milan@datajoin.net

http://datajoin.net

DataJoin

Data Model Mapping and Repository

[Document subtitle]

Milan Patel
5-2-2025

Table of Contents

Overview

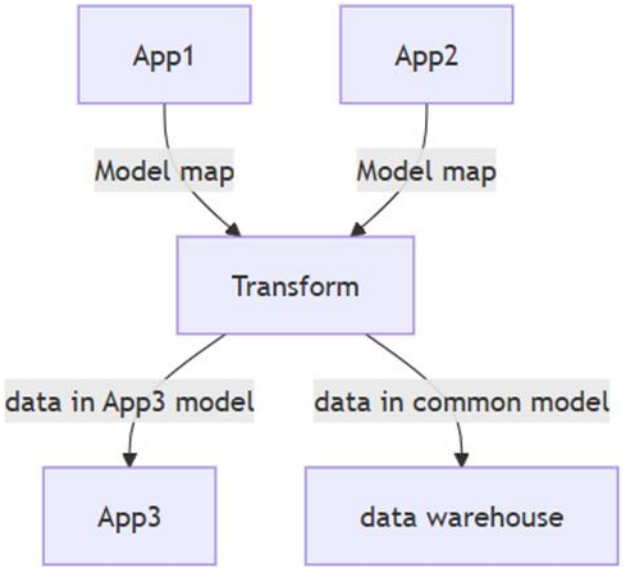
Problem: The process of mapping between models or schemas is time-consuming and requires manual effort. Once a subject matter expert completes the mapping, writing code to transform the data becomes straightforward.

Solution: The process of model or schema mapping can be semi-automated using algorithms. It is recommended that an organization maintains a mapping repository, which can be utilized for various projects such as application integration, data warehousing, data fabric, and AI algorithms.

DataJoin.net provides in-depth education and consultation on Model map repository.
https://github.com/milan888-design/correlate-data

App(data model) to App(data model) data transfer

Each App has its own data model. Thus, App to App data transfer requires ETL (extract data from source App, transform the data using model map, and load data in target App)



Data Model

An App has Entity type data and Instance (item) data. For example, laptop is a product type, specific serial number laptop is an instance(item). Typically, type data are small in size while instance (item) data are in large size. Whenever entities are stored in structure format there is a data model.

Both Entity type and Instance (item) data have the following:

- Entities have hierarchy
- Entities have attributes at each level in hierarchy
- Entities have relationships with other entities
- Some entities are singular such as Customer, Organization, Equipment.
- Some entities are composite since they are a combination of two or more entities such as Employment history (combination of person, organization and time period), Sales Order (Sales Order is item, customer, ship to address)

model1

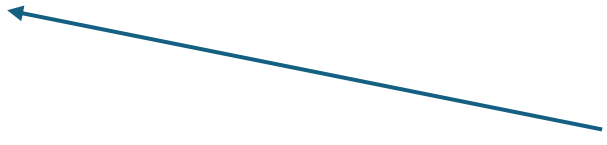
db1

product_type (category)

product_type_name
laptop
desktop

customer

customer_id	customer_name
c1	xyz
c2	abc



sales_order

order_id	order_type	description	product_type_name	customer_id	quantity	order_date
ord1	retail	new order	laptop	c1	200	2025-04-25

Data models have two parts:
Data model part 1: Schema (structure) description where entities (type and instance), hierarchy and attributes are stored.
Data model part 2: Entity type (category) list (its hierarchy and attributes)

Data model part 1 schema example: table product_type, table attributes product_type_id, product_type_name
Data model part 2 entity type (category) example: product_type = p1:laptop, p2:desktop

Model and mapping repository code

test _etl_db1 in postgreSQL is created using admin UI. This database has the following tables.
product_type: category or type data
sales_order: transaction or instance data
st_mapping_set_v3: mapping set. This is considered as mapping repository.
st_mapping_set_detail_v3: specific value or schema column level mapping. This is considered as mapping repository.
st_schema_table: all the table in one or more database. This is considered as model repository
st_schema_attribute: all attributes for all tables for one or more databases. This is considered as model repository.
Use test_etl_db1_backup.sql to create these tables and populate with test data.

Test_etl_db2 in postgreSQL is created using admin UI. This database has the following tables.
Item_type: category or type data
Sales_txn: transaction or instance data
Use test_etl_db2_backup.sql to create these tables and populate with test data.

test_etl_db1 and test_etl_db2 data and foreign key relationship between tables

model1

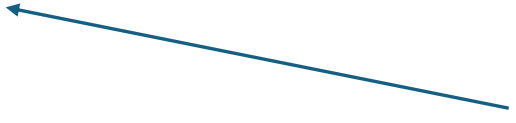
db1

product_type (category)

product_type_name
laptop
desktop

customer

customer_id	customer_name
c1	xyz
c2	abc



sales_order

order_id	order_type	description	product_type_name	customer_id	quantity	order_date
ord1	retail	new order	laptop	c1	200	2025-04-25

milan@datajoin.net

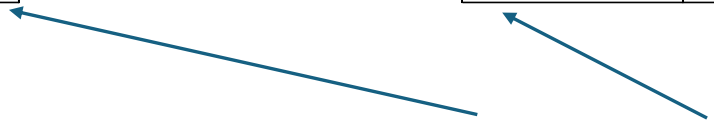
model2
db2

item_type (category)

item_type_name
handheld
tower

customer_names

customer_id	customer_name
c1	xyz
c2	abc



sales_transactions

id	txn_type	description	item_type_name	customer_id	qty	txn_date
ord1	retail	new order	handheld	c1	200	2025-04-25

Model mapping repository tables in test_etl_db1

Data model has two parts.
Data model part 1: Schema (structure) description where entities (type and instance), hierarchy and attributes are stored.
Data model part 2: Entity type list (its hierarchy and attributes). You may consider type (category) tables as part of data model.

Model mapping repository stores mapping as defined below:

1. Entity to entity mapping (class to class or value to value mapping). This is part 2 of data model
2. Schema to Schema mapping (structure to structure) mapping. This is part 1 of data model

Two tables are used to store the above mentioned mapping (st_mapping_set_v3, st_mapping_set_detail_v3)

st_mapping_set_v3

set_id	leftmodel	leftdatabas	lefttable	lefttattribute	maptype	rightmodel	rightdatabase	righttable	rightattribute
set1	model1	db1	sales_order		schema to schema	model2	db2	sales_txn	
set2	model1	db1	product_type	product_type_name	category to category	model2	db2	item_type	item_type_name

st_mapping_set_detail_v3

row_id	set_id	pair_id	leftattribute	rightattribute	SpecificVsGeneral	FromAttribute Value	ToAttribute Value	Match Rating	MapBy	Reviewed By	tranform type	transform lookupset
1	set1		product_type_name	item_type_name							lookupset	set2
2	set2				specifictospecific	laptop	handheld	100	algorithm	mtp		

Model mapping (schema and category/type) needs the concept of mapping set. Since there can be thousands of mappings, it is necessary to separate them using the set concept.

For schema to schema mapping, create a set for combination of two tables that need mapping (set1)
For set1, create rows in st_mapping_set_detail_v3 for combination of two columns/ attributes that are mapped.
In set_id=set1 and row_id=1, two schema columns are mapped. For this combination of columns, set2 is assigned for value mapping.
In schame to schema mapping, it is possible to have many columns mapped to one column. For example, columns names year, month, day can be mapping to date column in target database. The mapping should use this format: year:month:day = date
JSON or XML schema can be used in the above table by using JSON path or XML path.

