Capitolul 10. Modele pentru standardizarea datelor

Intocmit: *Paulina Mitrea – responsabil modele standardizare date*

Definiție: Un model de date standard sau un standard industrial pentru modele de date (ISDM-Industrial Standard Data Model) este un model de date care este aplicat pe scară largă în unele domenii/industrii și împărtăşit într-un anumit grad între actorii domeniilor/industriilor respective.

Geneza:

Aceste modele de standardizare date sunt în general definite de către organismele de standardizare, furnizorii de baze de date sau de furnizorii de sisteme de operare.

Motivație/Utilitate:

Standardele pentru modele de date permit schimbul de informații mai ușor și mai rapid, deoarece, prin standardizare, organizații eterogene ajung să aibă un vocabular standard și semantică, cât şi format și standarde de calitate pre-negociate pentru schimbul de date sau interoperabilitatea datelor.

Standardizarea datelor afectează arhitectura software, deoarece soluțiile care se abat de la standard pot cauza probleme de partajare a datelor cât și de altă natură, dacă datele nu respectă standardul.

Modelele standard cele mai eficiente s-au dezvoltat în domeniul administrativ&guvernamemntal, în industria bancară, de asigurări, farmaceutică și auto, pentru a reflecta standardele stricte aplicate colectării informațiilor cetățenilor/clienților, cât şi pentru datele privind confidențialitatea clienților, siguranța consumatorilor şi/sau fabricarea „just in time”.

În mod obișnuit, se utilizează modelul relațional foarte popular şi eficient al gestionării bazelor de date, dar unii mai folosesc si modelul ierarhic, mai ales pentru datele utilizate în producție sau pentru cele mandatate de la nivel guvernamental (de exemplu codurile DIN - ***Deutsches Institut für Normung*** specificate de la nivelul guvernamental pentru Germania).

Deși formatul standardului poate avea compromisuri de implementare, obiectivele subiacente ale acestor standarde sunt de **a facilita schimbul de date (interoperabilitatea datelor).**

Exemple de Modele de Date Standard agreate la nivel mondial

- ISO 10303 CAE Data Exchange Standard - include propriul limbaj de modelare a datelor denumit EXPRESS
- ISO 15926 - Process Plants including Oil and Gas facilities Life-Cycle data
- IDEAS Group Ontologie – cu o fundamentare convenită de departamentele de apărare din Australia, Canada, Franța, Suedia, Marea Britanie și SUA

- Common Education Data Standards (CEDS) - dicționar de date susținut de guvernul SUA care este utilizat pe scară largă în Statele Unite ale Americii
- SIF - specificație de interoperabilitate folosită ca model de date standard în Australia, Marea Britanie şi SUA

Cele mai eficiente şi actuale standarde în domeniul administrativ/guvernamental:

- **UBL**
- **ISA**
- **UNCE/CEFACT**
- **SEMIC**

Standardul UBL

Standardul UBL este un set coordonat de componente gramaticale XML (eXtensible Markup Language) care permite identificarea fără echivoc a datelor si informațiilor din documentele care sunt interschimbate într-un anumit context/domeniu.

Astfel, UBL adoptă o abordare bazată pe modelul de date pentru a proiecta documente standardizate care pot fi exprimate în XML. De fapt toate tipurile de documente UBL sunt alcătuite din componente ale unui model comun de date numit Bibliotecă Comună (Common Library).

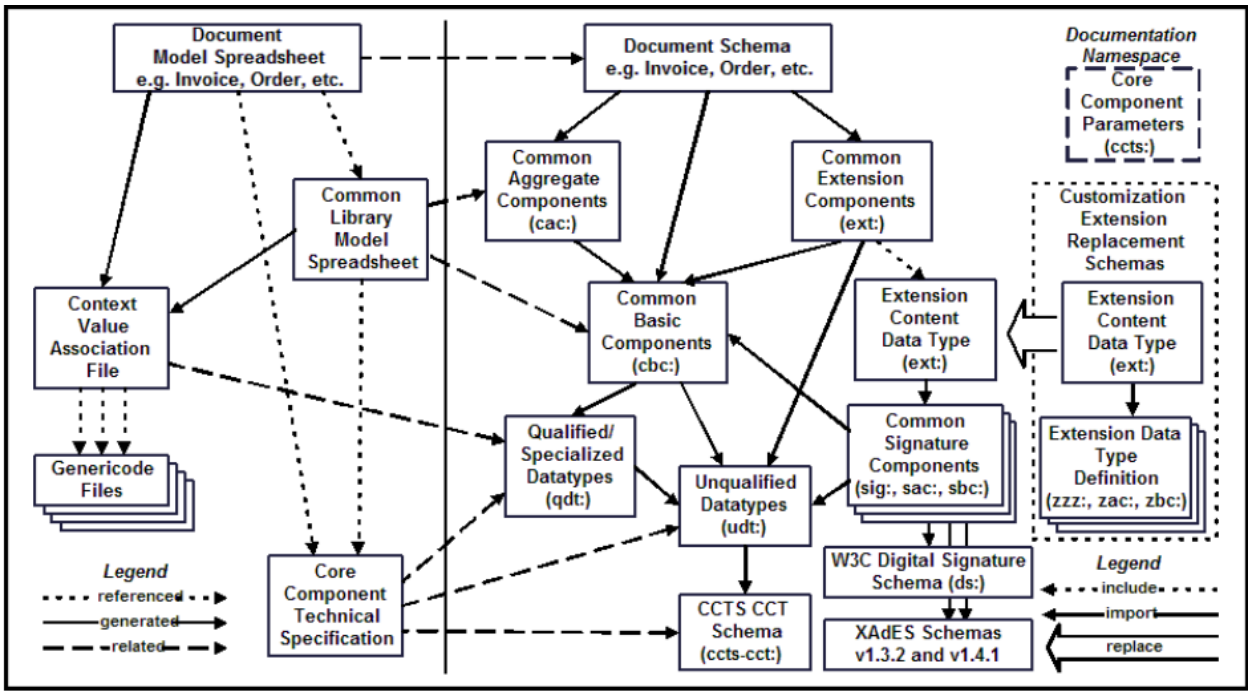
În utilizarea sa normală, termenul “model de date” înseamnă un model care descrie datele necesare și create de procesele de business. Un model de date ar trebui să fie conceptual (nu tehnologic) și trebuie să se concentreze pe semantica (semnificațiile și relațiile) componentelor pe care le descrie.

