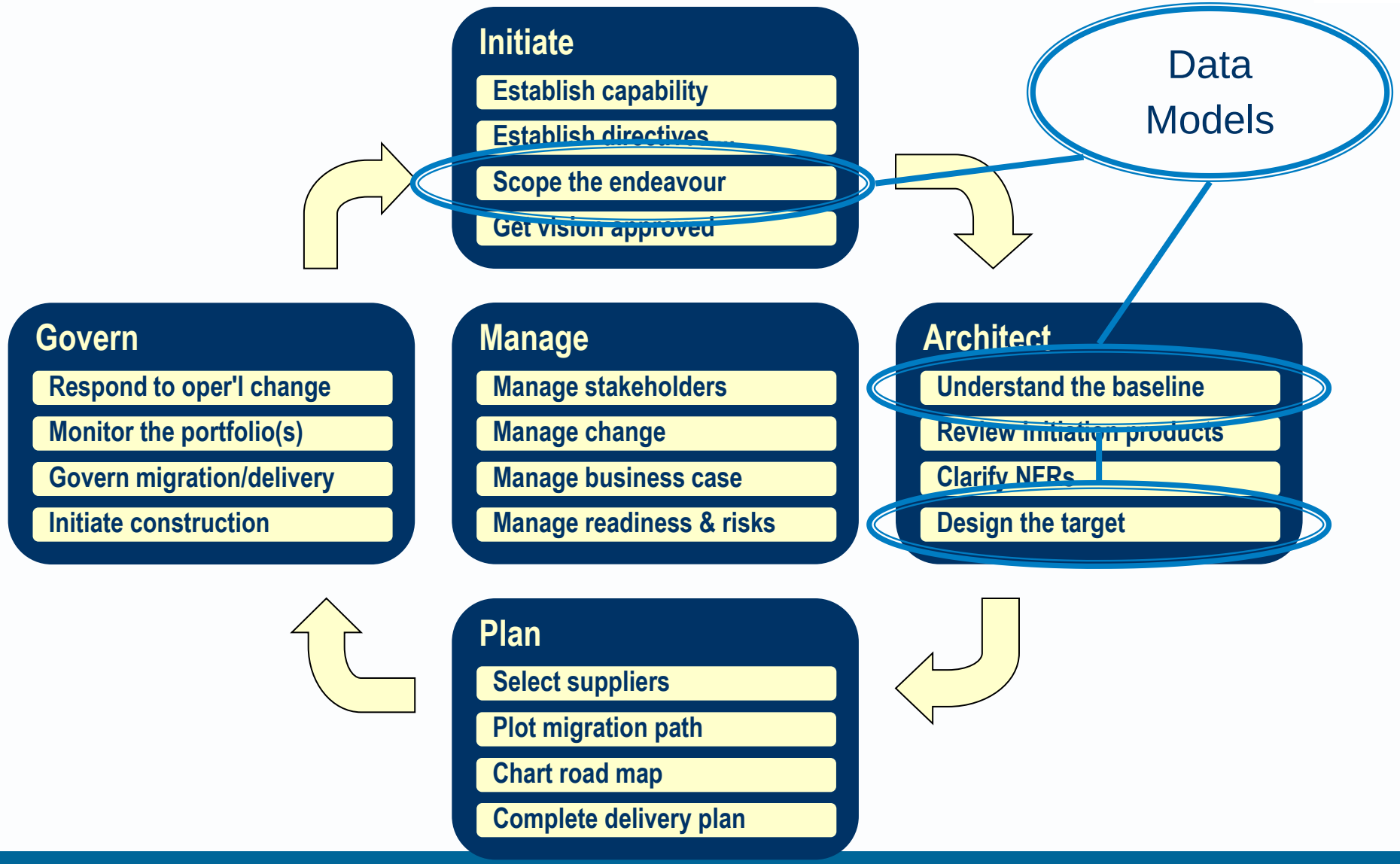
Avancier Methods (AM)

TECHNIQUES

Data models

Where in the AM solution architecture process?

Avancier



First, data architects should understand data structures

- An enterprise or solution architect should understand structural models of data processing systems
- The data architect is concerned with
  - data at rest - data stores
  - data in motion - data flows or messages
- The logical data structures of those data stores and data flows
- Hereafter, some notes on data modelling notations and issues

Entities have identities

- The human parties to any conversation about an entity
  - (say you are discussing a mutual friend)
- must be able to distinguish that entity from other instances of the same an entity type
  - (say, other mutual friends)
- We use identifiers to do this.

