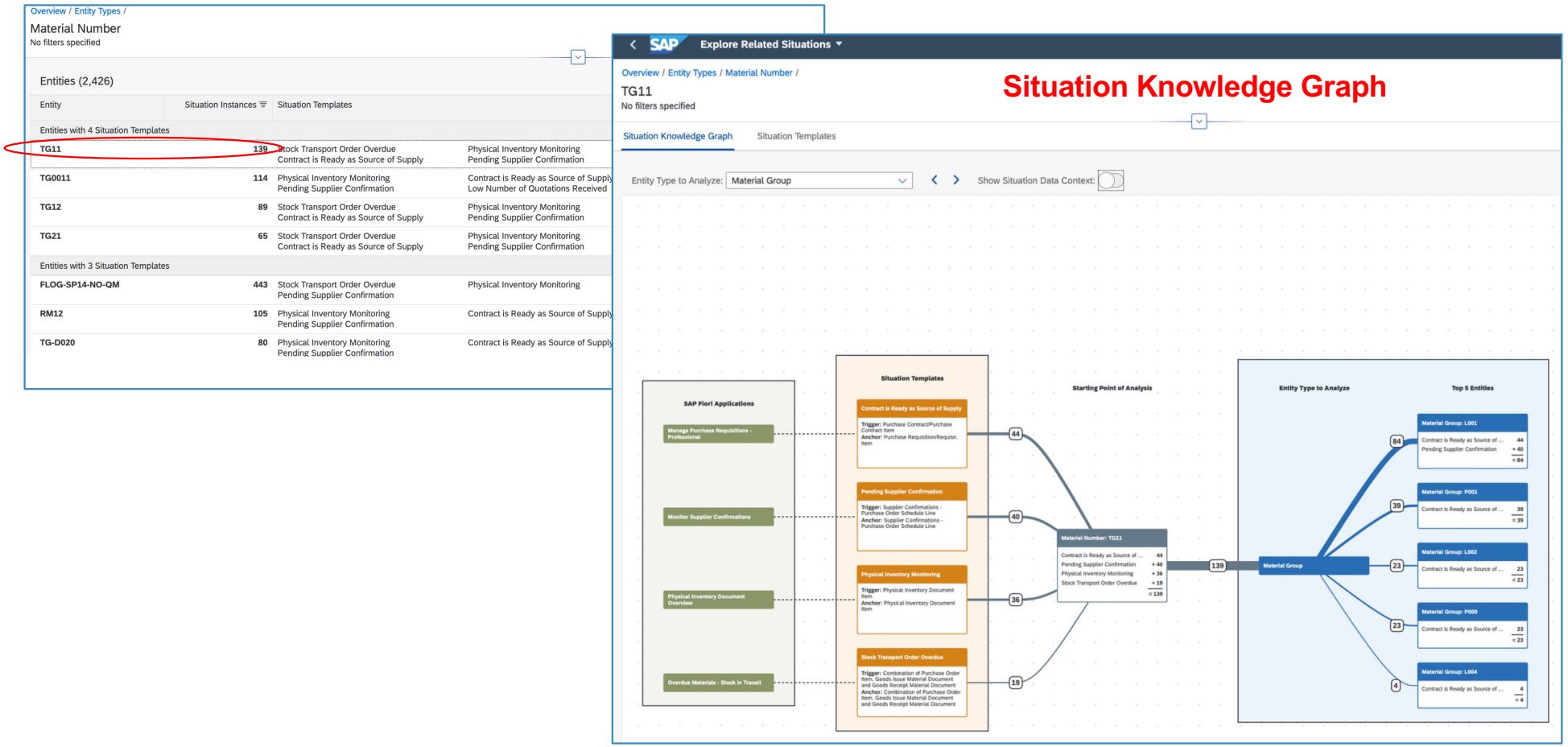
In this application, we follow an interaction pattern that is driven by the business entities themselves.