Utilizarea unui model de date semantic pentru a descrie, spre exemplu, Biblioteca Comună UBL, s-a dovedit a fi benefică în mai multe moduri:

1. Experții în business-logic pot lucra pe semantica proceselor de business UBL și a cerințelor lor de date fără a fi nevoie de fapt să cunoască tehnologia XML. Această metodologie se bazează pe principiile analizei și modelării datelor și utilizează, pentru notația sa, specificația tehnică a componentelor principale ebXML (electronic business XML definit in standardul ISO 15000-5).
2. Modelul Comun de Date (Common Data Model) asigură ca semantica comună a componentelor să nu fie pierdută atunci când aceste componente sunt reutilizate în contexte diferite.
3. Este posibil să se genereze automat, din modelele de date UBL, nu numai reprezentările XML ale tipurilor de documente - cum ar fi XML Schema și RelaxNG, dar și reprezentări non-XML -cum ar fi ASN.1. Acest lucru garantează modul de utilizare a standardului UBL privitor la posibilele modificări în notații și sintaxe.
4. Deoarece modelul de date semantice UBL este similar cu un model de date relațional, unele comunități de utilizatori folosesc de asemenea acest model de date pentru propriile reprezentări interne de date.

UBL însoțește modelul de date semantic cu descrieri textuale detaliate, explicații și exemple pentru fiecare dintre funcțiile și mesajele sale specificate, precum și reguli clare pentru extensie și o serie de subseturi și resurse de instrumente, cum ar fi dicționare de localizare în limbile chineză, daneză, germană, Italiană, japoneză, coreeană, spaniolă și turcă. Acest lucru permite comunităților să creeze extensii explicite, profilate local pentru contextul lor de utilizare.

Pentru a pune în discuție o anumită realitate, fundamentul modelului de date semantic UBL poate fi evidențiat folosind reprezentarea realizării modelului de date UBL, conform standardului OASIS UBL, prin diagrama de mai jos.

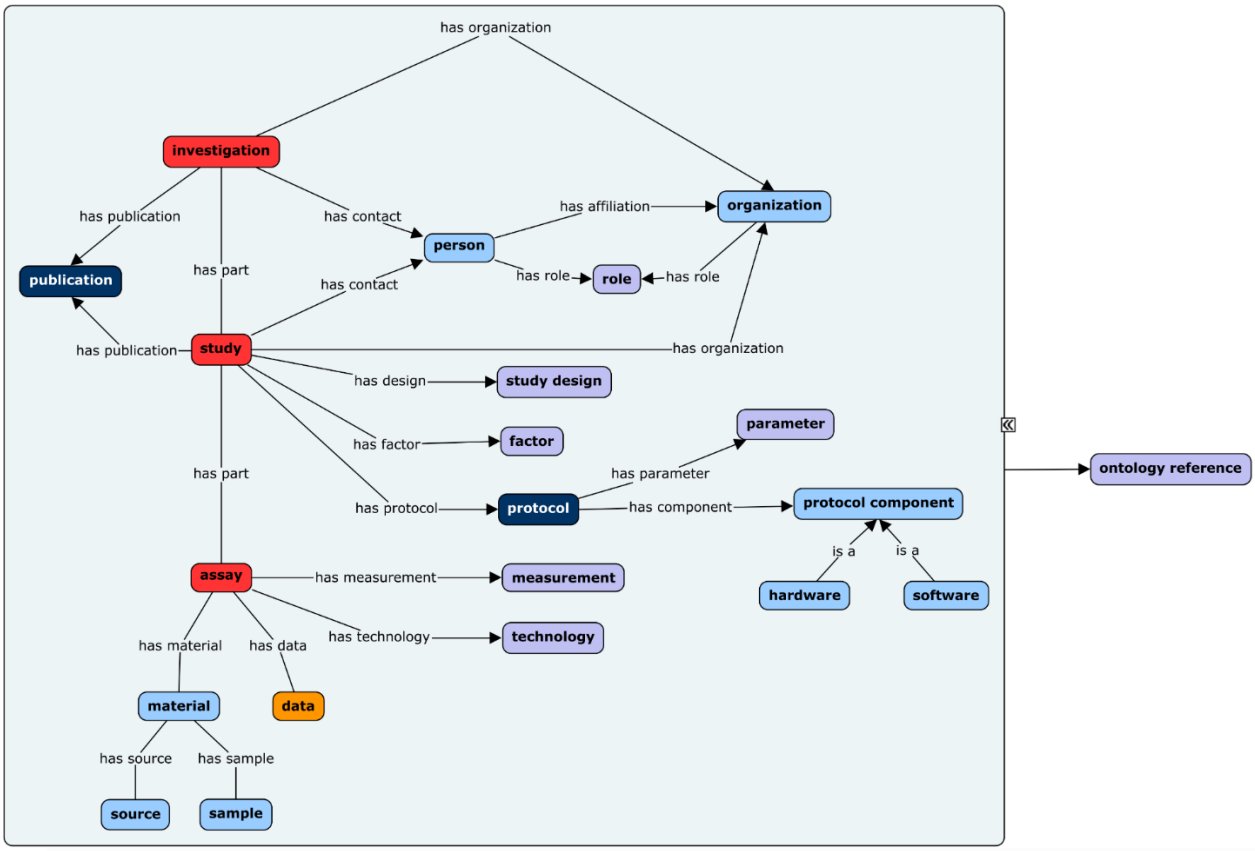


Pentru o analiză mai detaliată, sunt furnizate versiuni UML, HTML și spreadsheet (excel).

Standardul ISA

Specificația ISA definește un model abstract al cadrului metadatelor, metadatele constând de fapt din date despre date. Modelul abstract ISA a fost implementat în două specificații de format - ISA-Tab și ISA-JSON, ambele având instrumente și servicii asociate acestora. Specificațiile formatelor sunt, de asemenea, disponibile pentru instrumente suplimentare menite a se folosi de avantajul oferit de conținutul ISA formatat.

Harta de concepte de mai jos prezintă obiectele / entitățile ISA și relația dintre ele:



[http://isa-specs.readthedocs.io/en/latest/isamodel.html]

Modelul ISA este alcătuit din trei entități de bază pentru captarea metadatelor experimentale:

- `Investigation / Investigatie`
- `Study/ Studiu`
- `Assay/Verificare(test)`

O entitate Investigation conține toate informațiile necesare pentru a înțelege obiectivele generale și mijloacele utilizate într-o secvență de evenimente; paşii secvenței de evenimente sunt descrişi într-un Study, urmat de Assay. Pentru fiecare Investigation pot exista unul sau mai multe studii Study associate acestuia; pentru fiecare Study pot exista una sau mai multe entități Assay.

Cele trei entități se detaliază dupa cum urmează:

Investigation

O entitate ***Investigation*** este destinată pentru:

1. Inregistrarea metadatelor referitoare la analiza unei secvențe de evenimente
2. Conectarea obiectelor aferente studiului asociat unei investigații (acest lucru devenind necesar doar atunci când două sau mai multe obiecte de studiu trebuie grupate)

Entitatea ***Investigation*** este utilizată pentru a înregistra metadatele referitoare la descrierea contextului investigației, cum ar fi titlul și descrierea investigației, precum și cele referitoare la persoanele conexe și documente auxiliare (inclusiv publicații, etc). Elementele de studiu și de analiză sunt grupate în cadrul unei investigații pentru a înregistra alte metadate în contextele relevante.