“The” as an informal identifier

- Where the number of stakeholders and entity instances is small we usually get away with using “the”.

- When my wife and I discuss “the” house plan – we assume we both refer to
  - one plan (one description entity) of
  - one house (one thing entity).

- Our conversations break down when it turns out we do not share an understanding of which instance “the” refers to.

Names and numbers as more formal identifiers

- The more stakeholders and more entity instances are involved – we more we humans invent more formal identifiers.

- At first names
- Then numbers

- *This has nothing at all to do with computers*

Identifiers reveal entity types

- Every time you come across an identifier range (be it names or numbers) used by humans, you have found an entity type that is important to the humans who invented and use those identifiers.

- *This has nothing at all to do with computers*

- An attribute in one system model can be an entity in another
  - "Size" may be just a property of a thing.
  - "Size" becomes a thing in its own right where humans
    - give each size each a unique identity (Small, Medium, Large)
    - use those identities to locate things of that size
    - record attributes of the size (Maximum Diameter, Minimum Diameter)

Why is this helpful to know?

- Because recognising the need for unique identifiers gives us a way to distinguish an entity type from an attribute type.

- If we don't draw this distinction in a consistent way, then
  - The decision as to what is an entity, attribute or relationship becomes a matter of taste.
  - And we have no consistent way to draw or read those box-line diagrams commonly known as
    - conceptual models
    - domain models
    - data models

30 top tips – or Universal Modelling Principles

- Model with a purpose
- Analyse the data in required I/O data flows
- Analyse the data to “1st normal form”
- Analyse the data to “3rd normal form”
- Consider uniquely identifiable attributes as entities
- Study the rules that are pre and post conditions of process steps
- Consider the same rules as constraints on relationship cardinalities
- Consider (and name) an association from both ends
- Look for constraints in triangles - remove redundant relationships
- Look for link entities to resolve N-to-N associations
- Look for redundancy in 1-to-1 associations
- Look to separate a type from its instances
- Look for constraints in double V associations
- Consider recording enquiry interactions as well as updates
- Consider how data will be serialised into a required data flow
- Do access path analysis
- Denormalise for faster enquiry/report processes
- Consider the wider and longer-term perspective
- Establish data history and versioning requirements
- Beware class hierarchies in models of persistent data
- Use the business’s natural primary keys as a guide
- Consider exclusion arcs in place of subtypes
- Don’t invent super types just because you can
- Minimise multiple inheritance
- Don’t invent concepts you don’t need
- Don’t map all class hierarchies to tables in the same way
- Consider separating type from instance
- Consider roles in place of sub types

Part one: basics

- Model with a purpose
- Analyse the data in required I/O data flows
- Analyse the data to “1st normal form”
- Analyse the data to “3rd normal form”
- Consider uniquely identifiable attributes as entities
- Study the rules that are pre and post conditions of process steps
- Consider the same Rules as constraints on relationship cardinalities
- Consider (and name) an association from both ends
- Look for constraints in triangles - remove redundant relationships
- Look for link entities to resolve N-to-N associations
- Look for redundancy in 1-to-1 associations
- Look to separate a type from its instances

Model with a purpose

- Do not get carried away with “conceptual” modelling
- You are not modelling the “real world” for its own sake
- You are only modelling that tiny, tiny, tiny part of the real world that your business needs to monitor and perhaps control

1. Analyse the required input and output data flows

1 Analyse required I/O

2 Analyse data stores

3 Build a data dictionary

4 Define data entities

5 Analyse relationships

6 Validate the model

7 Facilitate access paths

1. Define the outputs the system must supply

Project team
Project Number, Project Description
Employee Number, Name
Employee Number, Name
Employee Number, Name

Monthly Invoice/Receipt
Invoice Number, Credit Card Num
Order Number, Order Value
Order Number, Order Value
Invoice Total

2. Define data necessarily collected from external entities

Order
Order Number, Order Value, Credit Card Num
Order Item 1, Product Number, Quantity,
Order Item 2, Product Number, Quantity

Analyse the data in required I/O data flows

- The LDM should contain the data needed to produce outputs, and access paths that enable known transactions and queries.

