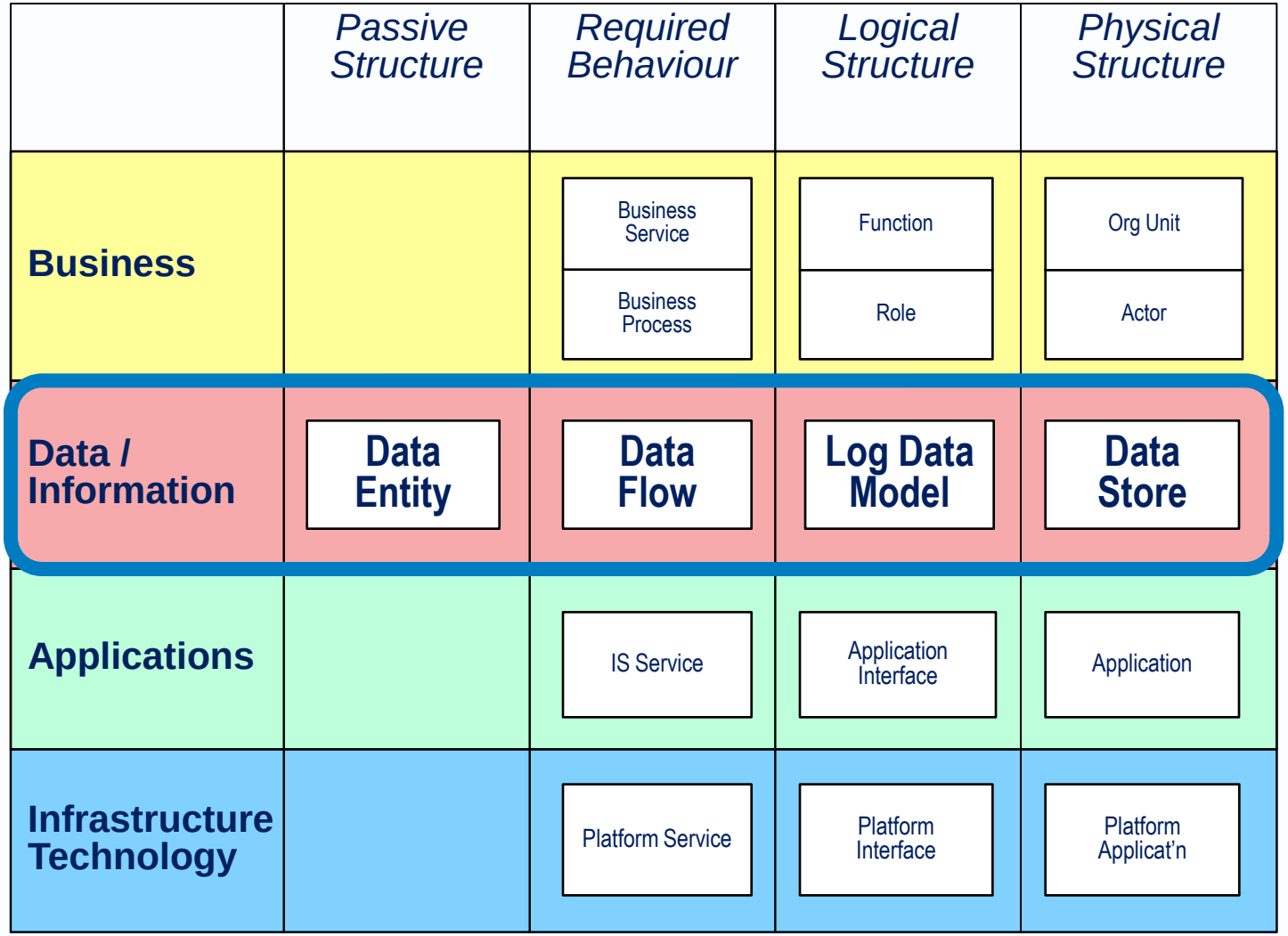
Avancier Methods (AM)

Data Architecture

Design data stores: document stores

The data/information architecture domain/layer



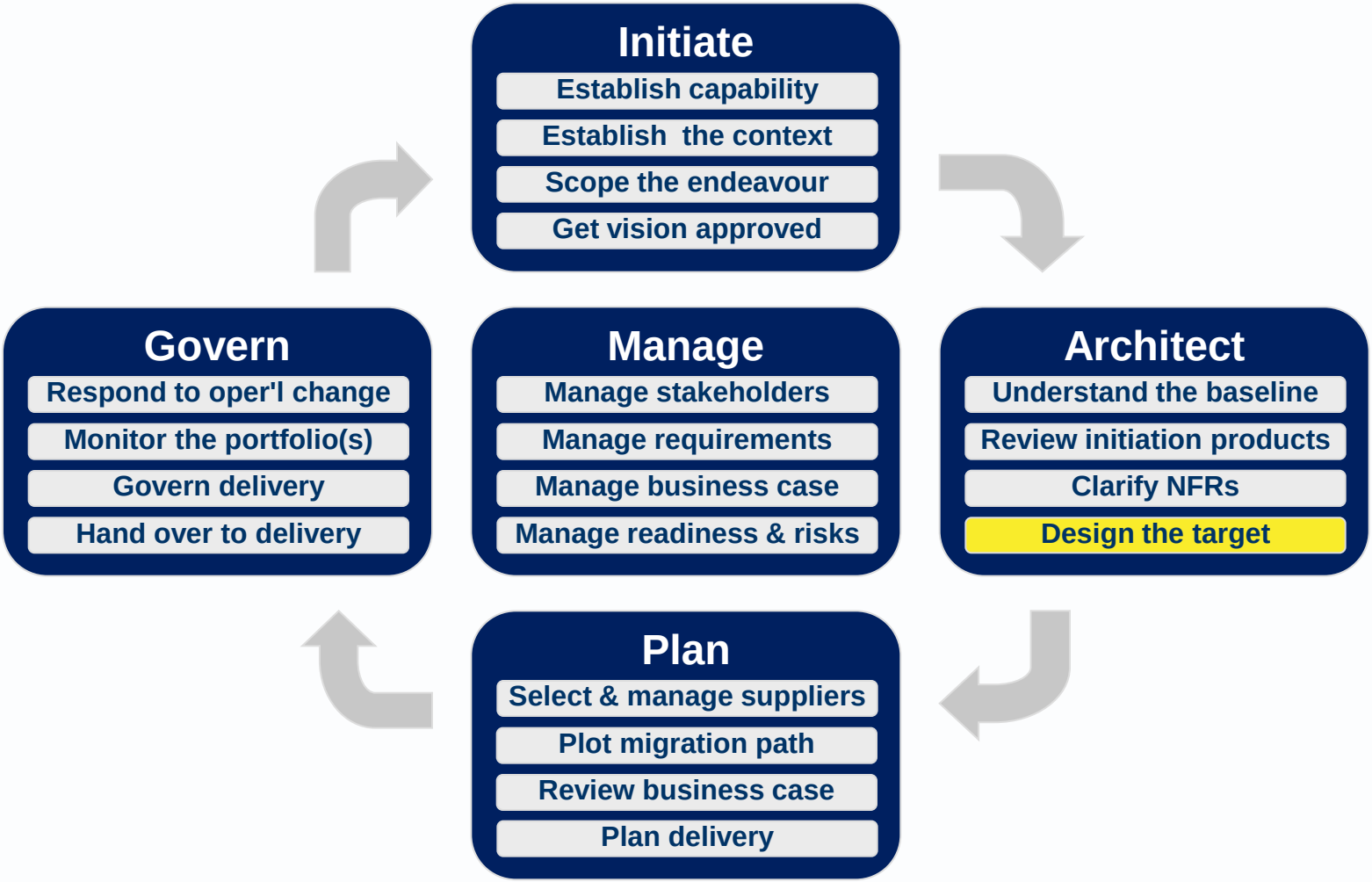
Digitisation of document creation and use