The entry page shows you which business entity types have the most issues related to the different situation templates and business areas.

In the example, it is the material, indicated by the "Material Number" entity type, which has a high number of situation instances from five different situation templates

Knowledge Graphs in SAP

S/4HANA: Explore Related Situations App

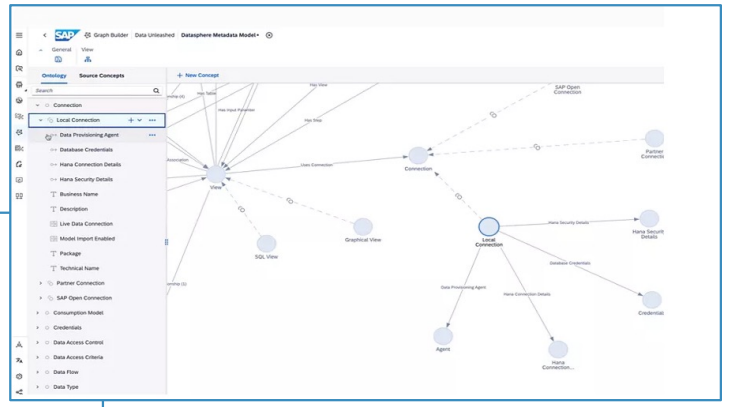


Situation Knowledge Graph

Knowledge Graphs on linked business data

SAP DATASPHERE – H1 2025

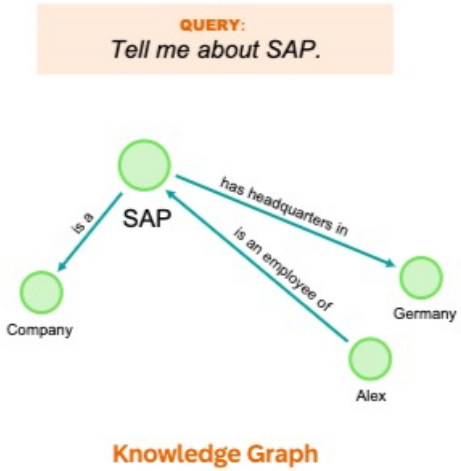
Sulzer



Knowledge Graphs

How they can help in leveraging your data and the motif of embedding into a Data Fabric

- **Improved LLM Query Performance**
  Knowledge graphs enable LLMs to efficiently recognize and navigate relationships from structured data, reducing query costs
- **Compatible with Ad Hoc Queries**
  Well-suited for the types of un-optimized, ad hoc queries often required by messy and arbitrary questions asked in natural language
- **Powerful Data Querying**
  Flexible, schema-less structure works well to describe and connect sparse, widely interconnected datasets