CUSTOMER entity	
Customer number	**Primary Key**
Customer name	
Customer address	

PRODUCT entity	
Product type	**Primary Key**
Product description	
Unit price	

ORDER entity	
Customer number	**Foreign Key**
Order number	Primary Key
Order amount	
Product type	**Foreign Key**

Data analysis

Order History SEQUENCE
- **Customer number**
- **Customer name**
- **Customer address**
- Set of Orders ITERATION
  - Order SEQUENCE
    - **Order number**
    - **Order amount**
    - **Product type**
    - **Product description**
  - Order END
- Set of Orders END

Order History END

2. Analyse the data stores people refer to

1 Analyse required I/O

2 Analyse data stores

3 Build a data dictionary

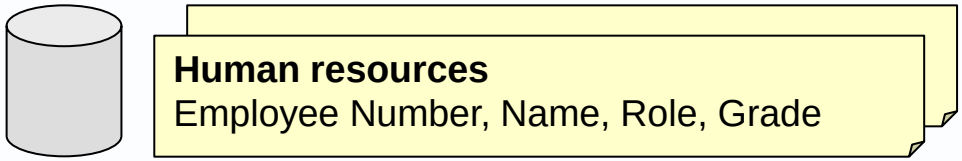
4 Define data entities

5 Analyse relationships

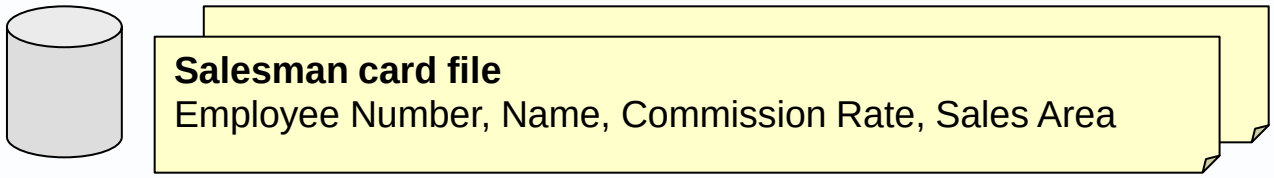
6 Validate the model

7 Facilitate access paths

- The HR department maintains a spreadsheet of all employees



- The sales manager has a card file with all salesmen in it



3. Build a data dictionary

1 Analyse required I/O

2 Analyse data stores

3 Build a data dictionary

4 Define data entities

5 Analyse relationships

6 Validate the model

7 Facilitate access paths

1. Define data items in input and output data flows
2. Define data items that must persist to make outputs possible
3. Define business rules associated with data items

Term	Currency Code
Facts	Currency Code [abbreviates] Currency [which denominates a] Value
Constraints	Currency [is a] three letter String [in the range] defined at ….
Derivation	
Term	**Item Value**
Facts	Item Value [is an attribute of an] Order Item Item Value [is associated with a] Currency
Constraints	Item Value [is a] Decimal Number [in the range] 000.00 to 999.99
Derivation	Item Value = Product Amount Ordered * Unit Price
Term	**Order Value**
Facts	Order Value [is an attribute of] Order Order Value [is calculated from] Item Values
Constraints	Order Value [is a] Decimal Number [in the range] 0000.00 to 9999.99
Derivation	Order Value = sum of (Item Values for an Order) - Discount

4. Define the data entities

Avancier

1 Analyse required I/O

2 Analyse data stores

3 Build a data dictionary

4 Define data entities

5 Analyse relationships

6 Validate the model

7 Facilitate access paths

- Look for primary keys used in the business
- Separate a repeating group into a detail entity
- Consider raising an attribute to become an entity

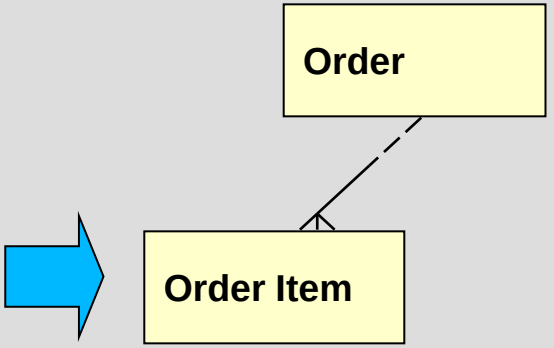
Identifiers reveal entity types

- Every time you come across an identifier range (be it names or numbers) used by humans,
- you have found an entity type that is important to the humans who invented and use those identifiers.
- *This has nothing at all to do with computers*