- In theory, data architects can design non-digital data flows.
- In practice, data architects mostly focus on business systems in which business information is to be digitised.

- Business facts are created and used during business operations
  - e.g. customer address, order id, product id, product amount, order receipt, order rejection
- These logical facts must be moved in flows from senders to receivers

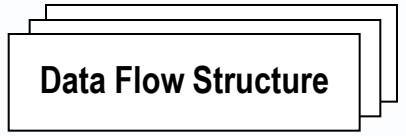
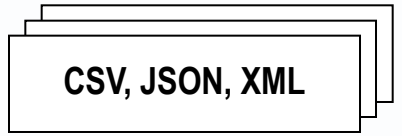
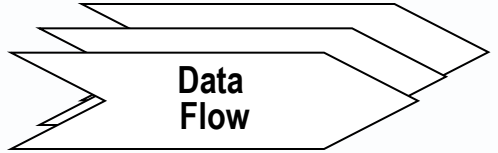
- Transport mechanisms include
  - Keyboard data entry
  - User interface display
  - Transaction message
  - Message in queue
  - Serial file

Design the target (AM level 2)



Canonical data model

Avancier

Conceptual	Defines the ideal or common types for data items in data flows 
Logical	
Physical	
Real	

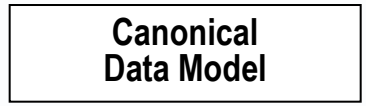
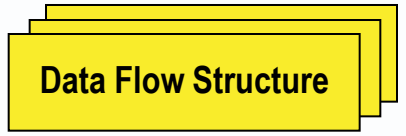
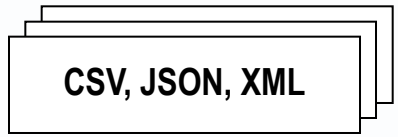
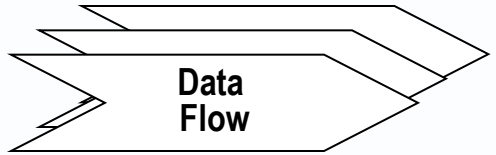
Design target data architecture (AM level 3 and 4)

Avancier

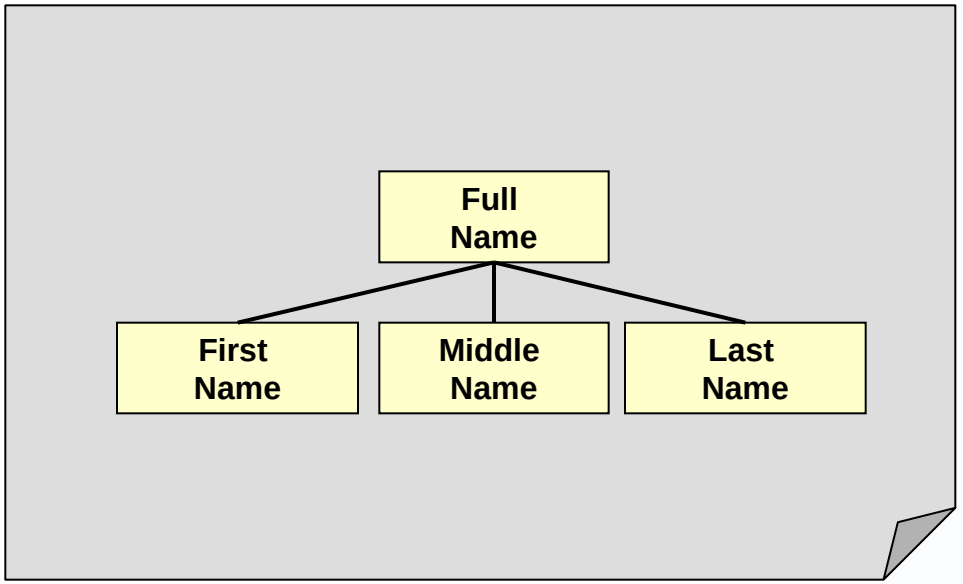
1. Define the business context for data creation and use
2. Define data flows (I/O messages, displays, forms and reports)
3. Define data dictionary or canonical data model
4. **Define data store(s): document stores**
    1. **Define each input document as a logical structure (a regular expression)**
    2. **Define additional context data to be stored (provenance information etc.)**
    3. **Code input data structures using chosen data format standard (XML, JSON…)**
    4. **Refine design to meet output requirements, keeping input documents intact.**
    5. **Design to ensure the database satisfies CIA and scalability requirements.**
5. Address data quality issues