Entitatea Investigation trebuie să înregistreze următoarele:

Denumire Proprietate	Tip de Data	Descriere
Identificator	Sir (string)	Un identificator sau un număr de acces furnizat spre exemplu de un depozit, care trebuie să fie unic la nivel local.
Titlu	Sir (string)	Un nume concis dat pentru identificarea ***Investigației***.
Descriere	Sir (string)	O descriere textuală a ***Investigației.***
Data depunere raport ***Investigatie***	Representare conform ISO8601 a datei calendaristice	Data la care Investigatia a fost raportată în vederea înregistrării/arhivării.
Data diseminării publice a raportului de ***Investigatie***	Representare conform ISO8601 a datei calendaristice	Data la care Investigatia este furnizată ca informație de interes public
Documentatie/Anexe ***Investigatie***	Lista Documentatiei/Anexelor raportului de ***Investigatie***	O listă de documente referitoare la ***Investigatie***.
Contacte	Lista de Contacte	O lista de contacte utile legate de ***Investigatie***

Study (Studiu)

Un ***Study*** (studiu) este un concept central care conține informații despre subiectul studiat, caracteristicile acestuia și tratamentele/procedurile aplicate.

Un studiu conține informații contextualizante pentru unul sau mai multe analize. Metadatele despre proiectul studiului, factorii de studiu utilizați și protocoalele de studiu sunt înregistrate în obiectele studiului, precum și informații similare investigației, inclusiv titlul și descrierea studiului, precum și publicații legate de acesta.

Assay (Verificare/Test)

Un ***Assay*** reprezintă un test efectuat fie pe un material luat de la un subiect, fie pe un subiect inițial integral, care produce măsurători calitative sau cantitative.

Un ***Assay*** poate grupa descrieri provenite din eşantioane procesate în teste conexe. Fiecare test urmărește în mod obișnuit pașii unui flux de lucru experimental specific, descris de un anumit protocol.

Standardul de date UNCE/ CEFACT

UN / CEFACT a creat Specificația Tehnică a Componentelor Principale (CCTS) pentru a facilita dezvoltarea modelelor de date într-un mod structurat. Modelele de date ar trebui să fie armonizate *la nivelul unui întreg* domeniu și *ar trebui* să asigure interoperabilitatea între companiile disparate și guverne în scenariile business-to-business (B2B) și de la business-to-government (B2G).

In conformitate cu acest standard, un model de date implică structuri semantice care pot fi utilizate pentru a crea profiluri de mesaje fără a trebui să ia în considerare sintaxa.

Problematica specifică

Fără un model de date care să conțină semantica standardizată a unui proces sau document, analiștii dintr-un anumit domeniu nu pot crea cu ușurință convergență. Atunci când se analizează cerințele documentului privind sistemele disparate, trebuie să se definească și să se stocheze un mijloc de înregistrare a fiecărui element cât și relația acestuia cu alte elemente, astfel încât să poată fi ușor recuperate pentru a evita redundanțele si ambiguitățile.

Atâta timp cât un model de date se referă la setul de date incluse într-un document (sau structura de stocare date) și la structura și metadatele conform cărora sunt organizate aceste date, modelul poate fi unul de tip ierarhic sau de referință, în care toate elementele de date ale documentului (sau ale structurii de socare/memorare) sunt structurate și armonizate. Aceasta înseamnă că fiecare element de document este contextualizat și etichetat astfel încât poziția sa în document este cunoscută, precum și poziția sa în ierarhie. Fiecare element este moștenit dintr-o componentă de bază, similar cu modul în care clasele se moștenesc pornindu-se de la o clasă de bază. Totuși, conform CCTS, entitatea „copil”

este un subset creat în baza restricțiilor/constrângerilor elementelor din clasa „părinte” (în paradigma modelului ierarhic).

Exemplu de utilizare la nivel European: Conform Recomandării 18 a UNCE (United Nations Economic Commission for Europe), Grupul de Lucru pentru Procesul de Afaceri al ONU/CEFACT a definit lanțul internațional de aprovizionare în termeni simpli de ***cumpărare, livrare și plată***. Au fost multe inițiative care să garanteze că această abordare poate fi conținută într-un model de date de referință și că toate elementele din cadrul acestuia pot fi conectate sau contextualizate la nivelul unuia dintre acești termeni.

Două astfel de modele ierarhice, care asigură structuri compatibile cu această abordare, sunt:

- WCO Data Model - Modelul de date WCO este un set de cerințe de date combinate cu atenție, care se sprijină reciproc și care sunt actualizate în mod regulat pentru a răspunde nevoilor procedurale și juridice ale organizațiilor
- UN CCL - Biblioteca principală a componentelor ONU (CCL) este o bibliotecă de semantică într-un model de date care este armonizat, auditat și publicat de către UN / CEFACT

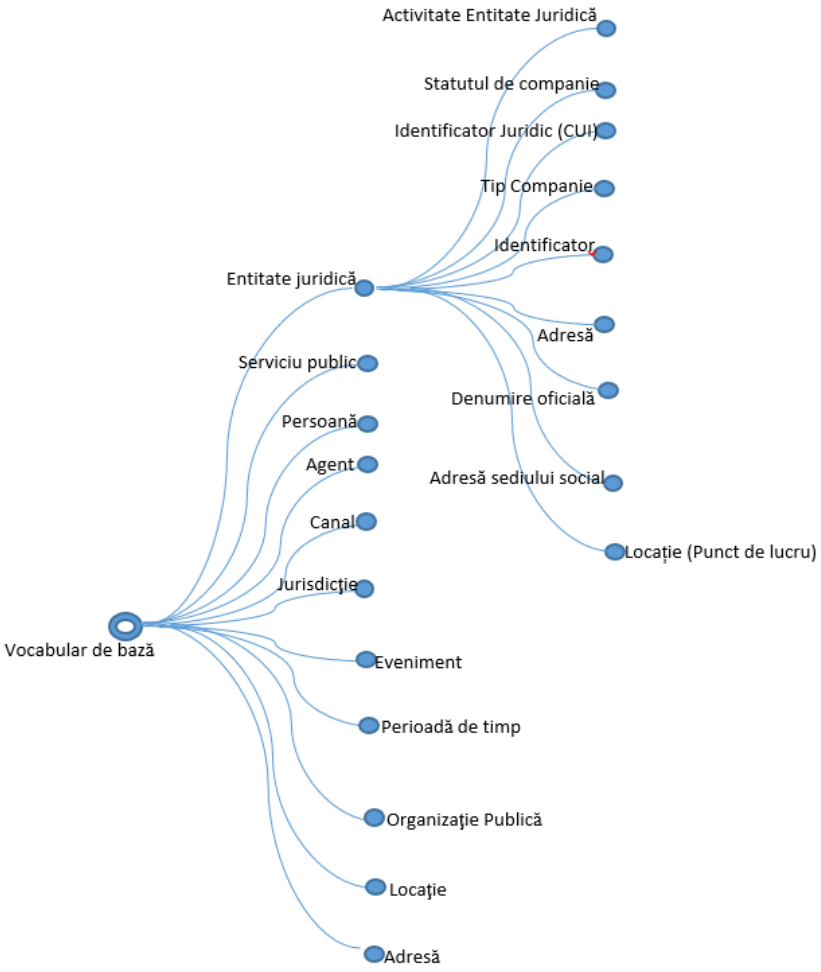
Ambele modele utilizează UN/CEFACT CCTS (Specificația Tehnică a Componentelor Principale), care asigură că acestea sunt interoperabile și pot fi cartografiate la nivel semantic. Odată ce semantica este convenită, este mai ușor să se furnizeze un standard de document utilizându-se o sintaxă specifică. Prin urmare, este o condiție prealabilă ca standardele de documente să fie bazate pe modele robuste. Acest

lucru este în contrast cu standardele anterioare ale documentelor, cum ar fi UN EDIFACT, în care nu fiecare document face parte dintr-un model de date ierarhic.