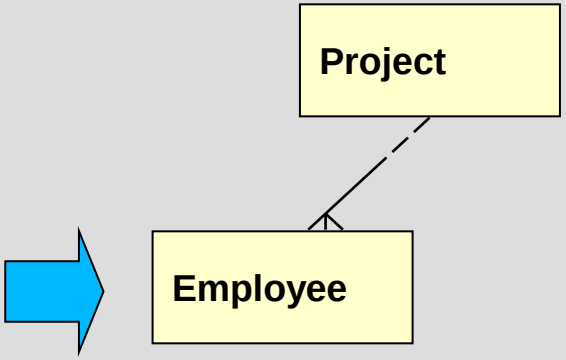
e.g.

- 4.1 Look for primary keys used in the business
- 4.2 Separate a repeating group into a detail entity

Order
Order Number, Order Value, Credit Card Num
Order Item 1, Product Number, Quantity,
Order Item 2, Product Number, Quantity



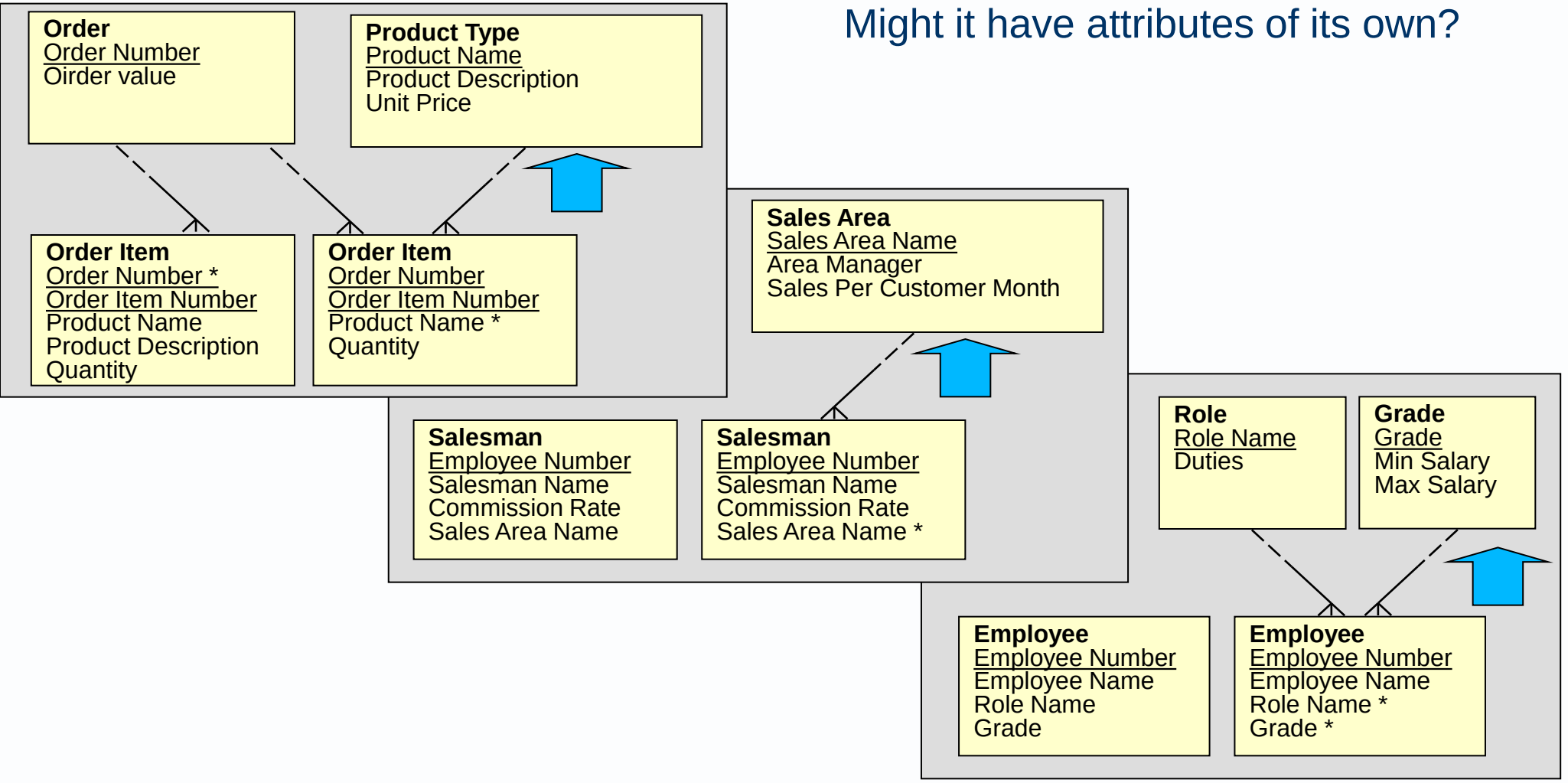
Project team
Project Number, Project Description
Employee Number, Name
Employee Number, Name
Employee Number, Name