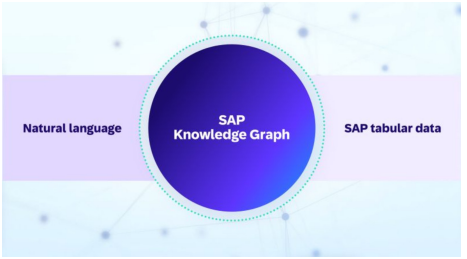


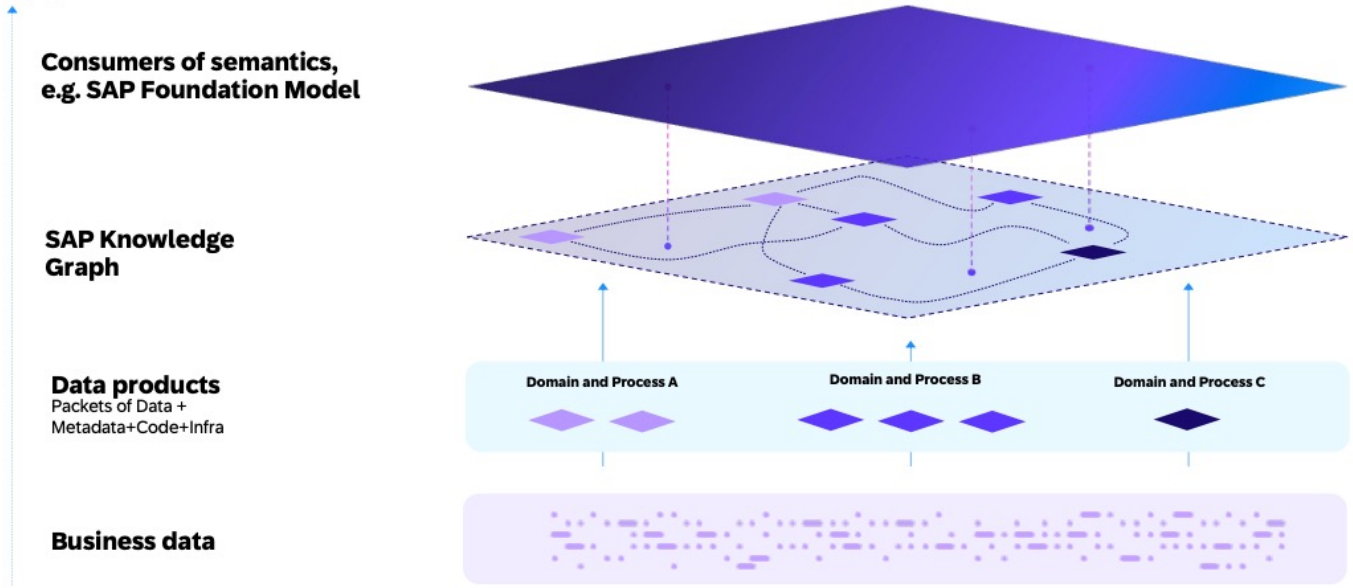
- **SAP Datasphere knowledge graph** automatically creates an **ontology** representing the relationships in the data, including the inherent business context from SAP application sources like SAP S/4HANA
- Extended and augmented via an **ontology editor**
- Finally, the data in SAP Datasphere is automatically applied to this ontology to create a **knowledge graph**

SAP Inside Track BARCELONA

SAP Knowledge Graphs
SEMANTIC LAYER

SAP Knowledge Graph solution captures meaning and context from Lower-level data layers to bring precise semantics to SAP Foundation Model and LLM applications.