Set2 is category to category mapping. Set2 has rows in st_mapping_set_detail_v3 table where values are mapped. For example, row2 has laptop mapped to handheld. This value mapping can be manual or it can be partially automated using the following python files.
modelmap_postgres_query1_query2_v2.py: Use two list for mapping and result is populated in st_mapping_set_detail_v3

Set maptype

Category to Category
Category to Schema
Component to Assembly
Compnent Item to Assembly Item
Category to Item
Schema to Schema
Schema to Category
Schema to Item
Category to Item

Category to Category
Assembly to Component
Assembly Item to Component Item
Item to Category
Item to Item
Item to Schema

Model repository for data model part 1 (schema)

Data model has two parts.
Data model part 1: Schema (structure) description where entities (type and instance), hierarchy and attributes are stored.
Data model part 2: Entity type list (its hierarchy and attributes). You may consider type (category) tables as part of data model. This is actual application tables with category or type data. Thus, there is not necessary need to store this data as model repository.

datajoin_generic_list_tables_v3.py: to use database name and extract all table names and populate st_schema_table
datajoin_generic_list_tables_columns_v3.py: to use database name and extract all columns (attributes) and populate

st_schema_table

schema_table_key	schema_key	tablename
test_etl_db1|product_type	test_etl_db1	product_type
test_etl_db1|sales_order	test_etl_db1	sales_order
test_etl_db1|st_mapping_set_detail_v3	test_etl_db1	st_mapping_set_detail_v3
test_etl_db1|st_mapping_set_v3	test_etl_db1	st_mapping_set_v3
test_etl_db1|st_schema_attribute	test_etl_db1	st_schema_attribute
test_etl_db1|st_schema_table	test_etl_db1	st_schema_table

st_schema_attribute (subset of data only)

schema_attribute_key	schema_table_key	tablename	tableattribute	data_type
test_etl_db1|product_type|product_type_name	test_etl_db1|product_type	product_type	product_type_name	TEXT

test_etl_db1|sales_order|customer_id	test_etl_db1|sales_order	sales_order	customer_id	TEXT
test_etl_db1|sales_order|description	test_etl_db1|sales_order	sales_order	description	TEXT
test_etl_db1|sales_order|order_date	test_etl_db1|sales_order	sales_order	order_date	TEXT
test_etl_db1|sales_order|order_id	test_etl_db1|sales_order	sales_order	order_id	TEXT
test_etl_db1|sales_order|order_type	test_etl_db1|sales_order	sales_order	order_type	TEXT
test_etl_db1|sales_order|product_type_name	test_etl_db1|sales_order	sales_order	product_type_name	TEXT
test_etl_db1|sales_order|quantity	test_etl_db1|sales_order	sales_order	quantity	TEXT
test_etl_db1|st_mapping_set_detail_v3|activeflag	test_etl_db1|st_mapping_set_detail_v3	st_mapping_set_detail_v3	activeflag	TEXT
test_etl_db1|st_mapping_set_detail_v3|leftattribute	test_etl_db1|st_mapping_set_detail_v3	st_mapping_set_detail_v3	leftattribute	TEXT

ETL (extract, transform, load) and mapping code

To find set id that mapps two schema (modelmap_find_set_id.py)

When transforming to target schema attribute. You have to find what set id used to map two tables

Block A

```
leftdatabase='test_etl_db1'
lefttable='sales_order'
rightdatabase='test_etl_db2'
righttable='sales_txn'
maptype='schema to schema'
```

Block B

```
queryselect = text("""
    SELECT  set_id from st_mapping_set_v3
    WHERE  leftdatabase=:leftdatabase and lefttable=:lefttable and rightdatabase=:rightdatabase and righttable=:righttable and
maptype=:maptype
    """)
```

The result of the above is set1 which is used in the following to find the target value

To find value value mapped from model map repository (modelmap_find_lookupset_value.py)

Block A

Function to find lookukp set_id based on the target/right attribute anem

```
def find_lookupsetid(var_set_id, var_rightattribue):
uses set_id and rightattribute

queryselect = text("""
        SELECT leftattribute,transform_lookupset from st_mapping_set_detail_v3
        WHERE set_id=:set_id and rightattribute=:rightattribute
        """)
   return value_set_id # This line makes value_set_id the return value
# Now, when you call the function, the returned value will be stored in lookupset_id
lookupset_id = find_lookupsetid('set1','item_type_name')
```

Block B

Function to find the value from detail table using leftattribute value

```
def find_lookupvalue(var_valset_id, var_leftattributevalue):
  queryselect2 = text("""
      SELECT rightattributevalue from st_mapping_set_detail_v3
      WHERE  set_id=:set_id and leftattributevalue=:leftattributevalue
      """)
lookupvalue= find_lookupvalue(lookupset_id,'laptop')
```

st_mapping_set_v3

set_id	leftmodel	leftdatabase	lefttable	lefttattribute	maptype	rightmodel	rightdatabase	righttable	rightattribute
set1	model1	db1	sales_order		schema to schema	model2	db2	sales_txn	
set2	model1	db1	product_type	product_type_name	category to category	model2	db2	item_type	item_type_name

st_mapping_set_detail_v3

row_id	set_id	pair_id	leftattribute	rightattribute	SpecificVsGeneral	FromAttribute Value	ToAttribute Value	Match Rating	MapBy	Reviewed By	tranform type	transform lookupset
1	set1		product_type_name	item_type_name							lookupset	set2
2	set2				specifictospecific	laptop	handheld	100	algorithm	mtp		

To pick data from sales_order and insert in sales_txn (modelmap_postgres_etl.py)

Block A

Select data from source table sales_order

```
select_statement1 = text('SELECT order_id,order_type,description,product_type_name,customer_id,quantity,order_date FROM sales_order')
```

Block B

for every row selected use the following schema to schema transform.

```
    id = row.order_id
    txn_type = row.order_type
    description = row.description
    item_type_name = row.product_type_name # This maps to product_type_name from sales_order
    customer_id = row.customer_id
    qty = row.quantity
    txn_date = row.order_date
```

Block C

The following insert statement variable values are assigned above. It is possible to use schema to schema mapping and value mapping to do the transformation.

```
queryinsert = text("""
    INSERT INTO sales_txn (id,txn_type,description,item_type_name,customer_id,qty,txn_date)
    VALUES (:id, :txn_type, :description, :item_type_name, :customer_id, :qty, :txn_date)
    """)
```

To create automatic mapping between two entity types (modelmap_postgres_query1_query2_v2.py)

Block A

Provide input for the mapping such as where the mapping will be stored

```
ratio_limit = 50
mapby = 'fuzzywuzzy'
set_id = 'set2'
```

Block B

list1 or source values

```
select_statement1 = text('SELECT product_type_name FROM product_type')
```

list2 or target values

```
select_statement2 = text('SELECT item_type_name FROM item_type')
```

Block C

use many to many compare for list1 and list2

```
try:
  for value1_orig in lower_list1: # Use _orig to preserve original value for fuzzywuzzy
    for value2_orig in lower_list2:
      ratio = fuzz.ratio(value1_orig, value2_orig)
      if ratio > ratio_limit:
```

Block D

use insert string and assign values to the variables.

```
queryinsert = text("""
  INSERT INTO st_mapping_set_detail_v3 (set_id, leftAttributevalue, rightAttributevalue, matchrating, mapby)
  VALUES (:set_id, :left_attr_value, :right_attr_value, :match_rating, :map_by)
""")

# Execute the statement with a dictionary of parameters
result_insert = session_insert.execute(queryinsert, {
  'set_id': set_id,
  'left_attr_value': value1_orig, # Use original values, SQLAlchemy will escape
  'right_attr_value': value2_orig, # Use original values, SQLAlchemy will escape
  'match_rating': ratio,
  'map_by': mapby
})
```

Data model and mapping concepts

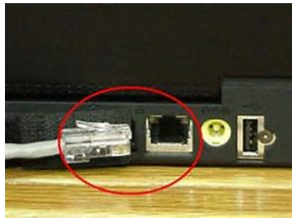
Entities are called with different names by different applications

People call the same things with different names.

- Example 1: Soft Drink = Soda
- Example 2: Country = Nation = State
- Example 3: **State = Province**
- Example 4: County = District
- Example 5: Zip Code = Postal Code

People call the different things with the names.

- Example 6: Unit (of mip) <> Unit (of measurement)
- Example 7: **Port (for ships) <> Port (of network router)**
- Example 8: Arms (parts of body) <> Arms (weapon)



milan@datajoin.net

Same entity is stored in a variety of formats (schema / structure)

Person entity with its attributes is stored in SQL, RDF, JSON, Key value pair, and XML formats.