4.3 Consider raising an attribute to become an entity

Is an attribute significant as an entity in its own right?

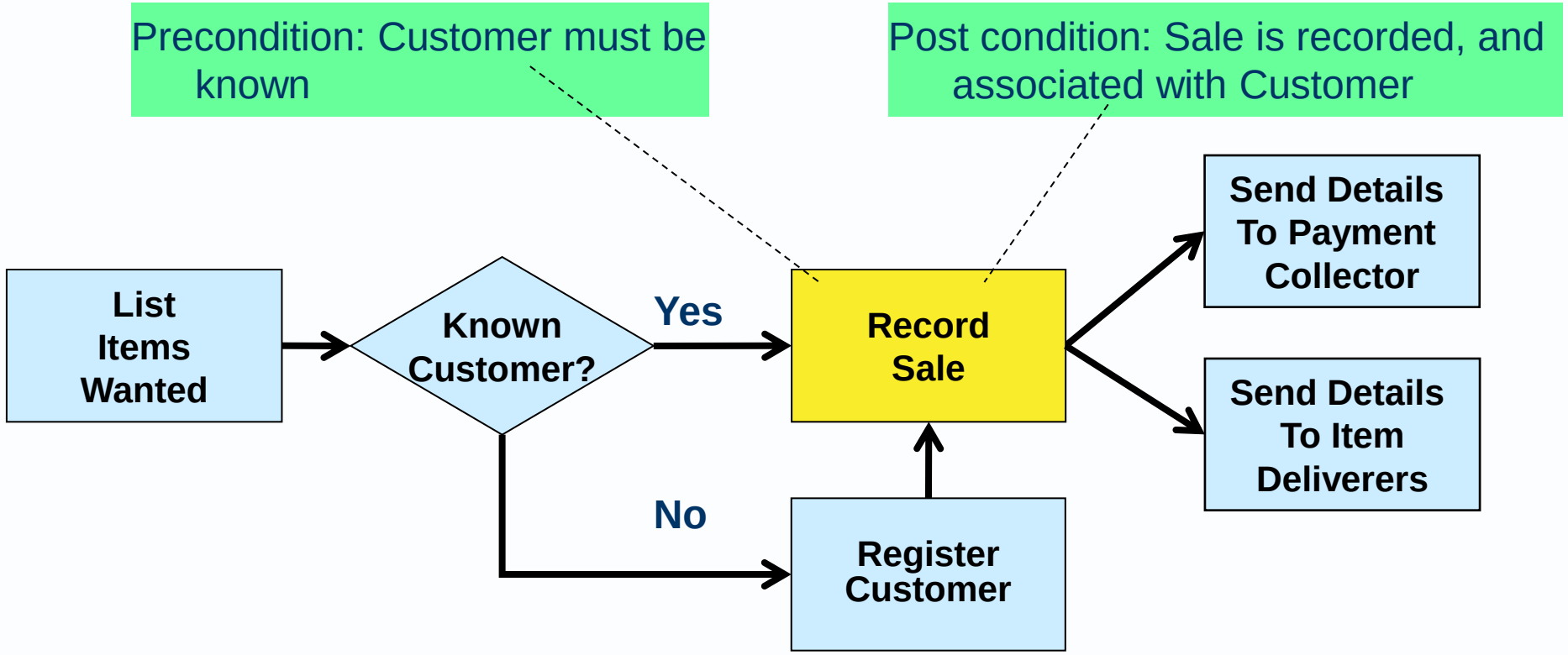
Might it have attributes of its own?