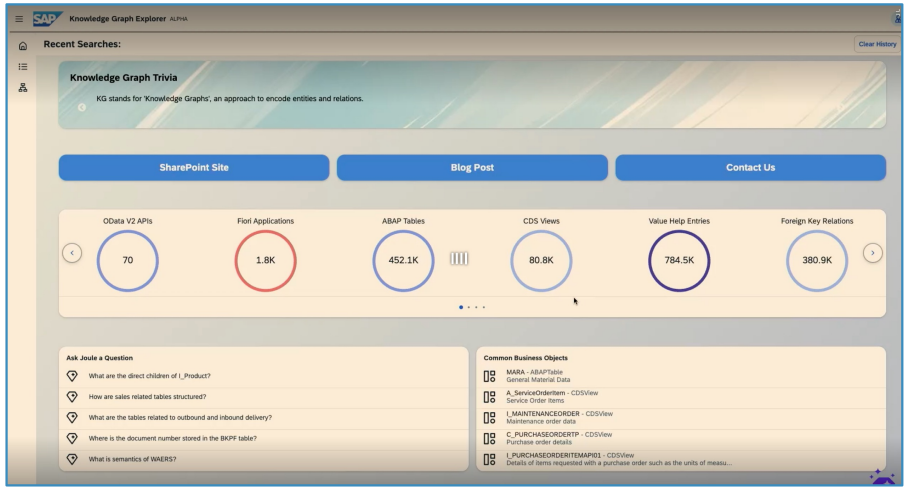


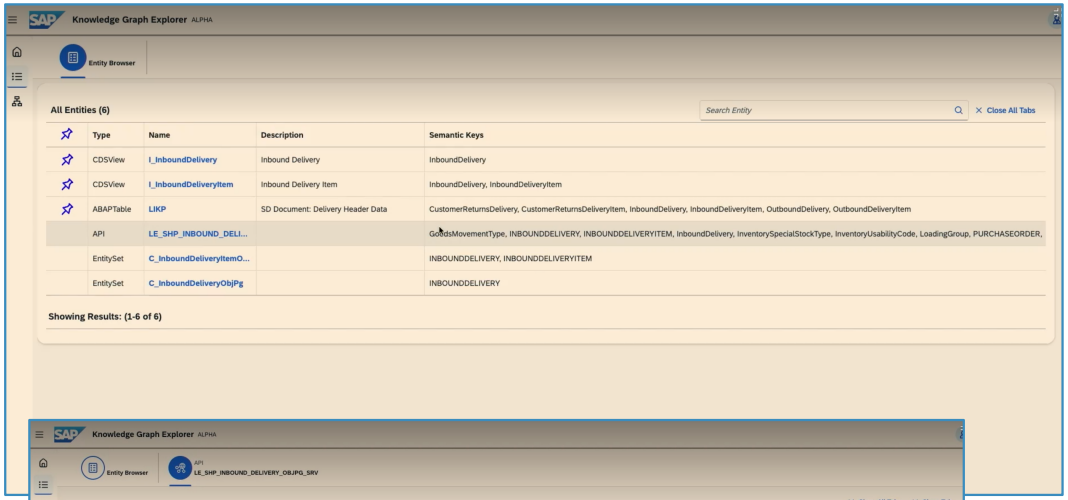


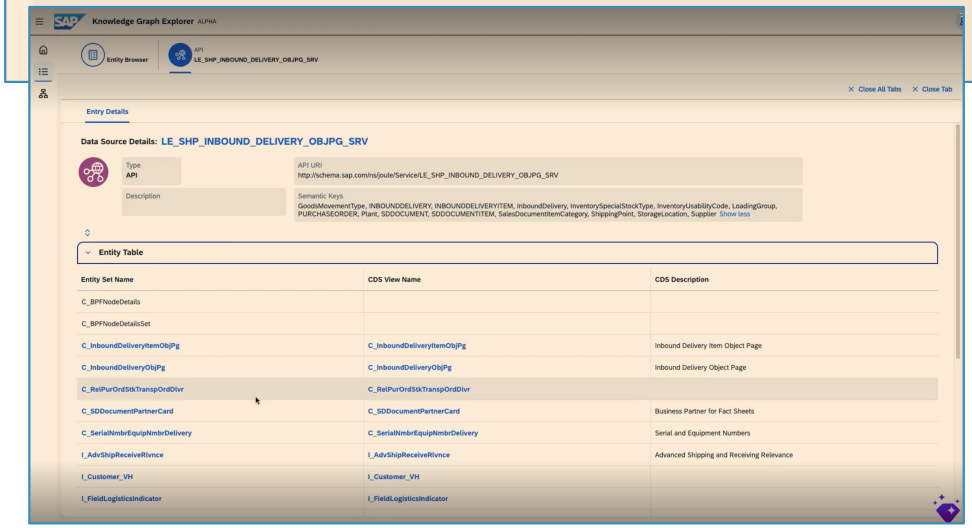
- 452.000 ABAP Tables
- 80.000 CDS Views
- 7,3 Millions of fields