Canonical data model

Avancier

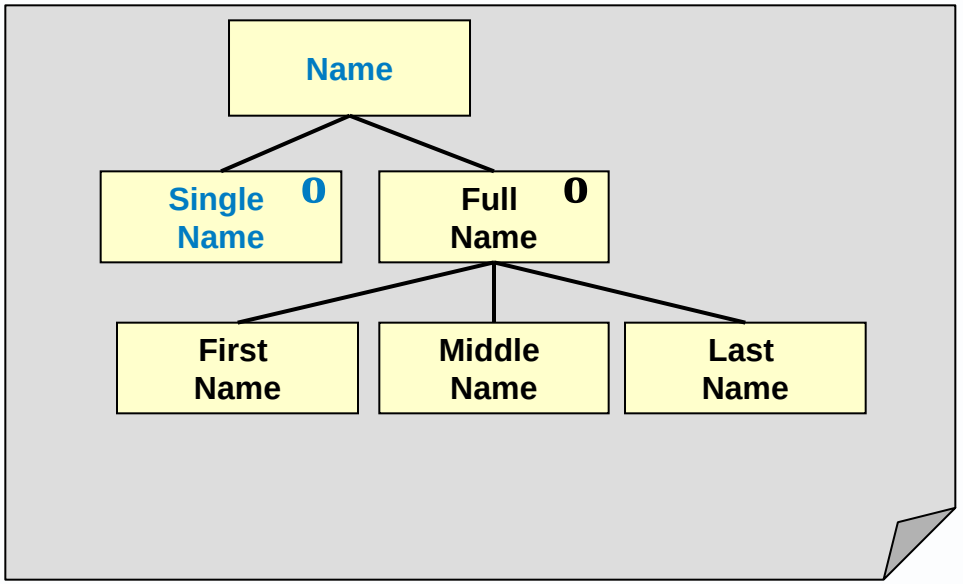
Conceptual	Defines the ideal or common types for data items in data flows 
Logical	The proper form to define the logical structure of a data flow is a regular expression 
Physical	
Real	

Document as a simple sequence of fields



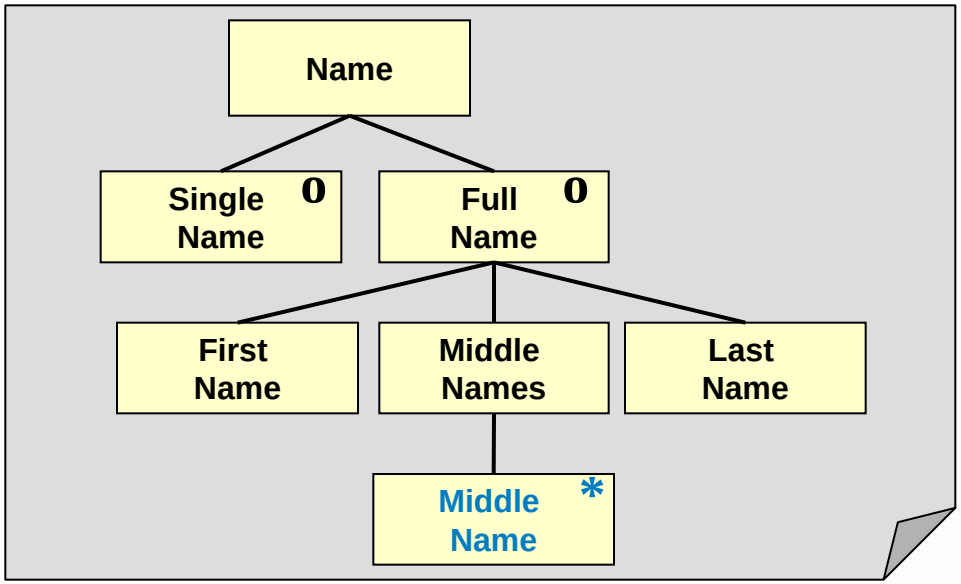
Full name SEQUENCE
First Name
Middle Name
Last Name
Full name END

Document with optional elements



Name SELECT
    Single Name
Name OR
    Full name SEQUENCE
        First Name
        Middle Name
        Last Name
    Full name END
Name END