Standardul SEMIC

Modelul de standardizare SEMIC (Semantic Interoperability Community) este generat în urma inițiativei Comisiei Europene de a îmbunătăți interoperabilitatea semantică a sistemelor interconectate de e-Government.

Un exemplu de aplicabilitate a modelelor recomandate în SEMIC Core Data Model Mapping Director, la datele administrative/guvernamentale de nivel general, este redat în cele ce urmeaza:



Modele de Standardizare Date pentru Arhitecturi Bazate pe Microservicii în domeniul administrativ/guvernamental

Arhitectura bazată pe microservicii structurează și împarte aplicația într-un set de servicii slab cuplate, fiecare serviciu implementând un set de funcționalități similare, din același context.

Un microserviciu este menit sa fie cât de atomic posibil, astfel încat să aibă o amprentă minimă, să aibă un context limitat cât mai bine definit şi mai redus, să reprezinta o arie cât mai îngustă de interes şi să fie capabil să pornească şi să se oprească rapid.

Datele deținute de fiecare microserviciu sunt private și pot fi accesate doar prin API-ul expus de către microserviciul respectiv. Această încapsulare a datei asigură faptul că microserviciile sunt slab cuplate și pot evolua independent unul față de altul. Ba mai mult, diferite microservicii pot folosi diferite tipuri de

baze de date, precum SQL sau NoSQL (de amintit: Azure CosmosDB, MongoDB, PostgreSQL, Cassandra etc), această abordare fiind denumita persistență poliglotă.

Pentru a implementa cu succes o arhitectură bazata pe microservicii sunt de urmat câteva bune practici:

- Bază de date separată per microserviciu
- Bazarea pe contracte între servicii
- Publicare date în containere
- Tratarea serverelor ca fiind volatile.

Pe lângă respectarea acestor bune practici, mai sunt diferite tehnici și design pattern-uri (şabloane de design) care pot fi urmate în implementarea microserviciilor.

O astfel de teoremă de urmat ar fi *teorema CAP* (numită și teorema Brewer, după Eric Brewer, cel care a formulat-o), care spune că pentru sisteme distribuite nu este posibil să se ofere mai mult de două din următoarele trei opțiuni:

- Consistență
- Disponibilitate (Availability)
- Toleranța partiționata (partition tolerance)

Aceasta înseamnă că, în momentul în care apar probleme în comunicare între diferite părți ale clusterului de microservicii, trebuie să se aleagă între consistență și disponibilitate, pentru că există doar două opțiuni: fie propagi eroarea pentru a o semnala, dar atunci nu mai ai parte de disponibilitate, fie prezinți utilizatorului date cache-uite, dar atunci renunți la consistență. În cazul sistemelor de baze de date tradiționale (ACID) se preferă consistența în fața disponibilității. În cazul sistemelor de microservicii, de cele mai multe ori se preferă disponibilitatea sistemului și nu consistența, de aceea mai sunt numite și BASE (Basically Available, Soft state, Eventually consistency).

CQRS - *Command Query Responsibility Segregation* - este un şablon de proiectare (design pattern) foarte des întâlnit, fiind de fapt o extindere a ideii lui Bertrand Meyer - CQS, conform căreia metodele de comandă realizează acțiuni dar nu returnează nimic, metodele de interogare returnează valori dar nu acționează asupra sistemului. În CQRS, cererile de scriere date (adică comenzile) și cererile de citire date (adică interogările) sunt separate în două modele diferite. Modelul de scriere acceptă comenzi și acționează asupra sistemului, modificând datele, pe când modelul de citire acceptă doar interogări și returneaza date către interfața utilizator, fără nicio modificare. Acest model se modifică doar în momentul în care se modifică și primul. De exemplu, într-o aplicație de tipul eCommerce adaugarea unei noi comenzi sau produs nou se poate salva într-o bază de date relațională tipică, pe cand dacă se doreşte căutarea de produse similare, aceasta se poate realiza cu ajutorul unui motor cu arbore. CQRS foloseste de fapt *Materialized View Pattern*, în care generezi în avans (denormalizezi datele şi le pregăteşti înainte ca interogarea să aibă loc efectiv) un tabel *read-only* cu date adunate din diverse microservicii.

Pentru rapoarte şi interogări complexe, care nu necesită date în timp-real, o abordare destul de comună este exportarea datelor din aşa numitele „baze de date fierbinţi” în baze de date „reci”.

Pentru a atinge o consistenţă între microservicii, o soluţie bună o reprezintă „eventuala-consistenţă”, care se poate realiza printr-o comunicare condusă de evenimente (*event driven communication*) sau un sistem „publicare-abonare” (publish/subscribe).

O mare parte din contextul de abordare bazat pe microservicii este dată, de fapt, de tehnologiile asociate care uşurează enorm procesele de dezvoltare, testare, implementare şi gestionarea codului şi a versiunii.

Microserviciile derivă din SOA (*Service Oriented Architecture*), chiar dacă actualmente sunt două lucruri total diferite. Unii specialişti consideră că microserviciile sunt „SOA implementat corect”.

Tabel de Comparare SOA şi Microservicii

SOA	MICROSERVICII
Arhitectură distribuită, componentizată, reutilizabilă	Arhitectură distribuită, componentizată, reutilizabilă
Servicii Web (SOAP/REST)	Servicii Web (REST)
XML	XML/JSON
Transport prin HTTP(S)	Transport prin HTTP(S)
Contract-decoupling (WSDL) (*)	No contract-decoupling

(*) Prin decuplarea contractului de servicii (*contract-decoupling*), implementarea serviciilor poate fi dezvoltată fără a avea un impact direct asupra consumatorilor de servicii.

Una dintre cele mai importante diferenţe dintre SOA (Service Oriented Architecture) şi Microservicii o reprezintă granularitatea functionalităţilor şi contextul unui anumit serviciu. Dacă o componentă a unui serviciu SOA poate efectua mai multe operatiuni într-un singur pachet, o arhitectură structurată pe microservicii ar descompune respectiva componentă în alte servicii mici şi independente, pornind de la prezumpția „cu cât mai mic, cu atât mai bine”.