SQL Table

Table: PERSON

NAME	HAIRCOLOR	ROLE
John Doe	Brown	Driver

RDF

Subject	Predicate	Object
Person.id1	Name	john doe
Person.id1	Hair color	Black
Person.id1	Role	Driver

XML

```
Person
 xsi:noNamespaceSchemaLocation="person.xs">
 <Person  id="id1">
   <Name>John Doe</Name>
   <Haircolor>Black</Haircolor>
   <Role>Driver</Role>
 </Person>
</Person_Info>
```

JSON

Person

```
{"Person  id":"id1"> ,
  "Name":"John Doe",
  "Haircolor":"Black",
  "Role":"Driver"}
```

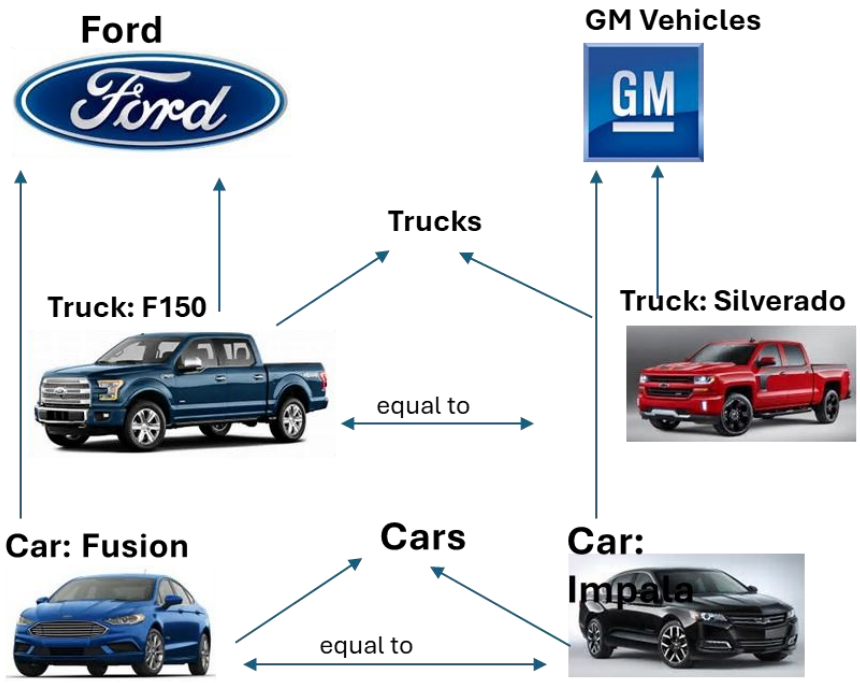
Accumulo / Hbase

row_id	Column Group	column	value	timestamp
id1	Person	Name	John Doe	2/14/14 11:30
id1	Person	Hair color	Black	2/14/14 11:35
id1	Person	Role	Driver	2/14/14 11:35

The same content (data) can be stored in different formats. Each format has pros and cons.
Physical model(SQL table, XML, and JSON) may be similar to Logical model (classes).
Physical model (RDF, Accumulo/Hbase) are never similar to Logical model(classes).
Schema mapping depends upon type of Physical model.

Mapping between entities are two types:

Entity to entity mapping (class to class mapping)



A thing can belong to multiple Categories

- F150 “belong to” Ford Vehicles

(primary family/ category)

- F150 “belong to” Trucks

(second family/categories)

“Mapping” is done at the same level

- F150 is “equal to” Silverado.
- F150 is “not equal to” GM Vehicles (different level)
- Ford Vehicles are “equal to” GM Vehicles
- Fusion is “equal to” Impala

Use of Category for query

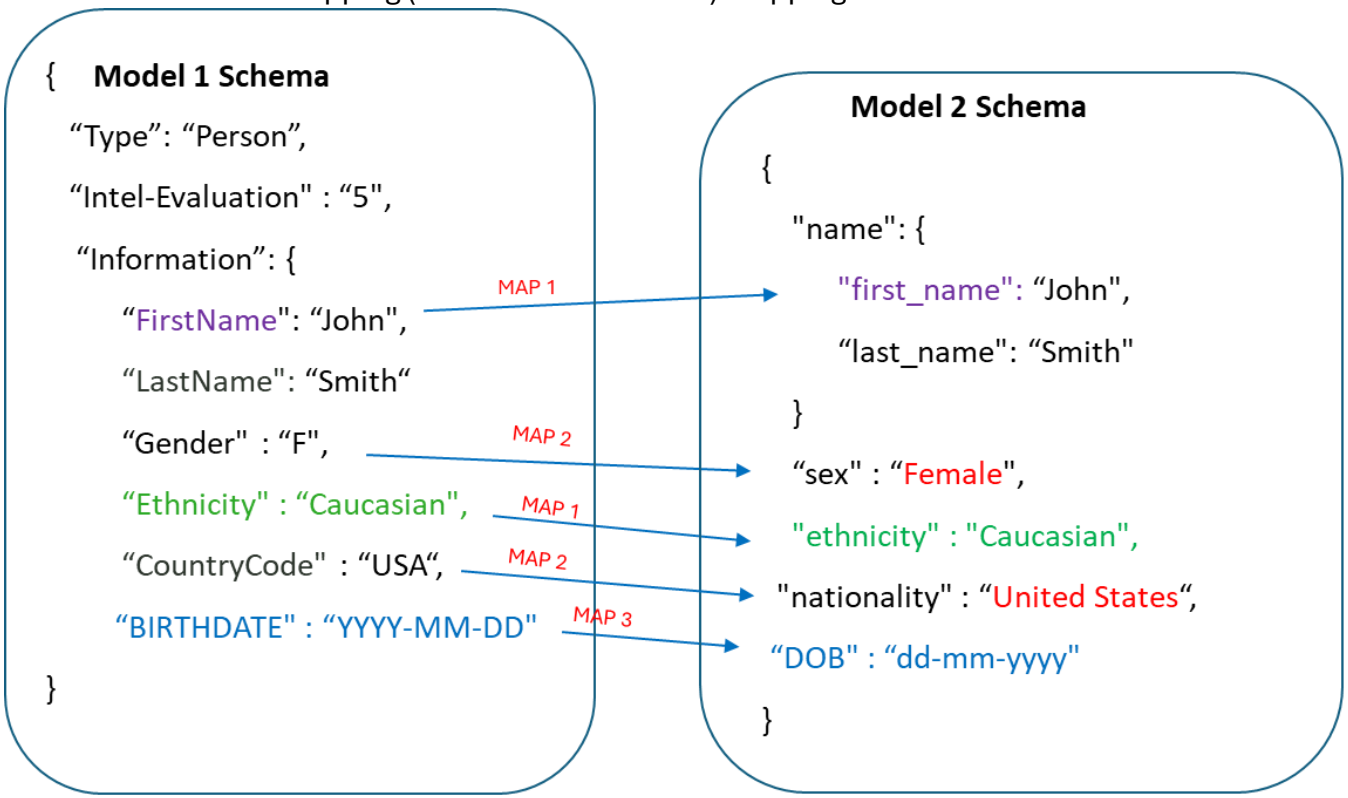
- If you know a category (GM Vehicles), you can find all its members
- If you know a member (Silverado), then, you can find all multiple categories it belong to

Use of mapping for query

- If you know a member (F150), then, you can find an equal member (Silverado) in another category only if you have explicit mapping (F150=Silverado)

milan@datajoin.net

Schema to Schema mapping (structure to structure) mapping



Entity hierarchy

Hierarchy	App1	Symbology 2525B	App2	WEG	Ontology
Level1	EQP,NFI	warfighting			Vehicle
Level2	ARMORED VEH,NFI	GROUND TRACK EQUIPMENT		Ground Systems	GroundVehicle
Level3	TANK,NFI	TANK	TANKS(P)	TANKS	Tank
Level4	TANK,MBT,125MM,T-72	TANK HEAVY	MAIN BATTLE TANKS(PMBV)	Main Battle Tanks	
Level5	TANK,MBT,125MM,T-72BK ROGATKA	TANK HEAVY	MAIN BATTLE TANKS(PMDU)	Main Battle Tanks	

There are multiple (class) hierarchies
Each hierarchy has its own Pros and Cons.
Each industry has its own hierarchy standards for Entities/ Classes they use.
Hierarchy (top to bottom) describes an Entity “from general to more specific”
Hierarchy has many purposes: Correlation, Summary Statistics, Filters, Query

Map between specific to specific, specific to general

From JC3IEDM	To Symbology	ValueDirection
Bridge	BRIDGE	bidirection
Arch	BRIDGE	LeftToRight
BoatFloatingBridge	BRIDGE	LeftToRight
BoxGirder	BRIDGE	LeftToRight
Cantilever	BRIDGE	LeftToRight
Ferry	BRIDGE	LeftToRight
Lift	BRIDGE	LeftToRight
MediumGirderMilitaryBridge	BRIDGE	LeftToRight
NotOtherwiseSpecifiedFloatingBridge	BRIDGE	LeftToRight
PontoonFloatingBridge	BRIDGE	LeftToRight
Raft	BRIDGE	LeftToRight
Slab	BRIDGE	LeftToRight
Stringer	BRIDGE	LeftToRight
Suspension	BRIDGE	LeftToRight
Swing	BRIDGE	LeftToRight
Truss	BRIDGE	LeftToRight
VehicleLaunchedMilitaryBridge	BRIDGE	LeftToRight

From JC3IEDM	To DCGS Norm	ValueDirection
Bridge	Bridges	bidirection
Arch	Bridges	LeftToRight
BoatFloatingBridge	Bridges	LeftToRight
BoxGirder	Bridges	LeftToRight
Cantilever	Bridges	LeftToRight
Ferry	Bridges	LeftToRight
Lift	Bridges	LeftToRight
MediumGirderMilitaryBridge	Bridges	LeftToRight
NotOtherwiseSpecifiedFloatingBridge	Bridges	LeftToRight
PontoonFloatingBridge	Bridges	LeftToRight
Raft	Bridges	LeftToRight
Slab	Bridges	LeftToRight
Stringer	Bridges	LeftToRight
Suspension	Bridges	LeftToRight
Swing	Bridges	LeftToRight
Truss	Bridges	LeftToRight
VehicleLaunchedMilitaryBridge	Bridges	LeftToRight
?	Bridges, Highway Viaducts And Trestles	?
?	Bridges, Contingency Highway Bridging	?
?	Bridges, Contingency Railroad Bridging	?
?	Dual Purpose Highway-Railroad Bridges	?