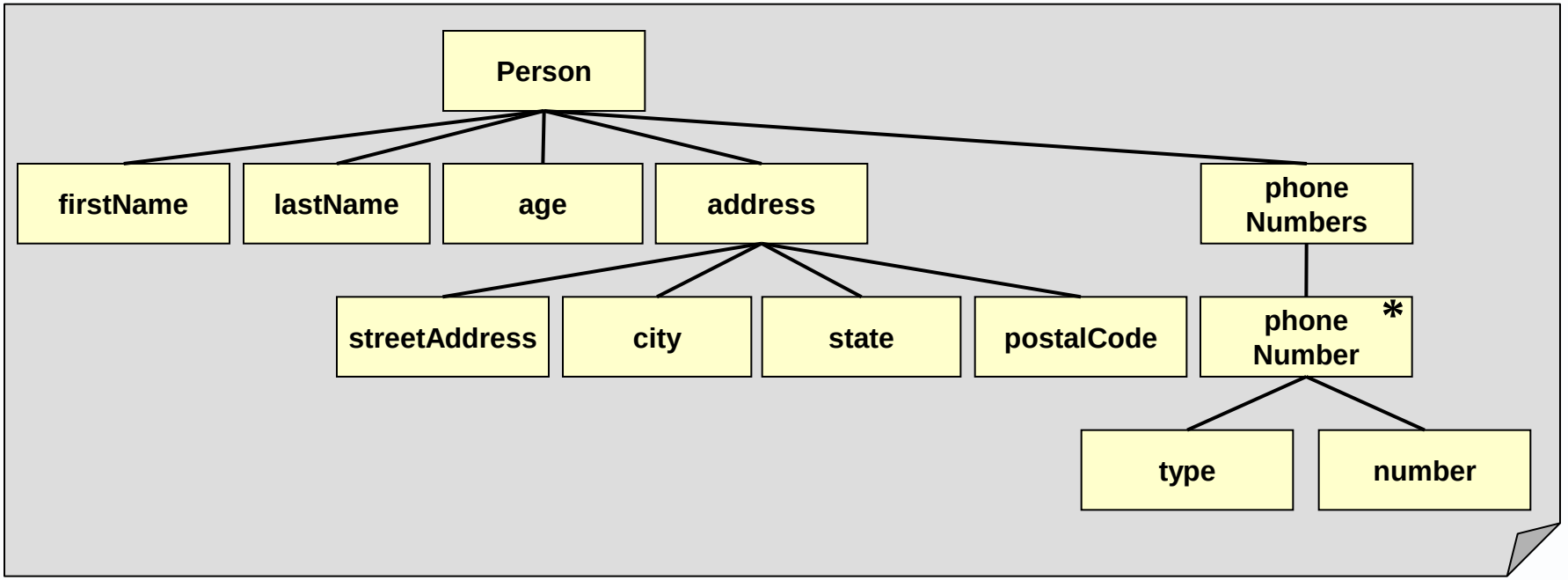
Document with iterated element



Name SELECT
    Single Name
Name OR
    Full name SEQUENCE
        First Name
        Middle Names ITERATION
            Middle Name
        Middle Names END
        Last Name
    Full name END
Name END

Data flow structure diagram (regular expression)

- A document in a data flow structure can be defined as
  - A hierarchical structure – a regular expression
  - A “Jackson structure” like this, or in the form of an XML schema



Code the input data structures using the data format standard

Avancier

Conceptual	Defines the ideal or common types for data items in data flows	**Canonical Data Model**
Logical	The proper form to define the logical structure of a data flow is a regular expression	**Data Flow Structure**
Physical	A data flow's content can be defined in various forms	**CSV, JSON, XML**
Real		**Data Flow**

There are many standard data flow formats,

Avancier

Digital image data
- TIFF version 6 uncompressed (.tif)
- JPEG (.jpeg, .jpg)
- PDF (.pdf)

Digital video data:
- MPEG-4 High Profile (.mp4)
- JPEG 2000 (.mj2)

Digital audio data
- Free Lossless Audio Codec (FLAC) (.flac)
- Waveform Audio Format (WAV) (.wav)
- MPEG-1 Audio Layer 3 (.mp3)

Documentation and scripts
- Open Document Text (.odt)
- Rich Text Format (.rtf)
- HTML (.htm, .html)
- plain text (.txt)
- widely-used proprietary formats
  - e.g. MS Word (.doc/.docx) or MS Excel (.xls/ .xlsx)
- XML marked-up text (.xml) to a DTD or schema, e.g. XHMTL 1.0
- PDF (.pdf)

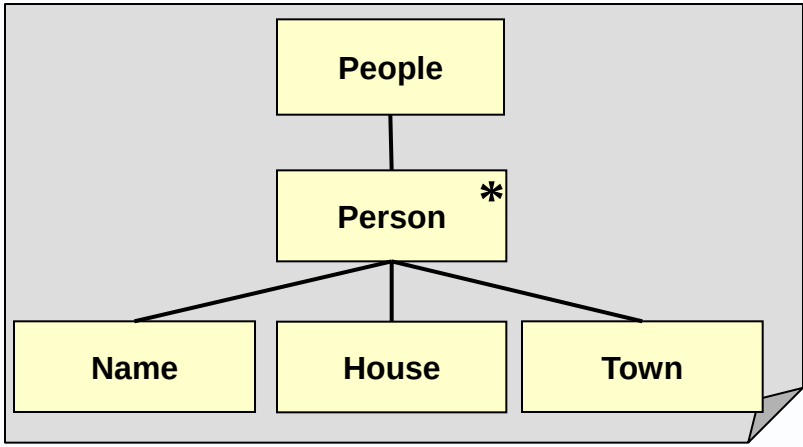
Geospatial data; vector and raster data
- ESRI Shapefile (essential -- .shp,.shx, .dbf;
- optional -- .prj, .sbx, .sbn)
- geo-referenced TIFF (.tif, .tfw)
- CAD data (.dwg)
- tabular GIS attribute data

Qualitative data, textual
- **eXtensible Mark-up Language (XML) text according to a Document Type Definition (DTD) or schema (.xml)**
- Rich Text Format (.rtf)
- plain text data, ASCII (.txt)
- Hypertext Mark-up Language (HTML) (.html)
- widely-used proprietary formats, e.g. MS Word (.doc/.docx)

Quantitative tabular data with extensive metadata
- SPSS portable format (.por)
- delimited text and command ('setup') file (SPSS, Stata, SAS, etc.) containing metadata information
- structured text or mark-up file containing metadata information, e.g. DDI XML file
- MS Access (.mdb/.accdb)

Quantitative tabular data with minimal metadata:
- **comma-separated values (CSV) file (.csv)**
- tab-delimited file (.tab) including delimited text of given character set with SQL data definition statements where appropriate
- delimited text of given character set -- only characters not present in the data should be used as delimiters (.txt)
- widely-used formats, e.g. MS Excel (.xls/.xlsx), MS Access (.mdb/.accdb), dBase (.dbf) and OpenDocument Spreadsheet (.ods)

Simple data flow format – CSV



- Comma-separated values
- For simple data flow structures
  - John, 3 South Street, Big Town
  - Mary, 44 North Street, Small Town
- Spreadsheet import/export
- Meaning is not usually contained in the data flow
- The raw data can appear meaningless
  - A, 0001, 23
  - B, 9888, 10
- So, sender and receiver must know the meaning of each data item

Self-describing data flow formats – JSON

- A JSON data flow contains not only
  - Data (data values) but also
  - Data descriptors (data types)
    - Number
    - String
    - Boolean
    - Array

- So the data flow can be sent to recipients who do not already know the data types, but can find them in the flow itself.