O regulă importantă este că fiecare microserviciu ar trebui sa deţină datele şi logica proprie. Acest principiu este similar cu cel din DDD (*Domain Driven Design*), unde fiecare context sau subsistem autonom trebuie să deţină propriile modele de domeniu.

Context Mapping Pattern este un şablon de determinare a limitelor unui context, des utilizat in DDD.

Standarde pentru modelul de date bazat pe microservicii pentru CJ Cluj

- **Pe nivelul tehnologiilor actuale, formatul preferat atât pentru comunicare sincronă (REST) cât şi asincronă (AMQP) este JSON**

JSON (JavaScript Object Notation) este un format pentru interoperabilitatea datelor foarte ușor de înțeles şi utilizat. Este uşor de asimilat/adoptat atât de către specialiştii IT, cât şi la nivelul sistemele de calcul –prezentând uşurință în parsare și generarea task-urilor. Se bazează pe un subset al limbajului de programare JavaScript, ECMA-262 ediția a treia.
JSON este de fapt un format de tip text, care este complet independent de limbaj, dar utilizează convenții cunoscute programatorilor din familia de limbaje C, inclusiv C, C ++, C #, Java, JavaScript, Perl, Python și multe altele.
Aceste proprietăți fac din JSON un limbaj ideal pentru interoperabilitatea datelor.

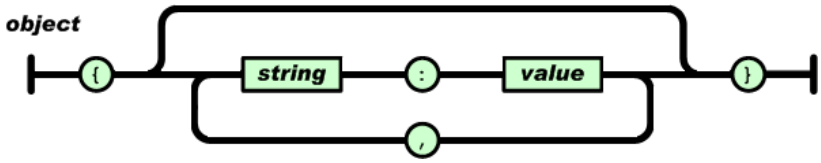
Formatul JSON este construit pe două structuri:

- O colecție de perechi (nume,valoare) - în diferite limbaje, acest lucru este realizat fie ca un obiect, înregistrare, structură, dicționar, listă cu chei, array asociativ sau tabelă hash (structură de date care implementează un tip de date abstract numit matrice asociativă - o structură care poate mapa chei la valori)
- O listă ordonată de valori - în majoritatea limbajelor, acest lucru este realizat ca un array, vector, listă sau secvență.

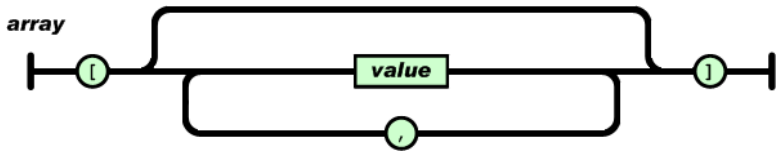
Acestea sunt structuri de date universale. Practic toate limbajele de programare moderne le suportă într-o formă sau alta.

În JSON, aceste structuri se concretizează după cum urmează:

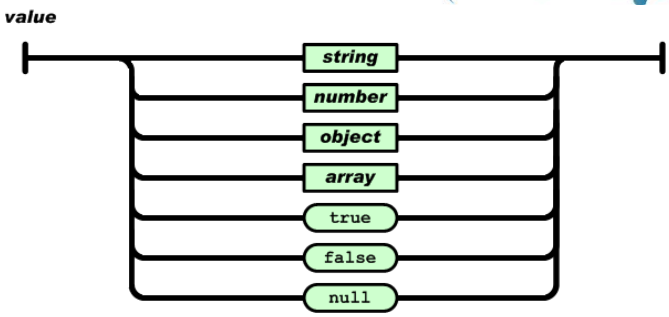
- Un obiect este un set neordonat de perechi (nume, valoare). Un obiect începe cu { (acolada stanga - deschisă) și se termină cu } (acolada inchisă). Fiecare nume este urmat de „:” (două puncte) și perechile (nume, valoare) sunt separate prin virgulă.



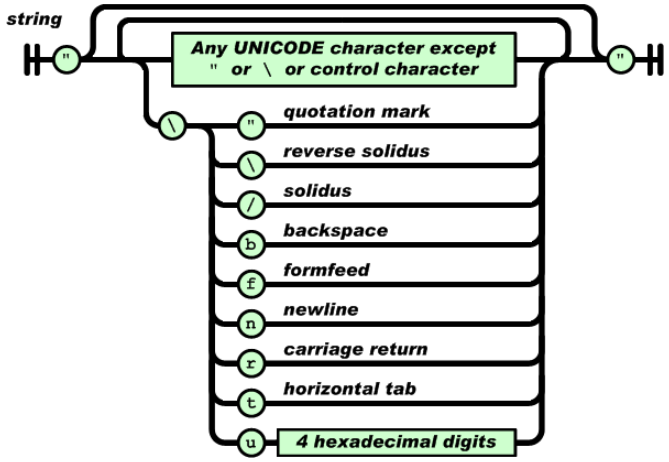
- Un array (tablou uni sau multi-dimensional) este o colecție ordonată de valori, care începe cu „[” (paranteza dreaptă deschisă) și se termină cu „]” (paranteza dreaptă inchisă), iar valorile sunt separate prin virgulă.



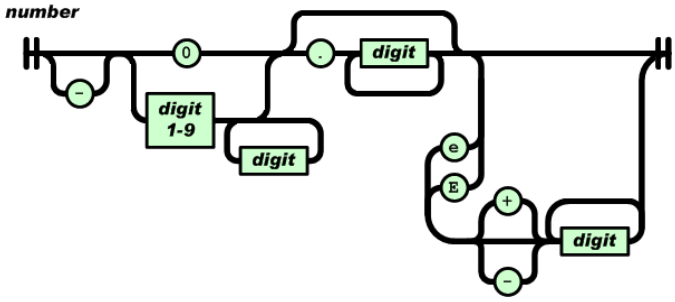
- O valoare poate fi un șir între ghilimele duble, sau un număr, sau TRUE, sau FALSE, sau NULL, sau un obiect sau un array. Aceste structuri pot fi imbricate.



- Un șir este o secvență de caractere de zero sau mai multe caractere Unicode, închise între ghilimele duble, folosind backslash ca şi caracter de evitare. Un caracter este reprezentat ca un șir de caractere unic. Un șir este foarte asemănător unui șir C sau Java.



- Un număr este foarte asemănător unui număr C sau Java, cu excepția faptului că nu sunt utilizate formatele octale și hexazecimale.



- Spațiul poate fi inserat între orice pereche de „tokeni” (simboluri).

Cu excepția câtorva detalii de codificare, aceste diagrame descriu complet formatul JSON.