Specific to specific is desirable, but may not be an option in some cases
Specific to General is allowed, however, it is necessary show that there is loss of details.
General to specific should not be allowed.
For Class mapping, tag the mapping if it is specific to specific or specific to general

Level of detail varies from model to model

Symbology
Entity type: INSTALLATION

Level 1	**Level 2**
RAW MATERIAL PRODUCTION/STORAGE	
PROCESSING FACILITY	
EQUIPMENT	
MANUFACTURE	
SERVICE, RESEARCH, UTILITY FACILITY	
MILITARY MATERIEL	
FACILITY	
	NUCLEAR ENERGY
	ATOMIC ENERGY REACTOR
	NUCLEAR MATERIAL PRODUCTION
	WEAPONS GRADE NUCLEAR MATERIAL STORAGE
	AIRCRAFT PRODUCTION & ASSEMBLY
	AMMUNITION AND EXPLOSIVES PRODUCTION
	ARMAMENT PRODUCTION
	MILITARY VEHICLE PRODUCTION
	ENGINEERING EQUIPMENT PRODUCTION
	BRIDGE
	CHEMICAL & BIOLOGICAL WARFARE PRODUCTION
	SHIP CONSTRUCTION
	MISSILE & SPACE SYSTEM PRODUCTION
GOVERNMENT	
LEADERSHIP	
MILITARY BASE/FACILITY	
	AIRPORT/AIRBASE
	SEAPORT/NAVAL BASE
TRANSPORT FACILITY	
MEDICAL FACILITY	
HOSPITAL	

MIM

Entity type: Facility
Level 1
DryDock
Runway
Jetty
LandMinefield
Apron

OtherMilitaryObstacle
MaritimeMinefield
Quay
MedicalFacility
MilitaryObstacle
Berth
OtherFacility

ComposedAntiTankObstacle
Basin
Harbour
Road
Airfield
Depot
Railway
WireObstacle
Bridge
Minefield
Network
Anchorage
Slipway

Level 2
Arch
BoatFloatingBridge
BoxGirder
Cantilever
Ferry
Lift
MediumGirderMilitaryBridge
NotOtherwiseSpecifiedFloatingBridge
PontoonFloatingBridge
Raft
Slab
Stringer
Suspension
Swing
Truss
VehicleLaunchedMilitaryBridge

Multi segment code: Each segment should be mapped separately

TABLE A-I. Symbol ID code positions and categories.

CODING SCHEME (1) (POSITION 1)	AFFILIATION (1) (POSITION 2)	BATTLE DIMENSION (1) (POSITION 3)	STATUS (1) (POSITION 4)
S - WARFIGHTING G - TACTICAL GRAPHICS W - METOC I - INTELLIGENCE M - MAPPING (reserved - under development) O - Military Operations Other Than War (MOOTW)	P - PENDING U - UNKNOWN A - ASSUMED FRIEND F - FRIEND N - NEUTRAL S - SUSPECT H - HOSTILE J - JOKER K - FAKER O - NONE SPECIFIED	P - SPACE A - AIR G - GROUND S - SEA SURFACE U - SEA SUBSURFACE F - SOF X - OTHER (No frame)	A - ANTICIPATED /PLANNED P - PRESENT
FUNCTION ID (6) (POSITION 5 - 10)	**SYMBOL MODIFIER (2) (POSITION 11, 12)**	**COUNTRY CODE (2) (POSITION 13, 14)**	**ORDER OF BATTLE (1) (POSITION 15)**
See tables A-III through A-IX for specific values.	See table A-II for specific values	See FIPS Pub series 10	A - AIR OB E - ELECTRONIC OB C - CIVILIAN OB G - GROUND OB N - MARITIME OB S - STRATEGIC FORCE RELATED

Symbology 2525B – 15 Character code with 10 Segments

coding schme	affiliation	battle dimention	status	function part1	function part2	function part3	symbol modifier	country code	order of battle	Full code	description
S	*	G	*	I-	--	--	H*	**	*	S * G * I- -- -- H* * * *	INSTALLATION
S	*	G	*	IM	--	--	H*	**	*	S * G * IM -- -- H* * * *	MILITARY MATERIEL FACILITY
S	*	G	*	IM	F-	--	H*	**	*	S * G * IM F- -- H* * * *	NUCLEAR ENERGY
S	*	G	*	IM	FA	--	H*	**	*	S * G * IM FA -- H* * * *	ATOMIC ENERGY REACTOR
S	*	G	*	IM	FP	--	H*	**	*	S * G * IM FP -- H* * * *	NUCLEAR MATERIAL PRODUCTION
S	*	G	*	IM	FP	W-	H*	**	*	S * G * IM FP W- H* * * *	WEAPONS GRADE
S	*	G	*	IM	FS	--	H*	**	*	S * G * IM FS -- H* * * *	NUCLEAR MATERIAL STORAGE
S	*	G	*	IM	A-	--	H*	**	*	S * G * IM A- -- H* * * *	AIRCRAFT PRODUCTION & ASSEMBLY
S	*	G	*	IM	E-	--	H*	**	*	S * G * IM E- -- H* * * *	AMMUNITION AND EXPLOSIVES PRODUCTION
S	*	G	*	IM	G-	--	H*	**	*	S * G * IM G- -- H* * * *	ARMAMENT PRODUCTION
S	*	G	*	IM	V-	--	H*	**	*	S * G * IM V- -- H* * * *	MILITARY VEHICLE PRODUCTION
S	*	G	*	IM	N-	--	H*	**	*	S * G * IM N- -- H* * * *	ENGINEERING EQUIPMENT PRODUCTION
S	*	G	*	IM	NB	--	H*	**	*	S * G * IM NB -- H* * * *	BRIDGE
S	*	G	*	IM	C-	--	H*	**	*	S * G * IM C- -- H* * * *	CHEMICAL & BIOLOGICAL WARFARE PRODUCTIO
S	*	G	*	IM	S-	--	H*	**	*	S * G * IM S- -- H* * * *	SHIP CONSTRUCTION
S	*	G	*	IM	M-	--	H*	**	*	S * G * IM M- -- H* * * *	MISSILE & SPACE SYSTEM PRODUCTION
S	*	G	*	IB	--	--	H*	**	*	S * G * IB -- -- H* * * *	MILITARY BASE/FACILITY
S	*	G	*	IB	A-	--	H*	**	*	S * G * IB A- -- H* * * *	AIRPORT/AIRBASE
S	*	G	*	IB	N-	--	H*	**	*	S * G * IB N- -- H* * * *	SEAPORT/NAVAL BASE

Composite Entity (an entity made up of multiple entities)

An address can be a called attribute of a sales order. An address can sales order related to address entity

Table: Sales Order

OrderNbr	Item	Qty	Price	Amount	Customer	Addresses	Dates
123	DVDPlayer	2	10	20	xyz	BilltoShipto	Order,ship,etc

Table: StudentGrade

id	Student Name	Subject	Year	Grade	highschool Name
id1	john doe	math	2010	10th	FHHS

Composite Entity

```
{
 "firstName": "John",
 "lastName" : "doe",
 "age"      : 26,
 "address"  : {
  "streetAddress": "naist street",
  "city"        : "Nara",
  "postalCode"   : "630-0192"
 },
 "phoneNumbers": [
  {
   "type"  : "iPhone",
   "number": "0123-4567-8888"
  },
  {
   "type"  : "home",
   "number": "0123-4567-8910"
  }
 ]
}
```

Primary Entity
Entity can be without any other entity. For example, Person, Equipment, Facility, Organization, etc.

Composite Entity
This combination of multiple primary entities. For example, Employment History is relationship between Person and Organization, Citizenship is relationship between Person and Country Government
Since Composite entity is a relationship of Primary entity, it should have only attributes which are relationship dependent. For example, Job description, Salary, etc.