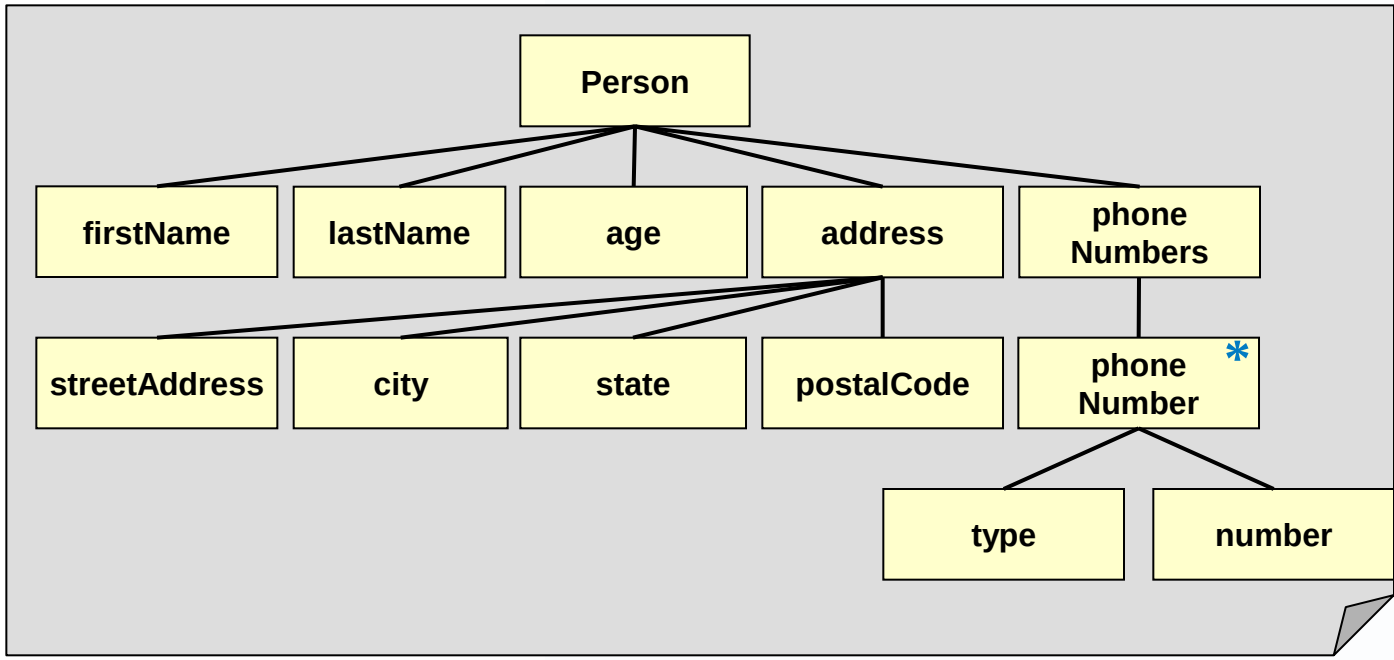
Self-describing data flow formats – JSON

- **Number** (double precision floating-point format in JavaScript, generally depends on implementation)
- **String** (double-quoted Unicode, with backslash escaping)
- **Boolean** (true or false)
- **Array** (an ordered sequence of values, comma-separated and enclosed in square brackets; the values do not need to be of the same type)
- **Object** (an unordered collection of key:value pairs with the ':' character separating the key and the value, comma-separated and enclosed in curly braces; the keys must be strings and should be distinct from each other)
- **null** (empty)

- Non-significant white space may be added freely around the "structural characters" (i.e. brackets "{ } [ ]", colons ":" and commas ",").

JSON data structure


```
{
    "firstName":
    "lastName":
    "age":
    "address":
        "streetAddress":
        "city":
        "state":
        "postalCode":
     "phoneNumbers": [
        {
            "type":
            "number":
        },
        {
            "type":
            "number":
        }
    ]
}
```



http://en.wikipedia.org/wiki/JSON

JSON representation of an object that describes a person.


```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021 },
     "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

The object has

string fields for first and last name,

a number field for age,

an object representing the person's address

an array of phone number objects.

Self-describing data flow format – XML

- XML (eXtensible Mark up Language)

- Like JSON, an XML data flow contains
  - not only the data (values) but also
  - descriptors of the data (types)

- How does XML differ from JSON?
  - Con: Clunkier
  - Pro: Can be supported by a separate XML Schema Definition (XSD)

XSD: XML Schema Definition language

- used to
  - define the data structure of an XML document
  - enables verification of a document's data integrity
  - loosens coupling between sender and receiver

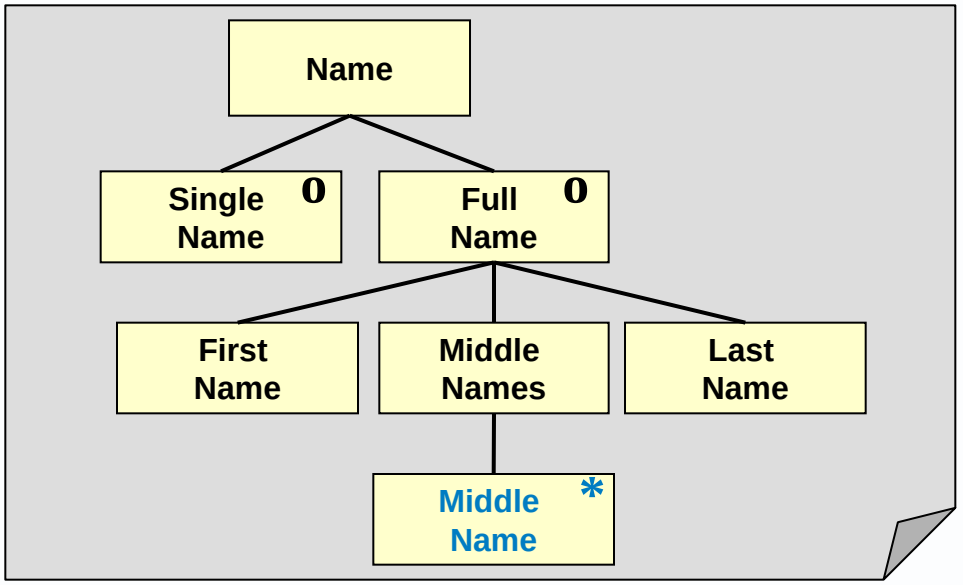
- E.g. a simple type - a subtype of string – with a range of values