O manieră mai direct exprimată în termeni de cod sursă ar fi următoarea:

```
json
    element

value
    object
    array
    string
    number
    "true"
    "false"
    "null"

object
    '{' ws '}'
    '{' members '}'

members
    member
    member ',' members

member
    ws string ws ':' element

array
    '[' ws ']'
    '[' elements ']'

elements
    element
    element ',' elements
element
    ws value ws
string
    '"' characters '"'
characters
    ""
    character characters
character
    '0020' . '10ffff' - '"' - '\'
    '\' escape

escape
    '"'
    '\'
    '/'
    'b'
    'n'
    'r'
    't'
    'u' hex hex hex hex
```

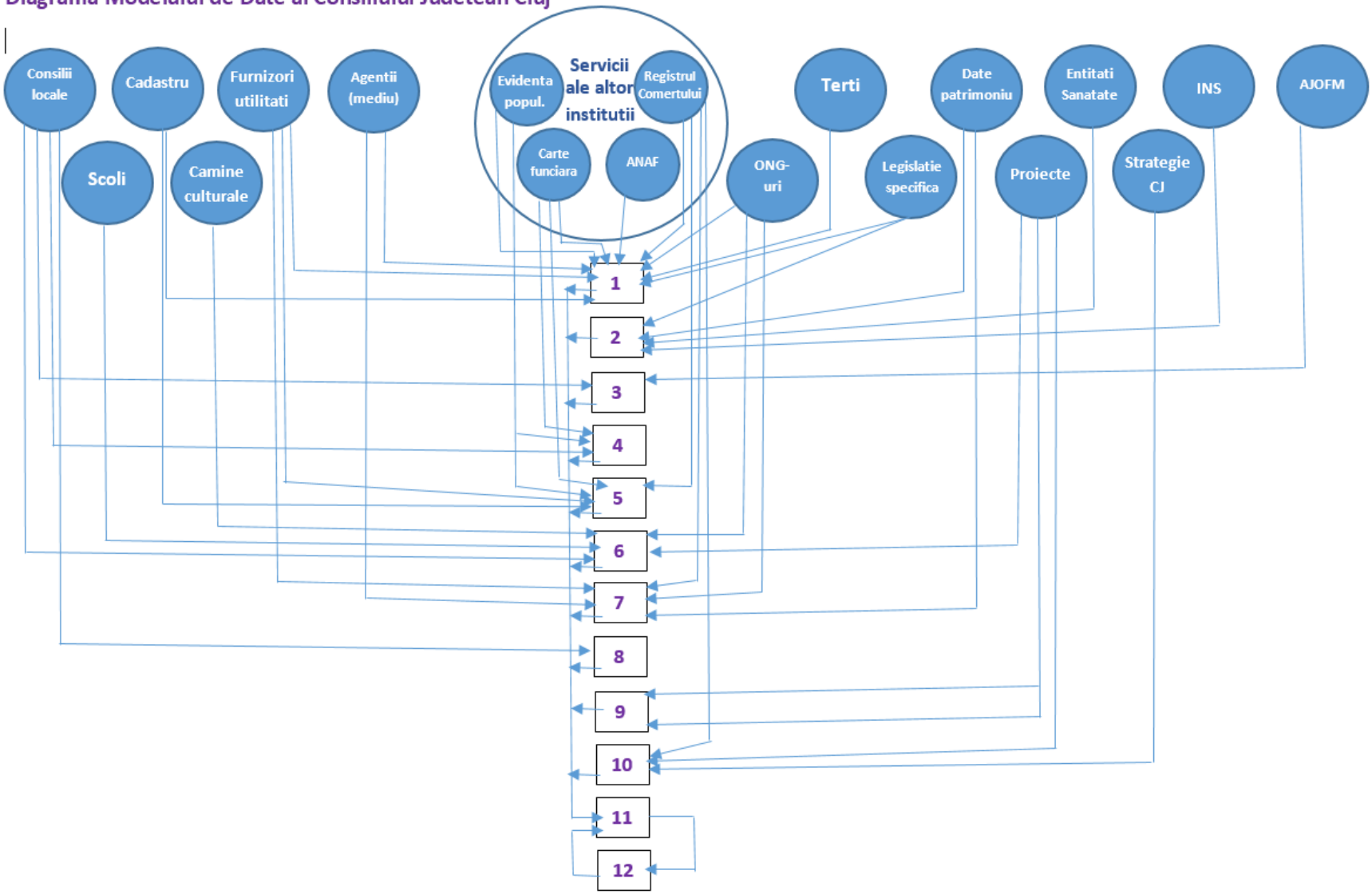
```
hex
    digit
    'A' . 'F'
    'a' . 'f'

number
    int frac exp

int
    digit
    onenine digits
    '-' digit
    '-' onenine digits

digits
    digit
    digit digits

digit
    '0'
    Onenine

Onenine
'1' . '9'

frac
    ""
    '.' digits

exp
    ""
    'E' sign digits
    'e' sign digits

sign
    ""
    '+'
    '-'

ws
    ""
    '0009' ws
    '000a' ws
    '000d' ws
    '0020' ws
```

Alte convenții utilizate în ceea ce priveşte datele:

- Numele propritatilor trebuie sa fie **snake_case** / convenție de scriere a cuvintelor compuse astfel încât cuvintele să fie separate cu un simbol de subliniere (_) în loc de un spațiu
- Propritățile variabile, care ar putea strica compatibilitatea backwards (***backwards compatibility***) la un moment dat, ar trebui stocate în **metadata** sau **additional_properties**

- Nicio valoare de tip ***null*** nu va fi transmisă sau returnată (lipsa unei valori implică ***null***)
- Niciun ***array*** gol nu trebuie sa fie ***null***
- ***Timestamp***-urile si câmpurile de tip ***date/time*** trebuie sa fie transmise in format UTC (***Coordinated Universal Time***)
- Valorile de tip Boolean nu pot fi ***null***
- Proprietățile de tip ***array*** trebuie sa fie la plural
- In cazul câmpurilor unde există anumite standarde recunoscute - ISO (ex. Country Code, Language Code) se va utiliza standardul respectiv.