Relationship representation formats

JSON

```
{
 "firstName": "John",
 "lastName" : "doe",
 "age"    : 26,
 "address" : [
{
   "streetAddress": "naist street",
   "city"      : "Nara",
   "postalCode"  : "630-0192"
 },
{
   "streetAddress": "main street",
   "city"      : "Rome",
   "postalCode"  : "555-6677"
 },
]}
```

```
{
 "firstName": "Jane",
 "lastName" : "doe",
 "age"    : 24,
 "address" : [
{
   "streetAddress": "naist street",
   "city"      : "Nara",
   "postalCode"  : "630-0192"
 },
{
   "streetAddress": "main street",
   "city"      : "Rome",
   "postalCode"  : "555-6677"
 },
]}
```

SQL Tables

Table: Person

PK	firstName	lastName	age
id1	John	Doe	26
id2	Jane	Doe	24

Table: Person Addresses

PersonPK	Address Pk
id1	id3
id1	id4
id2	id3
id2	id4

Table: address

PK	streetAddress	City	postalCode	PersonID
id3	nait street	Nara	630-0192	id1
id4	main street	Rome	555-6677	id1

Graph

Person and Address Nodes are in one Graph/ Table

PK	firstName	lastName	age
id1	John	Doe	26
id2	Jane	Doe	24
PK	streetAddress	City	postalCode
id3	nait street	Nara	630-0192
id4	main street	Rome	555-6677

Relationship or Edges

Source	Target
id1	id3
id1	id4
id2	id3
id2	id4

"Real world" data has hierarchy / many-to-many relationship
Many people are associated with many addresses
and an address is associated with many people
John Doe and Jane Doe both reside / own two home addresses
A specific address is associated with John Doe, Jane Doe and many previous owners /

Attribute Group

Example of attributes of a tank from product specification

RUSSIAN FEDERATION MAIN BATTLE TANK T-80B T-80U

SYSTEM	SPECIFICATIONS	SYSTEM	SPECIFICATIONS
Alternative designations	-	Armament-Main Gun:	Smoothbore gun
Date of introduction	1976/1987	Caliber, type, name:	125mm 2A46-2
Proliferation	1 and 3	Rate of Fire (rd/min):	6-8
Crew	3	Loader Type:	Autoloader; manual
Combat weight (mt)	44.5/46.0	Ready main gun rounds:	28 carousel
Chassis length overall (m)	6.98/7.01	Stowed rounds:	17
Height overall (m)	2.20/2.22	Elevation (°):	-7 to +14
Width overall (m)	3.60	Fire on Move:	Yes
Ground pressure (kg/cm2)	0.87/.0.92	Armament-Aux Weapon:	Turret-coax to main gun
AUTOMOTIVE		Caliber, type, name:	7.62mm PKT
Engine type (hp)	Gas turbine diesel (multi)	Max eff range-day (m):	1000
Engine type (hp) Upgrade	1000/1250	Max eff range-night (m):	850
Cruising range (km)	370/550 w extra fuel tanks	Fire on move:	Yes
Max road speed (kph)	70	Rate of fire (rd/min):	210
Max off-road speed (kph)	45	Armament-Aux Weapon:	Turret-TC cupola
Average cross-country (kph)	40	Caliber, type, name:	12.7mm
Max swim:	NA	Max aimed range-day (m):	1500
Fording depth (m)	1.8 unprep; 5.0 snorkel	Max eff range-night (m):	800
COMMUNICATIONS		Fire on move:	Yes
Radio	R-173; R-174	Rate of fire (rd/min):	210
External Intercom device	No	ATGM LAUNCHER:	
		Missile name-nomenclature:	AT-11 Refleks
PROTECTION		Launch method:	Gun
Applique armor:	NA/Hull side; track skirts	Missile guidance:	laser
Explosive reactive armor:	Kontakt-5	Launch rate (msl/min):	2-3
Active system:	Arena available	FIRE CONTROL	
Mine clearing:	Roller-plow available	FCS name:	1A332A/1A42
Self-Entrenching blade:	Yes	Thermal: TC-gunner	yes
NBC protection system:	Yes	Main gun stabilization	2E26M/2E42; 2-plane
Smoke equipment:	grenlaunch 4x2; VESS	Infrared	Yes
		Sights w/magnific: day (m)	5000
		Sights w/magnific: nt (m)	1000/2600

Single value vs array of values

Flat/single layer

```
{
 “id”: “1234",
 "firstName": "John",
 "lastName" : "doe",
 "age"      : 26
}
```

Single Attribute array

```
{
 “id”: “1234",
 "firstName": "John",
 "lastName" : "doe",
 "age"      : 26,
 “role"     : [“admin”,”standard”]
}
```

Attribute Set array

```
{
 “id”: “1234",
 "firstName": "John",
 "lastName" : "doe",
 "age"      : 26,
 “role"     : [“admin”,”standard”],
 "address"  : [
{
   "streetAddress":  "naist street",
   "city"        : "Nara",
   "postalCode"   : "630-0192"
 },
{
   "streetAddress":  "main street",
   "city"        : "Rome",
   "postalCode"   : "555-6677"
 },
]}
```

Description is combination of two or more codes

TABLE A-I. Symbol ID code positions and categories.

CODING SCHEME (1) (POSITION 1)	AFFILIATION (1) (POSITION 2)	BATTLE DIMENSION (1) (POSITION 3)	STATUS (1) (POSITION 4)
S - WARFIGHTING G - TACTICAL GRAPHICS W - METOC I - INTELLIGENCE M - MAPPING (reserved - under development) O - Military Operations Other Than War (MOOTW)	P - PENDING U - UNKNOWN A - ASSUMED FRIEND F - FRIEND N - NEUTRAL S - SUSPECT H - HOSTILE J - JOKER K - FAKER O - NONE SPECIFIED	P - SPACE A - AIR G - GROUND S - SEA SURFACE U - SEA SUBSURFACE E - SOF X - OTHER (No frame)	A - ANTICIPATED /PLANNED P - PRESENT
FUNCTION ID (6) (POSITION 5 - 10)	**SYMBOL MODIFIER (2) (POSITION 11, 12)**	**COUNTRY CODE (2) (POSITION 13, 14)**	**ORDER OF BATTLE (1) (POSITION 15)**
See tables A-III through A-IX for specific values.	See table A-II for specific values	See FIPS Pub series 10	A - AIR OB E - ELECTRONIC OB C - CIVILIAN OB G - GROUND OB N - MARITIME OB S - STRATEGIC FORCE RELATED

minE Alias	TACREP ALIAS (delete?)
ASSUMED FRIEND	AIR ASSUMED FRIEND
ASSUMED FRIEND	LAND ASSUMED FRIEND
ASSUMED FRIEND	SPACE ASSUMED FRIEND
ASSUMED FRIEND	SUBSURFACE ASSUMED FRIEND
ASSUMED FRIEND	SURFACE ASSUMED FRIEND
ASSUMED FRIEND	UNKNOWN ASSUMED FRIEND
FAKER (EXER HOSTILE)	UNKNOWN FAKER
FAKER (EXER HOSTILE)	AIR FAKER
FAKER (EXER HOSTILE)	LAND FAKER
FAKER (EXER HOSTILE)	SPACE FAKER
FAKER (EXER HOSTILE)	SUBSURFACE FAKER
FAKER (EXER HOSTILE)	SURFACE FAKER
FRIEND	AIR FRIEND
FRIEND	LAND FRIEND
FRIEND	SPACE FRIEND
FRIEND	SUBSURFACE FRIEND
FRIEND	SURFACE FRIEND
FRIEND	UNKNOWN FRIEND

Miscellaneous Complexity

(U) Machine Readable Schema Mapping Complex Scenarios

Table is CUI

Multiple source attributes maps to one target attribute

Target API	Source API
$Person.Full Name	$Person.firstName + " " + $Person.lastName

Array of source attributes maps to multiple single value target attribute. If source schema does not have specific attribute, then, there would be data loss.

Target API	Source API
$Person.MobilePhone	$Person.phoneNumbers[].type where type='Mobile' + " " + $Person.phoneNumbers[].number
$Person.HomePhone	$Person.phoneNumbers[].type where type='Home' + " " + $Person.phoneNumbers[].number
?	$Person.phoneNumbers[].type where type='Work' + " " + $Person.phoneNumbers[].number
?	$Person.phoneNumbers[].type where type='Fax' + " " + $Person.phoneNumbers[].number

Array of source attributes maps to multiple single target attribute. Only one can be mapped or imported. There would be data loss if source has multiples

Target API	Source API
$Organization.Contact	$.Organization.pointsOfContact[]

$Person.address.streetName	$Person.address[].streetAddress
$Person.address.city	$Person.address[].city
$Person.address.zipCode	$Person.address[].postalCode

Segment of a source attributes maps to a target attribute

Target API	Source API
$Person.firstName	$Person.Full Name (substring first name)
$Person.lastName	$Person.Full Name (substring last name)

Table is CUI

Table is CUI

A source attributes code must be transformed to full name or vise versa. Value map / normalization table should be used.