Knowlegde Graphs on linked business data

SAP BTP: KNOWLEDGE GRAPH EXPLORER NAVIGATOR



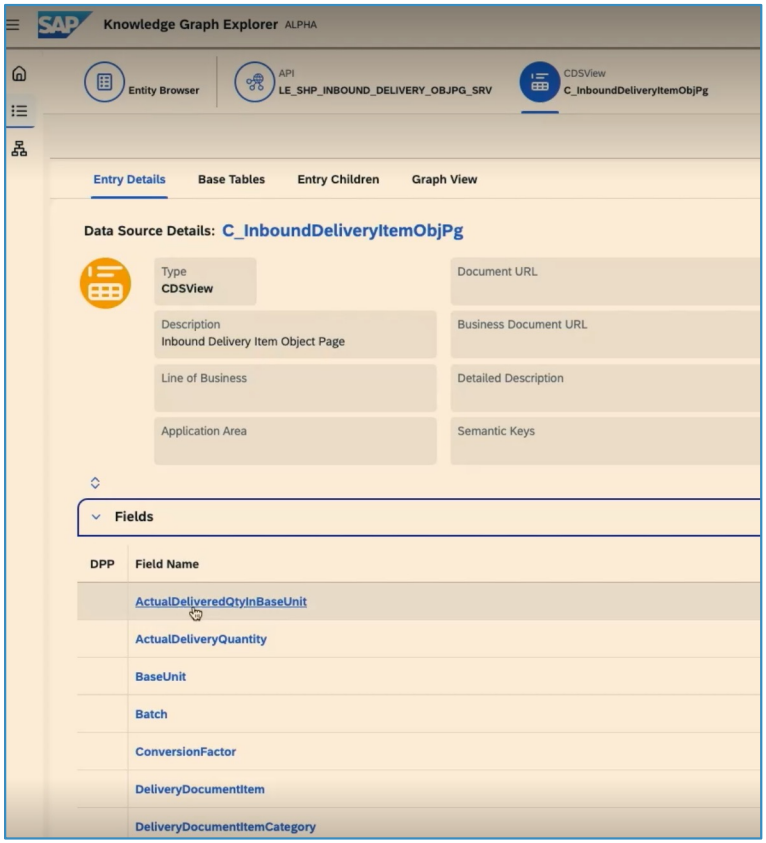


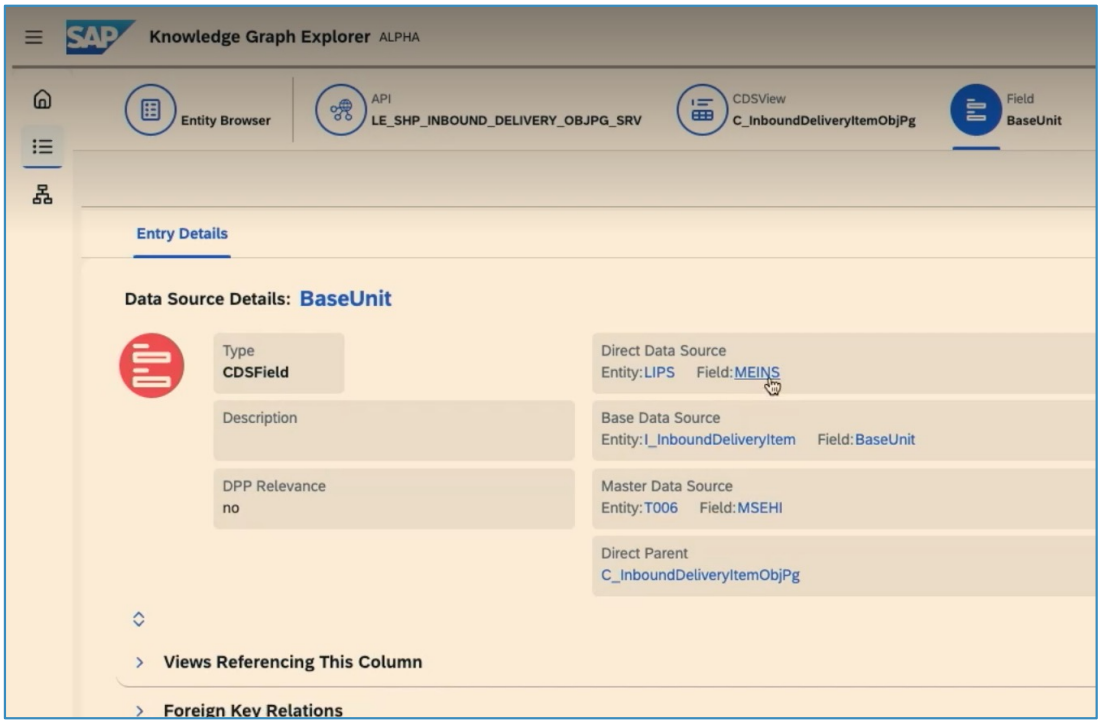


- 452.000 ABAP Tables