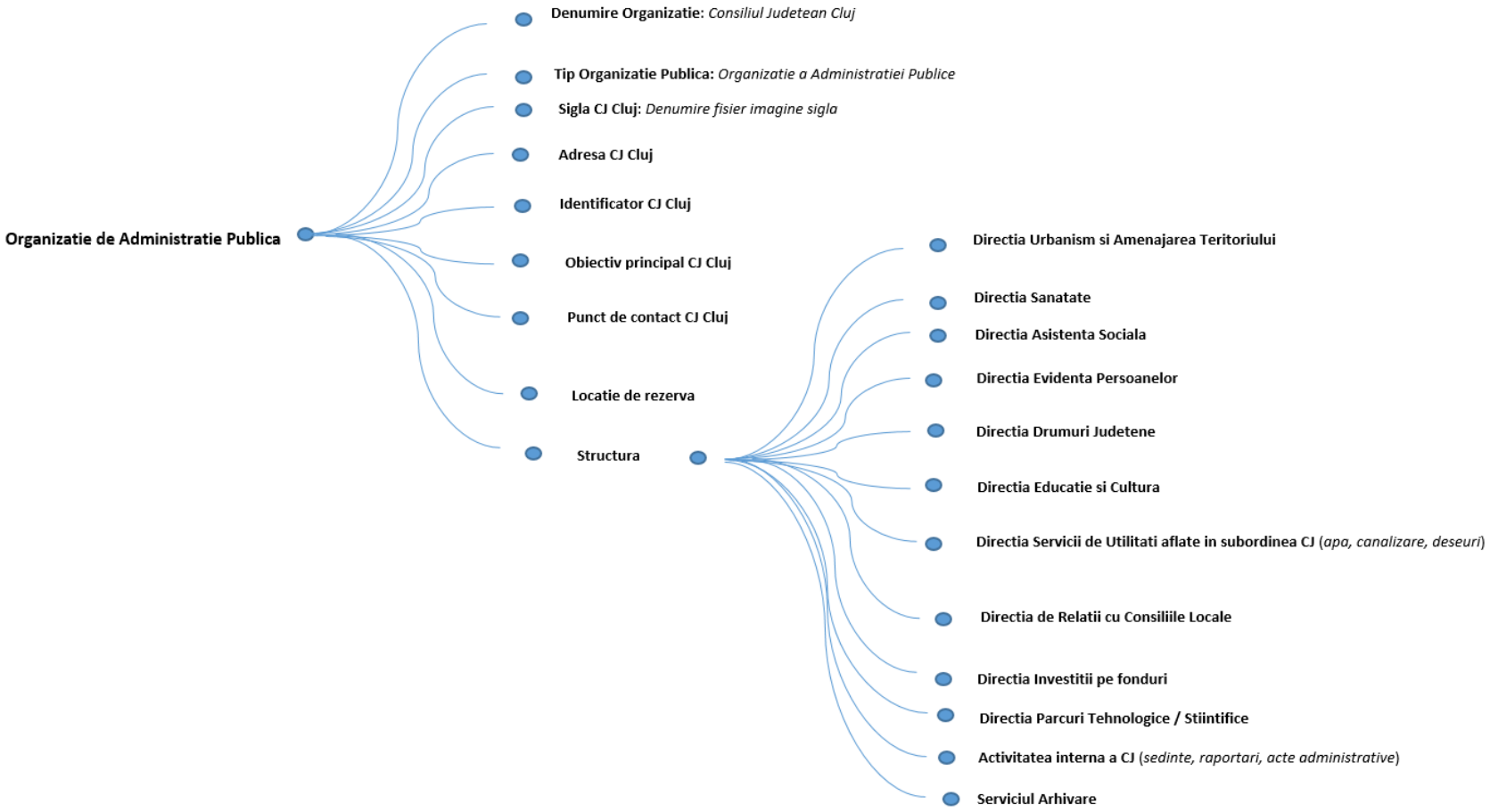
Diagrama Modelului de Date al Consiliului Judetean Cluj



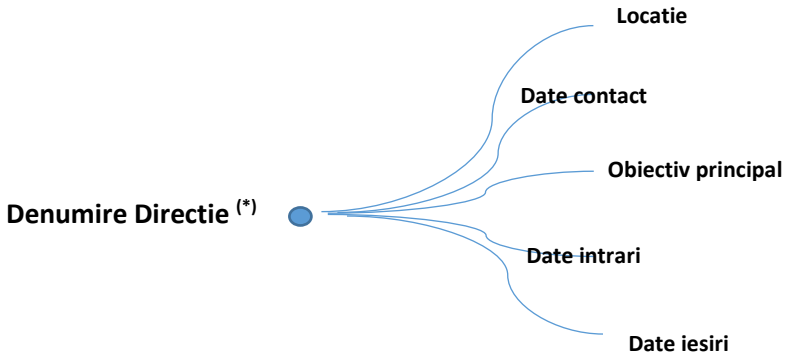
Legenda:

1. Directia Urbanism si Amenajarea Teritoriului
2. Directia Sanatate
3. Directia Asistenta Sociala
4. Directia Evidenta Persoanelor
5. Directia Drumuri Judetene
6. Directia Educatie si Cultura
7. Directia Servicii de Utilitati aflate in subordinea CJ (apa, canalizare, deseuri)
8. Directia de Relatii cu Consiliile Locale
9. Directia Investitii pe fonduri
10. Directia Parcuri Tehnologice / Stiintifice
11. Activitatea interna a CJ (sedinte, raportari, acte administrative)
12. Serviciul Arhivare

Detalierea Modelului de Date al CJ Cluj conform standardului SEMIC (*Semantic Interoperability Community*)



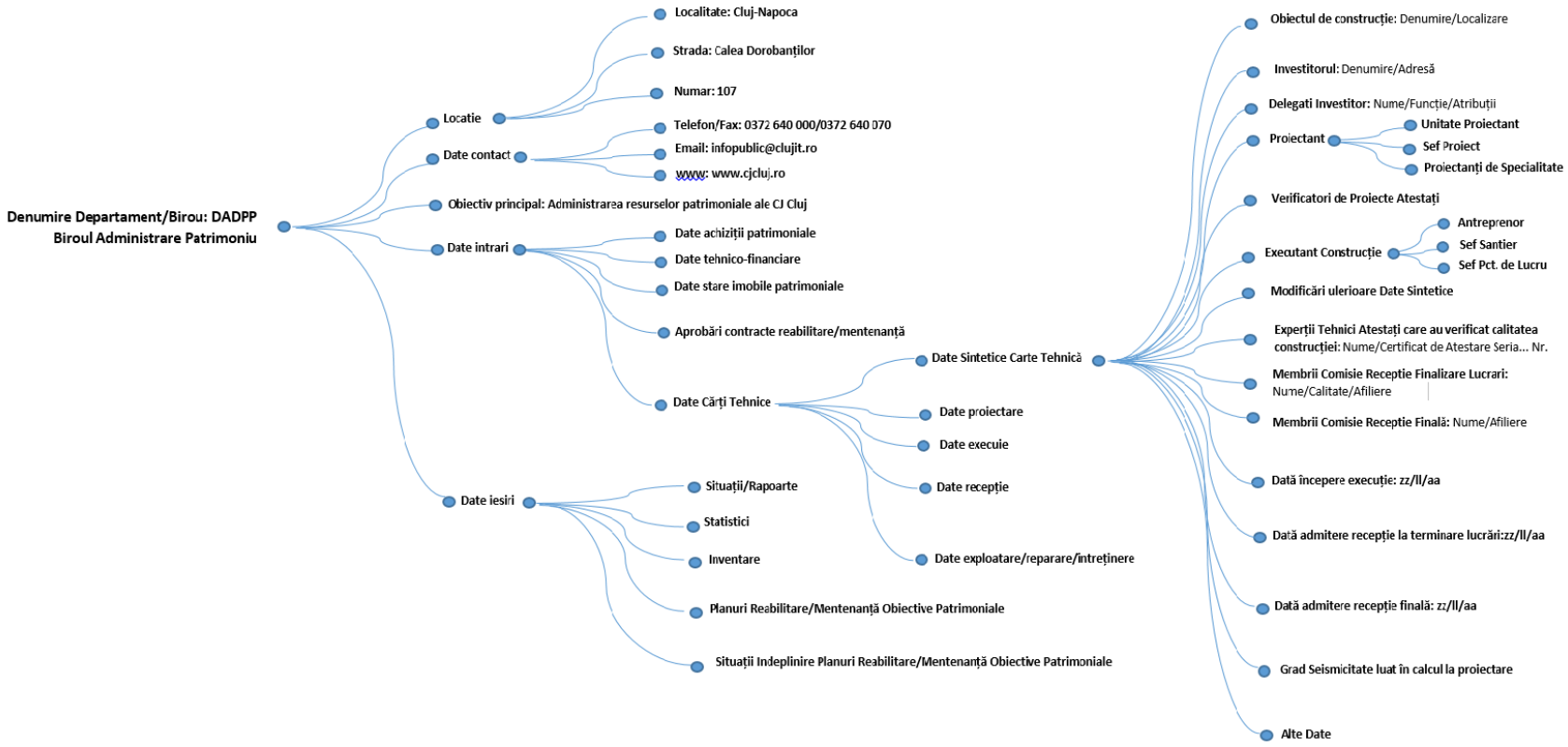
Sablonul SEMIC pentru detaliere date pe departamentele CJ Cluj