Target API	Source API
$Organization.CountryName	$.Organization.CountryCode

CountryCode	CountryName
CA	CANADA
CB	CAMBODIA
UK	UNITED KINGDOM
UN	FACILITY OWNED BY OR UNDER THE CONTROL OF THE UNITED NATIONS
UP	UKRAINE
UR	SOVIET UNION (FOR HISTORICAL PURPOSES)
US	UNITED STATES
UU	UNDESIGNATED SOVEREIGNTY

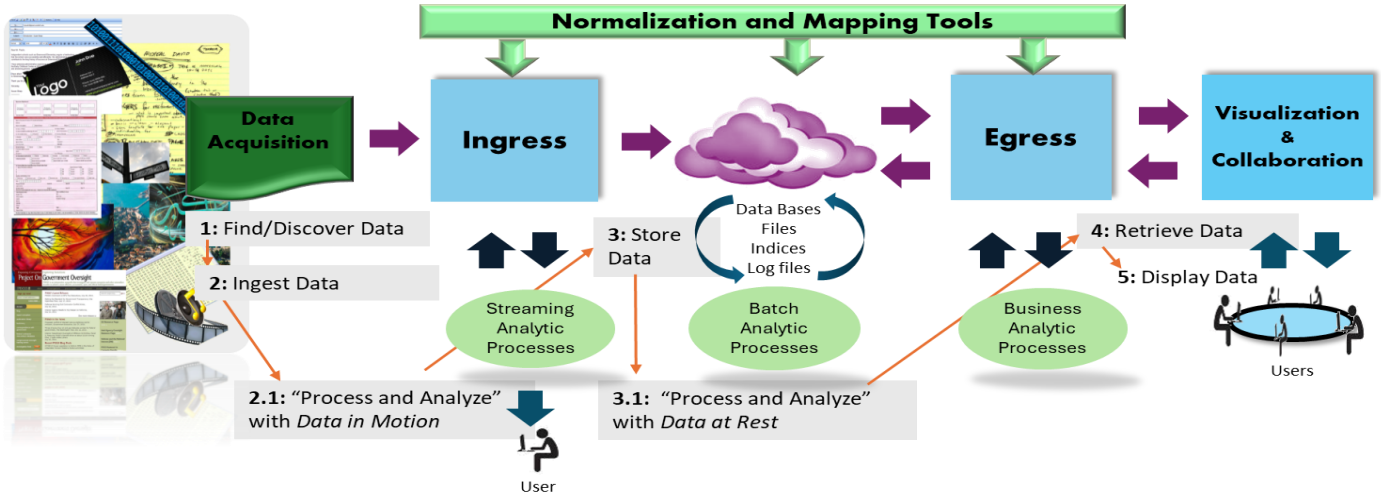
Table is CUI

(U) Other complexities

- Units of measures assumed but not specified. For example, Velocity 60 mean 60 m/hr and m/hr mean miles per hour.
- Date and phone format differences. 111.222.3333 vs (111) 222-3333
- Data type differences. Text vs. integer or Number.
- Same attribute/value name but different meaning. Port (computer) vs Port (ship docking port). Tank (weapon) vs. Tank (water tank)
- Different attribute / value name but exactly the same meaning. Last Name vs. Family Name.
- Different level in class hierarchy. Automobile vs car vs truck. Automobile is at high level or more general.

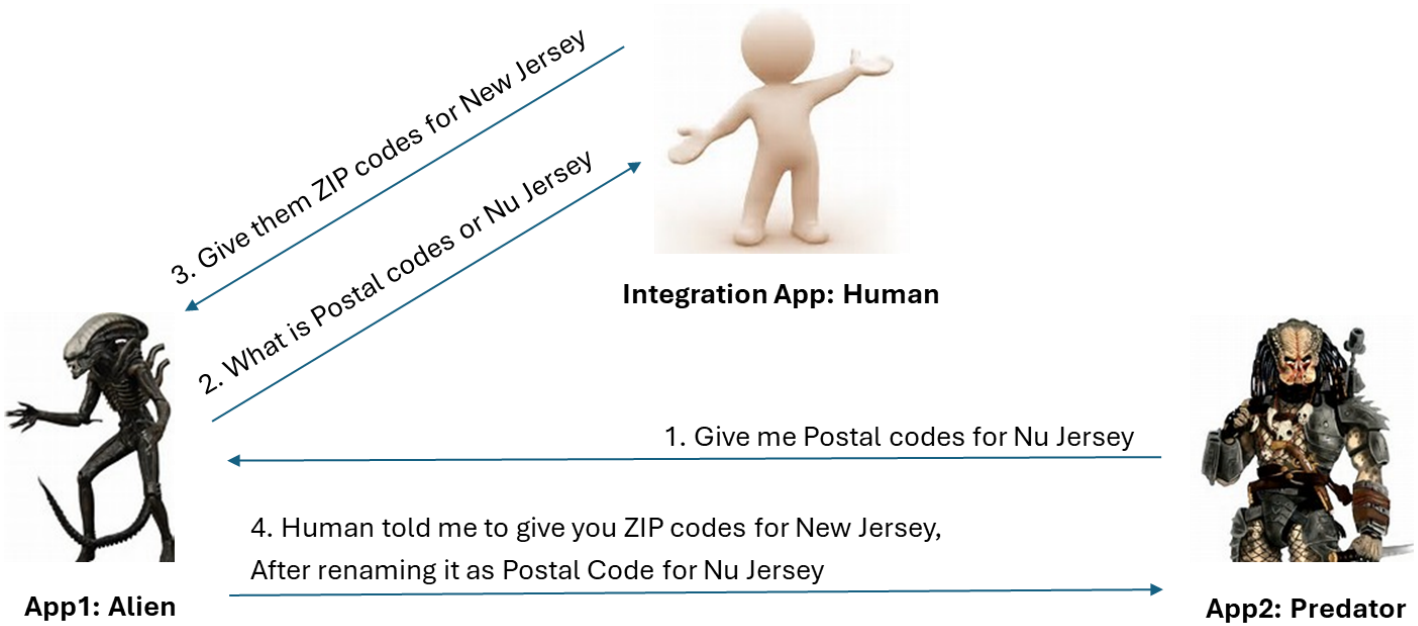
Mapping needs

Multiple sources are consolidated in one place (data warehouse)



Find/Discover Data – Source data has data can be in SQL, XML, JSON, Key Value pair. Data models' definition can be in a variety of formats.
Ingest Data – Data model is needed to ingest specific data in specific format.
Store Data – Data can be stored in SQL, XML, JSON, Key Value pairs. Data can be stored in its original data model or in a common model. Normalization means change values to a common model/Classes/Ontology.
Retrieve Data – If data is stored in its original data models, then, it is necessary to know the original data models. If data is stored in one common data model and normalized, then only one data model is needed to retrieve the data.
Display Data – Same dilemma as Retrieving data. If data is not in one common data model, then a user would have to make sense out of multiple data models displayed on screen. Also, any analytics would have to understand multiple data models.