- 80.000 CDS Views
- 7,3 Millions of fields

Knowlegde Graphs on linked business data

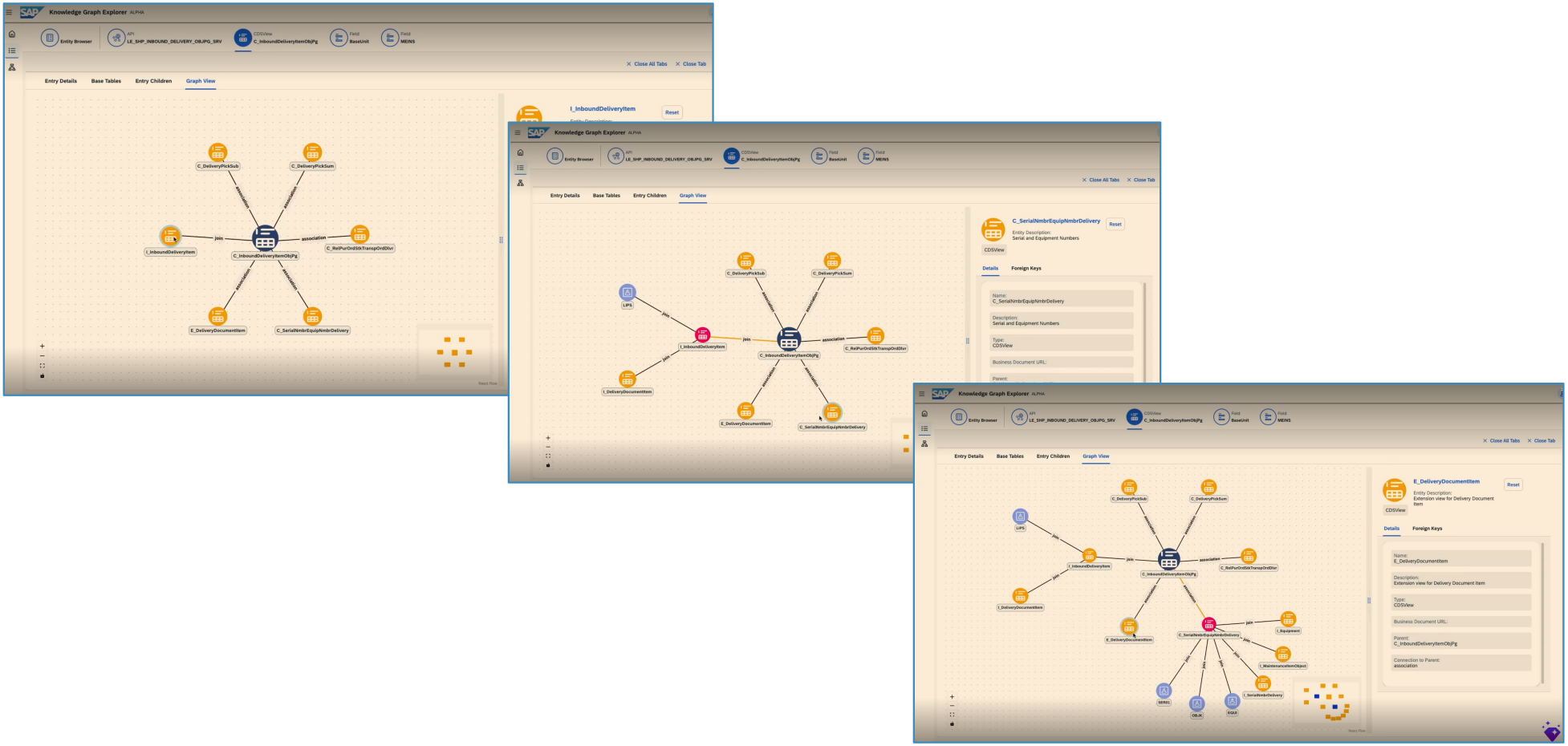
SAP BTP- KG Explorer Navigator: looking for inbound Delivery