```
?xml version="1.0"?
xsd:schema xmlns:xsd=http://www.w3.org/1999/XMLSchema

    xsd:simpleType name="PersonTitle" base="xsd:string"
        xsd:enumeration value="Mr." /
        xsd:enumeration value="Ms." /
        xsd:enumeration value="Dr." /
        xsd:enumeration value="Rev." /
    /xsd:simpleType

    xsd:complexType name="Text" content="textOnly" base="xsd:string" derivedBy="restriction" /
```

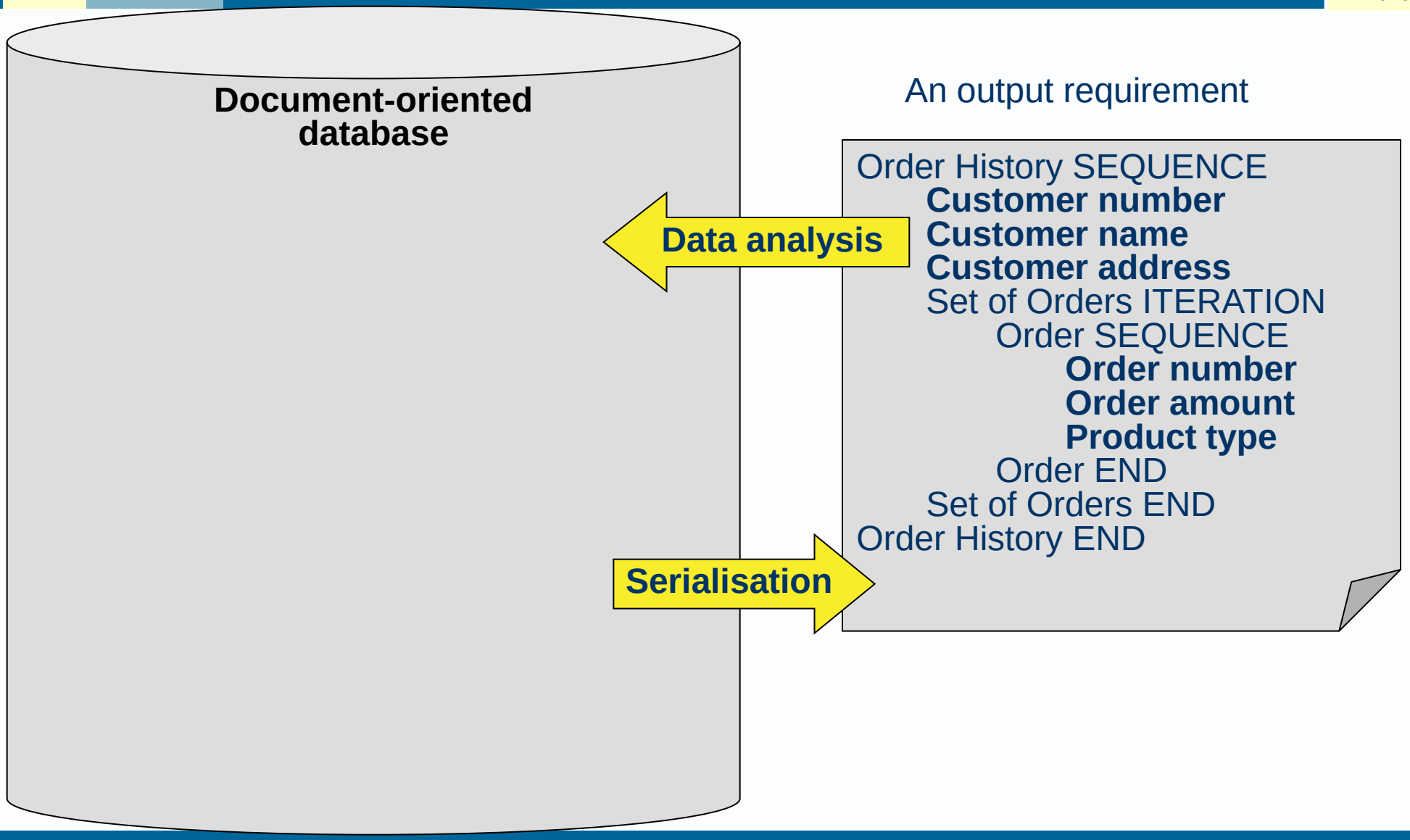
Document as XML schema



Name SELECT
    Single Name
Name OR
    Full name SEQUENCE
        First Name
        Middle Names ITERATION
            Middle Name
        Middle Names END
        Last Name
    Full name END
Name END

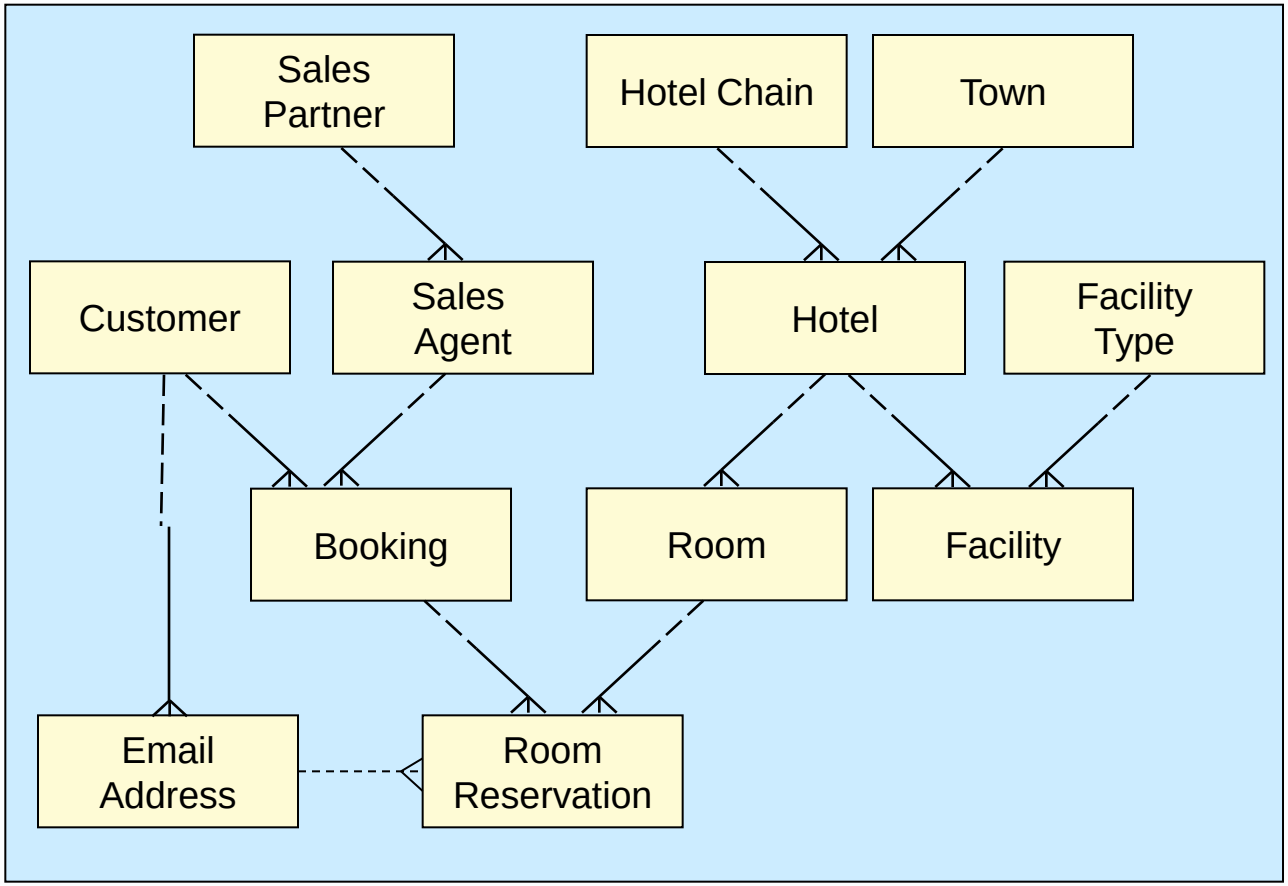
```
xsd:choice
        xsd:element name="SingleName" type="Text" minOccurs="1" maxOccurs="1" /
        xsd:sequence
          xsd:element name="FirstName" type="Text" minOccurs="1" maxOccurs="1" /