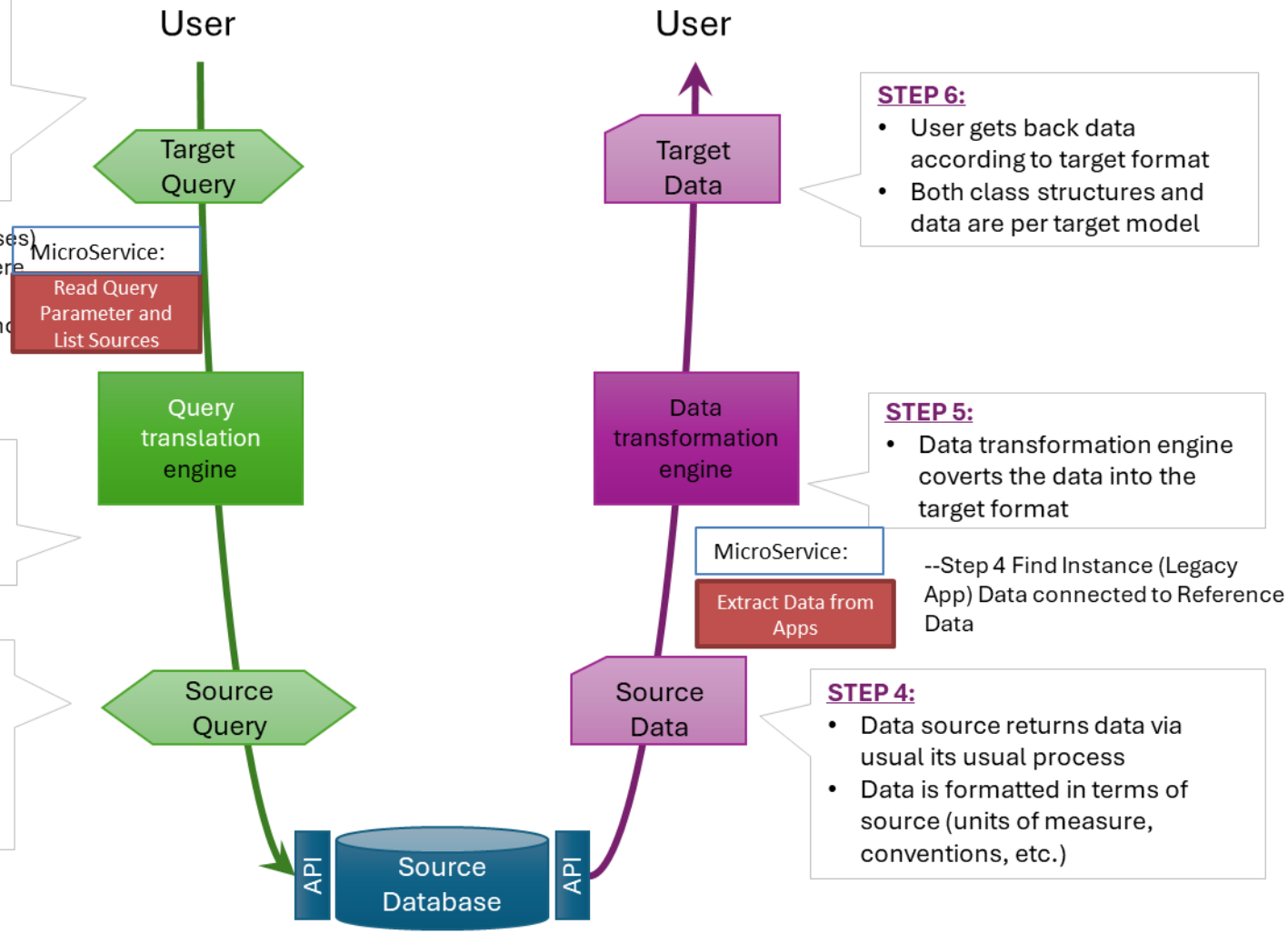
Integration between two applications



Integration as a Data Solution is needed because:
Two Apps cannot be at the same location / hardware
Two Apps will remain in their current state since there is no funding to upgrade
Two Apps are under different organizations
In short, you are stuck with what you have

Cross model query (federated query)

STEP 1:
- User issues a query on the target model
- E.g. CCO
- Target model can be data model or ontology
- User does not have to know source models



STEP 2:
- Translate the query into source model queries

STEP 3:
- Source query is issued against the native source data
- E.g. TED
- Can be data model or ontology

STEP 4:
- Data source returns data via usual its usual process
- Data is formatted in terms of source (units of measure, conventions, etc.)

STEP 5:
- Data transformation engine coverts the data into the target format

STEP 6:
- User gets back data according to target format
- Both class structures and data are per target model

Model Mapping Automation

Access database for schema/class mapping for full word match, partial string match, and Levenstein algorithm
Python Levenshtein distance algorithm for matching percentage
Class hierarchy and manual map of top-level classes for focused automatic search
Mapping of multisegmented code of 2525B, 2525C, and 2525D for focused automatic search
Mapping of MIM and 2525C for focused automatic search MIM and 2525C
Example of top-level manual match for focused automatic search
Class/taxonomy match to discover schema match
Partial words match examples
Class map between specific to specific, specific to general using parent class

Python Levenshtein distance algorithm for matching percentage

MTF List	MTF Value	mim Class	mimATTRIBUTE	Matching Ratio
AIRSPACE CONTROL MEANS IDENTIFICATION	transition altitude	mil.mip.mimbase.concept.object.feature.controlfeature.AirspaceControlMeans	transition_altitude_dimension	79
ENEMY INDIVIDUAL DATA	name	mil.mip.mimbase.concept.object.actor.person.Person	name	100
ENEMY INDIVIDUAL DATA	nationality	mil.mip.mimbase.concept.object.actor.person.Person	nationality	100
ENEMY UNIT	entity operational status	mil.mip.mimbase.concept.object.actor.organisation.unit.Unit	operational_status_code	83
EVENT DATA	event description	mil.mip.mimbase.concept.action.Event	description_text	85
EVENT DESCRIPTION	event description	mil.mip.mimbase.concept.action.Event	description_text	85
INDIVIDUAL DATA	nationality	mil.mip.mimbase.concept.object.actor.person.Person	nationality	100
MESSAGE IDENTIFICATION	originator	mil.mip.mimbase.concept.object.informationresource.InformationResource	originator_name	80
PERSONAL DATA	nationality	mil.mip.mimbase.concept.object.actor.person.Person	nationality	100
REFERENCE	originator	mil.mip.mimbase.concept.object.informationresource.InformationResource	originator_name	80
SUPPLEMENTAL FACILITY DATA	axis orientation	mil.mip.mimbase.concept.object.facility.Facility	orientation	81
VESSEL DATA	maximum speed, knots	mil.mip.mimbase.concept.object.materiel.equipment.vessel.Vessel	maximum_speed	81

Levenstein algorithm results need manual check for less than 100% matching ratio

Mapping of MIM and 2525C for focused automatic search MIM and 2525C
Example of top-level manual match for focused automatic search

	2525B SegmentName
position1	Coding Scheme
position2	Affiliation
position3	Battle Dimention
position4	Status
position5-6	Function Part1
position7_8	Function Part2
position9_10	Function Part3
position11_12	Symbol Modifier
position13_14	Country Code
position15	Order of Battle

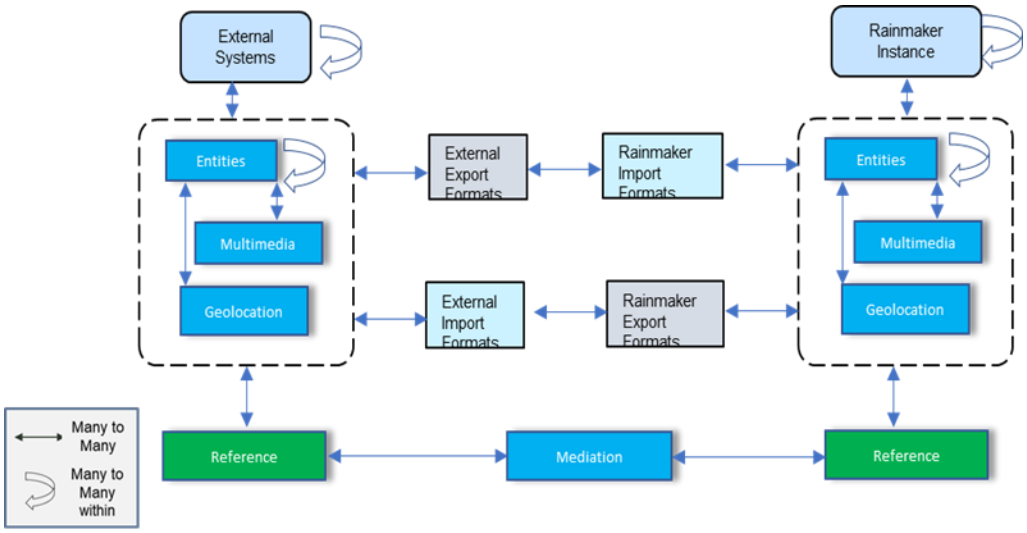
	2525C SegmentName
position1	Coding Scheme
position2	Affiliation
position3	Battle Dimention
position4	Status
position56	Function Part1
position78	Function Part2
position910	Function Part3
position11_12	Symbol Modifier
position13_14	Country Code
position15	Order of Battle

	2525D SegmentName	
position1_2	Symbol Set	
postion3_4	Entity	(Digits 1and 2)
position5_6	Entity Type	(Digits 3 and 4)
position7_8	Entity Subtype	(Digits 5 and 6)
position9_10	MOD 1	
position10_11	MOD 2	