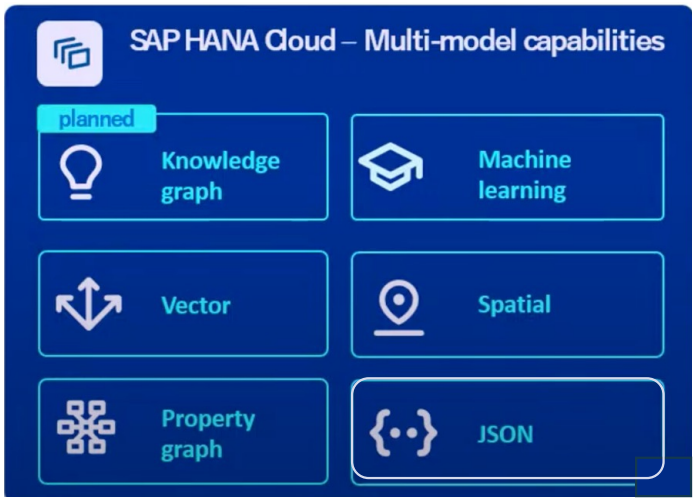
Knowlegde Graphs on linked business data

SAP BTP- KG Explorer Navigator: looking for inbound Delivery



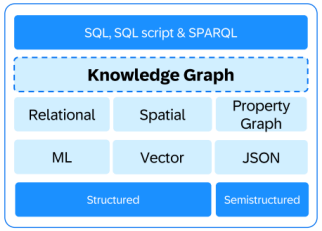
Knowledge Graphs: Connecting the Facts

SAP HANA Cloud's KG Engine for Business Context



With the upcoming **knowledge graph engine**, SAP HANA Cloud gains the **ability to manage and query semantically connected relationships** – all within the same database as other relational and multi-model engines.

- **Native RDF Triple Store:** One of the key features of the SAP HANA Cloud knowledge graph engine is its native support for RDF triple store
- **SQL and SPARQL Interoperability:** The knowledge graph engine will support SPARQL, the specialized query language for knowledge graph data, while tightly integrating it with SQL. This is achieved with the introduction of new SQL.



SAP HANA Cloud knowledge graph engine vs the SAP HANA Cloud graph engine designed to handle **property graphs**

Sulzer

SAP Inside Track BARCELONA

Thank you!

Q & A

Antonio Leites
Sulzer España

antonio.leites@sulzer.es

Backup Slides

- Introduction to Knowledge Graphs
- Building a Knowledge Graph
- SAP and Knowledge Graphs

Knowlegde Graphs vs Property Graphs

SAP HANA Cloud engines

Sulzer

SAP HANA Cloud knowledge graph engine vs the SAP HANA Cloud graph engine designed to handle **property graphs**

Property Graphs: These graphs are ideal for analysing relationships between entities.

- In a property graph, **data** is represented as **nodes** (entities) and **edges** (relationships), where each edge can hold different properties that describe the connection between nodes.