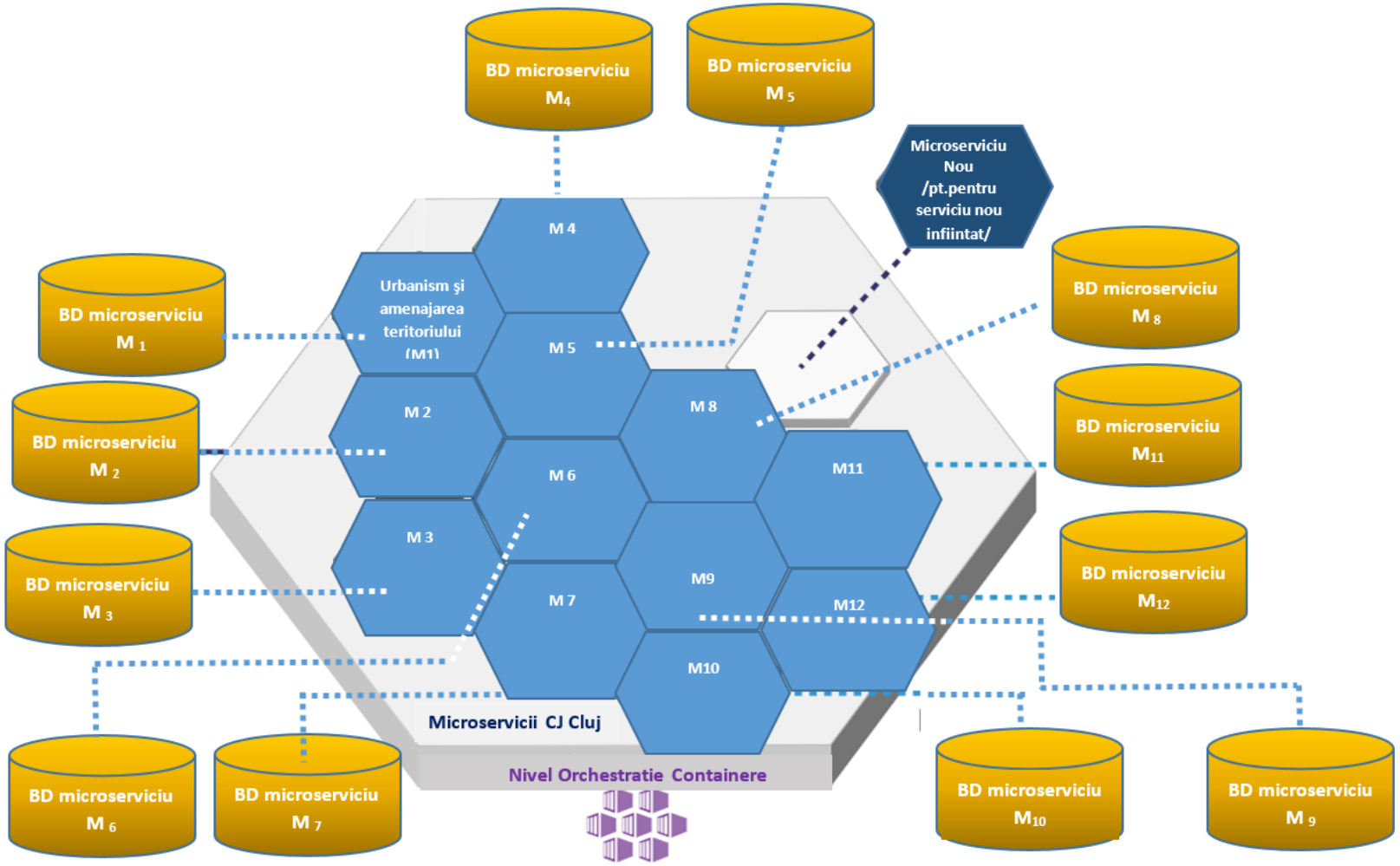
(*) Se intocmeste detaliat pe toate cele 12 directii ale CJ Cluj

Model SEMIC detaliere date pentru Departamentul/Direcția de Administrare a Domeniului Public şi Privat/Biroul Administrare Patrimoniu al CJ Cluj

`



Modelul de date pentru CJ Cluj orientat pe microservicii: Arhitectura de Baze de Date Hibride bazata pe conceptul de Persistența Poliglotă



Legenda:

M1. Componenta Microserviciu Software pentru Directia Urbanism si Amenajarea Teritoriului

M2. Componenta Microserviciu Software pentru Directia Sanatate

M3. Componenta Microserviciu Software pentru Directia Asistenta Sociala

M4. Componenta Microserviciu Software pentru Directia Evidenta Persoanelor

M5. Componenta Microserviciu Software pentru Directia Drumuri Judetene

M6. Componenta Microserviciu Software pentru Directia Educatie si Cultura

M7. Componenta Microserviciu Software pentru Directia Servicii de Utilitati aflate in subordinea CJ (apa, canalizare, deseuri)

M8. Componenta Microserviciu Software pentru Directia de Relatii cu Consiliile Locale

M9. Componenta Microserviciu Software pentru Directia Investitii pe fonduri

M10. Componenta Microserviciu Software pentru Directia Parcuri Tehnologice / Stiintifice

M11. Componenta Microserviciu Software pentru Activitatea interna a CJ (sedinte, raportari, acte administrative)

M12. Componenta Microserviciu Software pentru Serviciul Arhivare