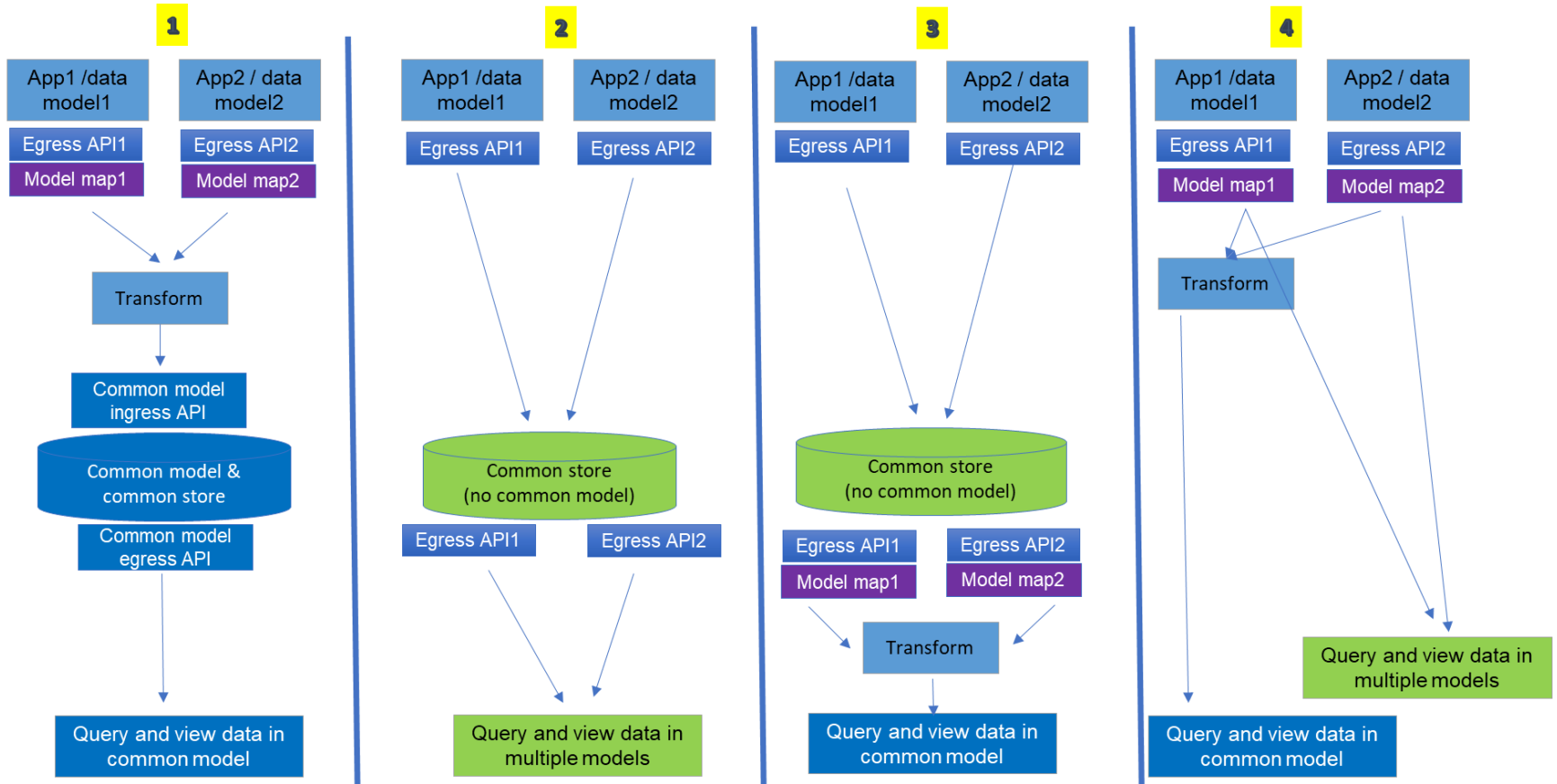
L1	L2	L3	L4	L5	L6	L7	S1	S2	S3	S4
mim	concept	object	materiel	equipment	landvehicle	MilitaryTypeUtilityVehicle	Warfighting Symbology	Ground Track	Ground Track Equipment	Ground Vehicle
mim	concept	object	materiel	equipment	vessel	SurfaceVessel	Warfighting Symbology	Sea Surface Track		
mim	concept	object	materiel	equipment	aircraft	RotaryWingAircraft	Warfighting Symbology	Air Track	Military	Rotary Wing
mim	concept	object	materiel	equipment	aircraft	FixedWingAircraft	Warfighting Symbology	Air Track	Military	Fixed Wing
mim	concept	object	materiel	equipment	weapon		Warfighting Symbology	Ground Track	Ground Track Equipment	Weapon
mim	concept	object	actor	organisation	unit		Warfighting Symbology	Special Operations Forces (SOF) Unit	SOF Unit Aviation	
mim	concept	object	actor	organisation	unit		Warfighting Symbology	Ground Track	Unit	
mim	concept	object	actor	organisation	unit		Warfighting Symbology	Special Operations Forces (SOF) Unit	SOF Unit SOF Unit Naval	
mim	concept	object	feature	meteorologicfeature	Atmosphere		METOC	Atmospheric	Pressure Systems	
mim	concept	action	OtherEvent				Tactical Graphics	Tasks		
mim	concept	action	OtherEvent				Stability Operations	OPERATIONS		
mim	concept	action	OtherEvent				Stability Operations	Violent Activities		
mim	concept	action	OtherEvent				Stability Operations	Items		

Data source traceability

A Hardware (Server) contains Software
A Hardware (Server) is on a Network
A Hardware (Server) has base Operating System
A Network is Hardware
A Network is Software
An Operating System (OS) is a Software
A Container / VM has an Operating System
A Container / VM has Software Applications
A Container/ VM has Data Stores (Database)



Data fabric vs Data mesh



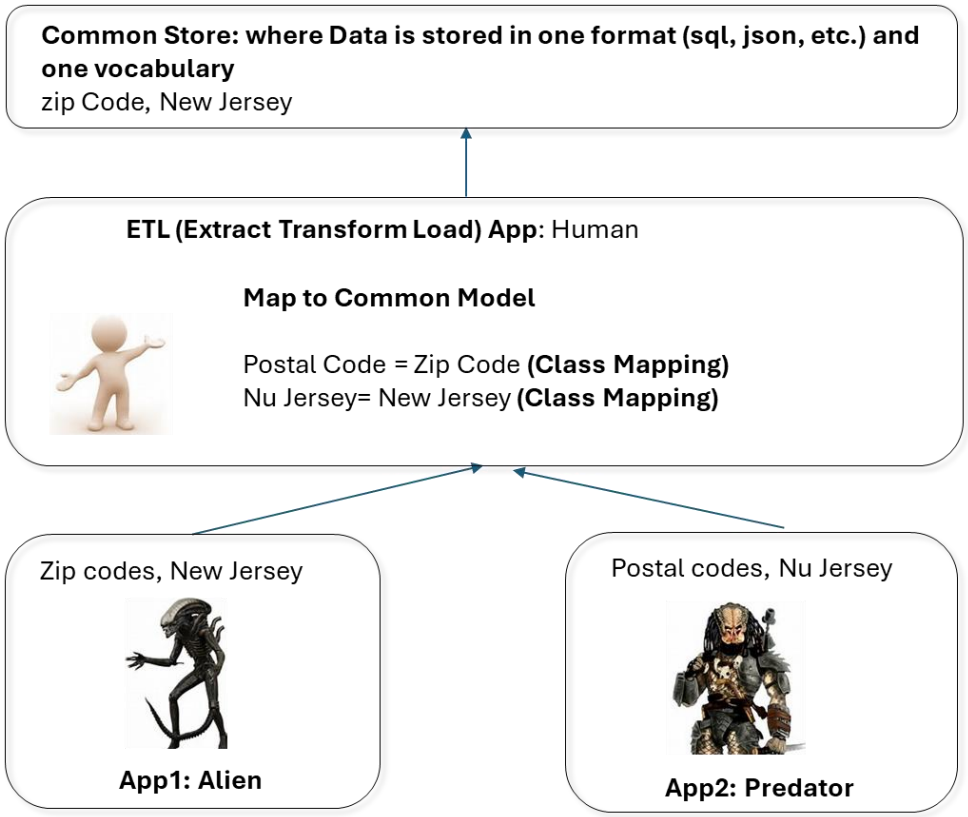
- **Simple query to pull data from one model / store**
- **End user interacts with one model**

- **Complex query to pull data from multiple models.**
- **End user interacts with multiple model**

- **Complex query to pull data from multiple models and transform**
- **End user interacts with one model**

- **Complex query to pull data from multiple models and transform to one model**
- **End user interacts with one model**

Data in Common Model and Common Store



Common Store (data warehouse) is a Data Solution when:
Two Apps are willing to provide data
Data from multiple apps are extracted, transformed and loaded in data warehouse