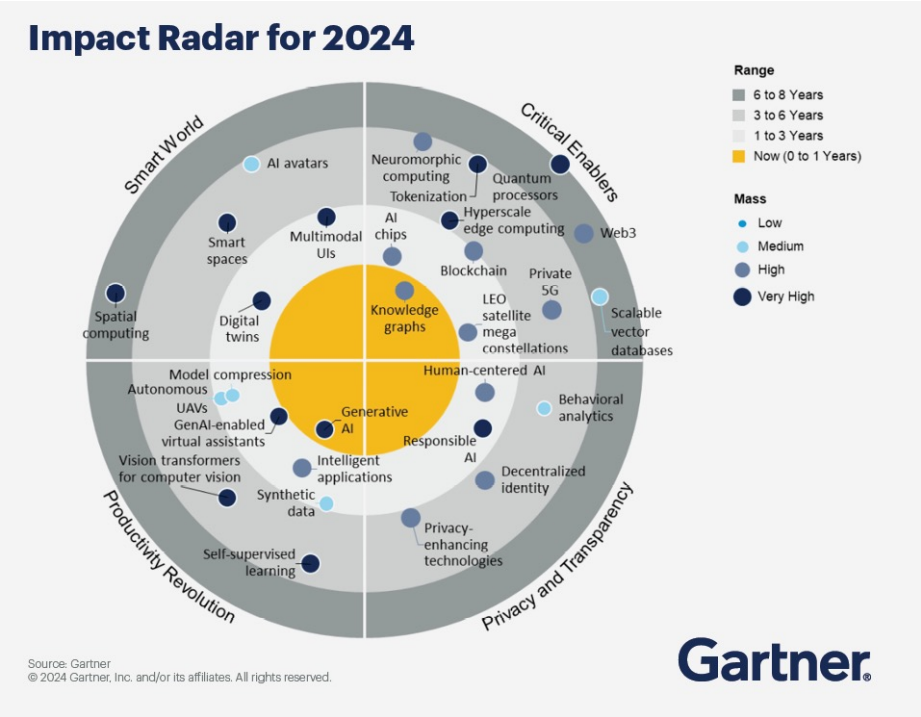
Knowledge Graphs: focus on establishing facts and the logical relationships between those facts.

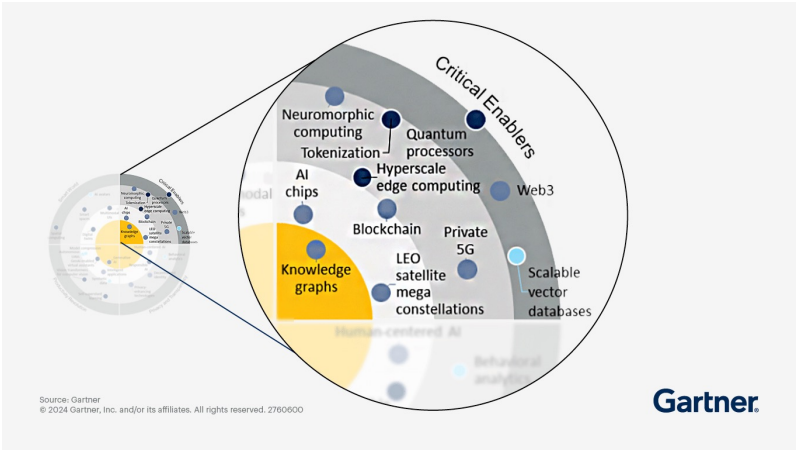
- They use a **subject-predicate-object** structure (triples), where each fact is stored as a relationship.

SAP Inside Track BARCELONA

Knowlegde Graphs: Emerging Technologies and Trends

THE GARTNER EMERGING TECH IMPACT RADAR





Critical Enablers This theme focus on expectations for **emerging applications** — some of which will enable new use cases and others that will enhance existing experiences .

Knowledge Graphs (KGs) are **machine-readable data structures** that describe the relationship between heterogeneous data via a network of nodes and links.

GenAI actions for tech product leaders:

…

Prioritize the most prevalent use cases, such as **enterprise search/knowledge mining** and virtual agents, as these evidently already deliver real value to users.

…