          xsd:element name="MiddleName" type="Text" minOccurs="0" maxOccurs="unbounded"/
          xsd:element name="LastName" type="Text" minOccurs="1" maxOccurs="1" /
        /xsd:sequence
/xsd:choice
```

Refine the database design to fulfil output requirements



Partitioning a network – perhaps for scalability

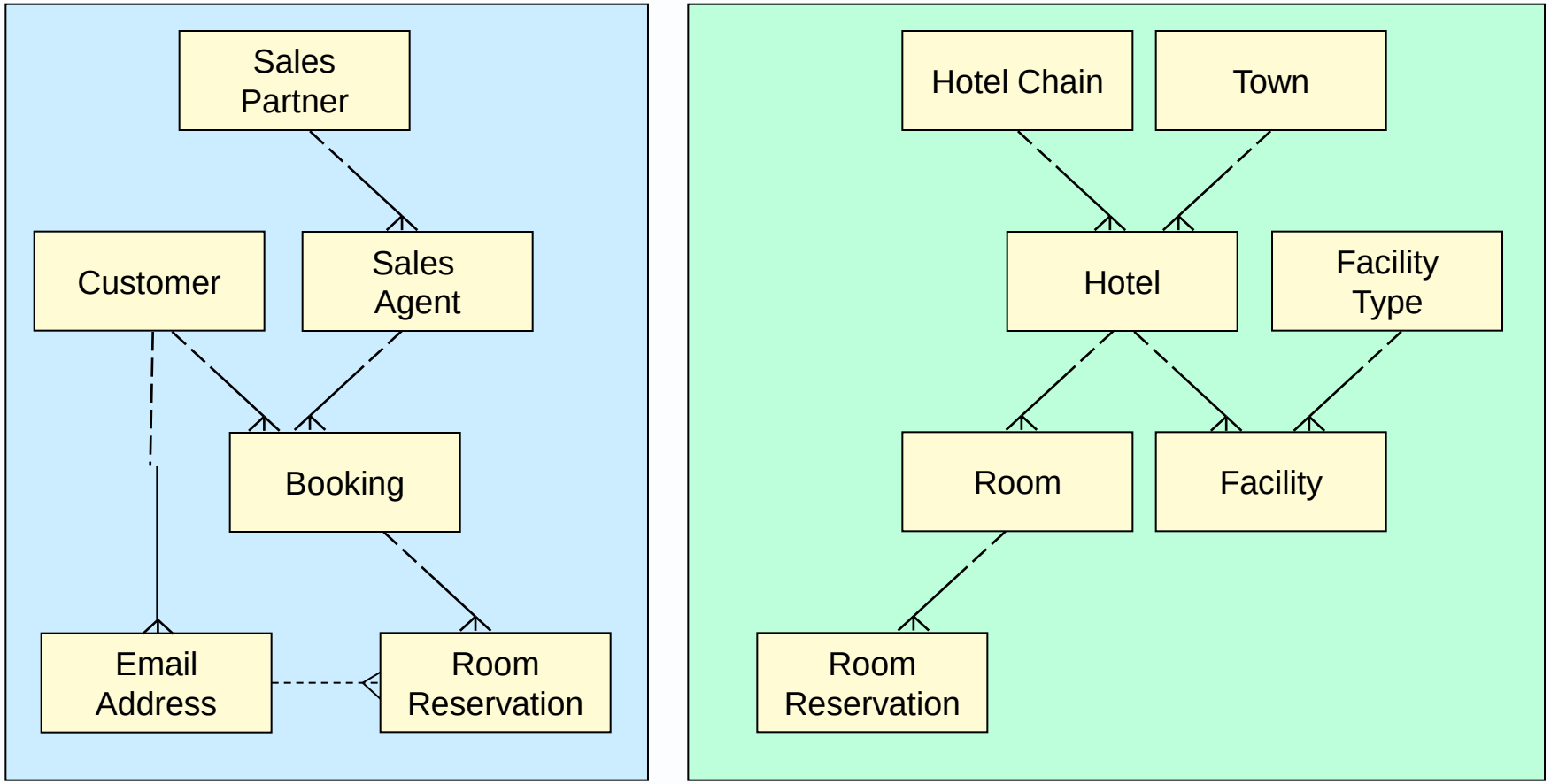
- How would you partition this data model – functionally?



Partitioning a network for scalability

Avancier

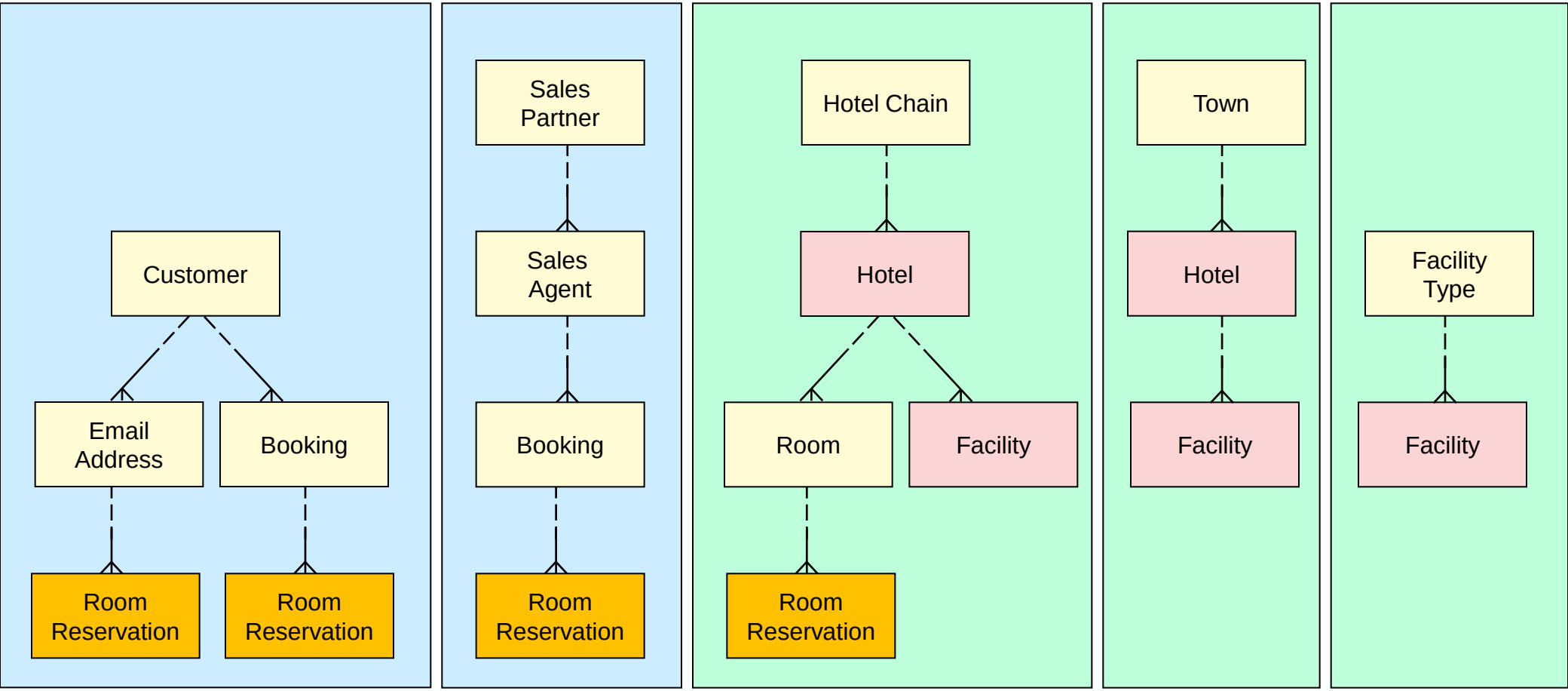
- A natural division



Horizontal partitioning of network into hierarchical structures

Avancier

- Each hierarchical data structure is describable in an XML schema



Design target data architecture (AM level 3 and 4)

Avancier

1. Define the business context for data creation and use
2. Define data flows (I/O messages, displays, forms and reports)
3. Define data dictionary or canonical data model
4. **Define data store(s): document stores**
    1. **Define each input document as a logical structure (a regular expression)**
    2. **Define additional context data to be stored (provenance information etc.)**
    3. **Code input data structures using chosen data format standard (XML, JSON…)**
    4. **Refine design to meet output requirements, keeping input documents intact.**
    5. **Design to ensure the database satisfies CIA and scalability requirements.**
5. Address data quality issues

The physical reality

Avancier

Conceptual	Defines the ideal or common types for data items in data flows	**Canonical Data Model**
Logical	The proper form to define the logical structure of a data flow is a regular expression	**Data Flow Structure**
Physical	A data flow's content can be defined in various forms	**CSV, JSON, XML**
Real	At the bottom-level of a communication stack is the physical medium wires, Microwaves, Sound waves	**Data Flow**