Consider uniquely identifiable attributes as entities

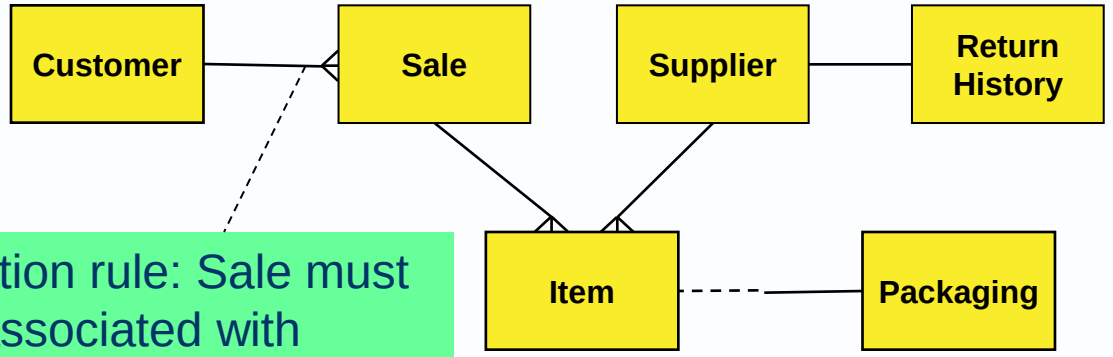
- Whether a recorded fact is considered to be an entity or attribute depends on the system domain and the needs of the system's users.
- Anything given unique names or numbers by the system's users can be regarded as entities
  - Customers have unique customer numbers
  - Cheques have unique cheque numbers
- Facts whose values may be duplicated without care are attributes of entities.
  - Customer addresses
  - Cheque amounts
- But where you want to analyse data using the value of an attribute then those values might be treated as primary keys
  - Customer religion
  - Product colour

Study rules that are pre and post conditions of process steps



Consider the same Rules as constraints on relationship cardinalities

- Associations often specify constraints on which entities relate to which other entities



Association rule: Sale must be associated with Customer

This diagram uses the CACI notation..

- (This CACI style of association line shows optionality using dotted lines and cardinality using the presence or absence of a crow's foot)

5. Analyse the relationship between two entity types

Avancier

1 Analyse required I/O

2 Analyse data stores

3 Build a data dictionary

4 Define data entities

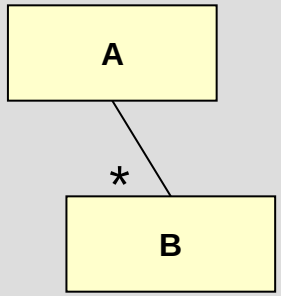
5 Analyse relationships

6 Validate the model

7 Facilitate access paths

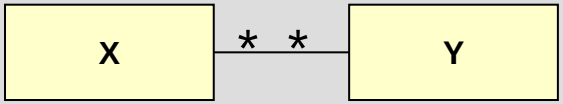
See Universal Modelling Principles (UMP) to follow

Consider an association from both ends



- Does entity A relate to more than one B? Yes
- Does entity B relate to more than one A? No

- Does entity X relate to more than one Y? Yes

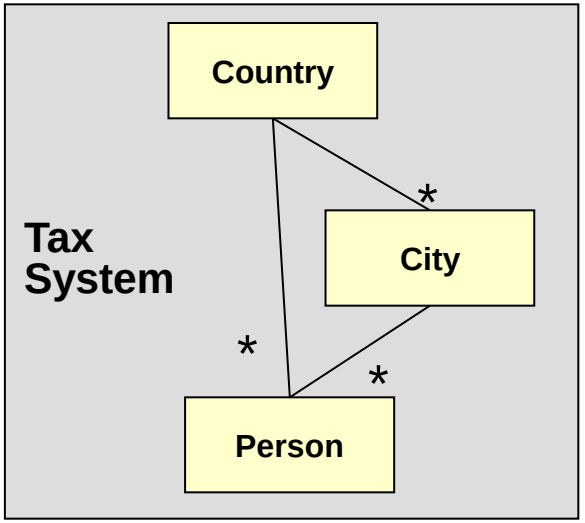


- Does entity Y relate to more than one X? Yes

Look for constraints in triangles

- Does one Country contain more than one City? Yes
- Is one City located in more than one Country? No

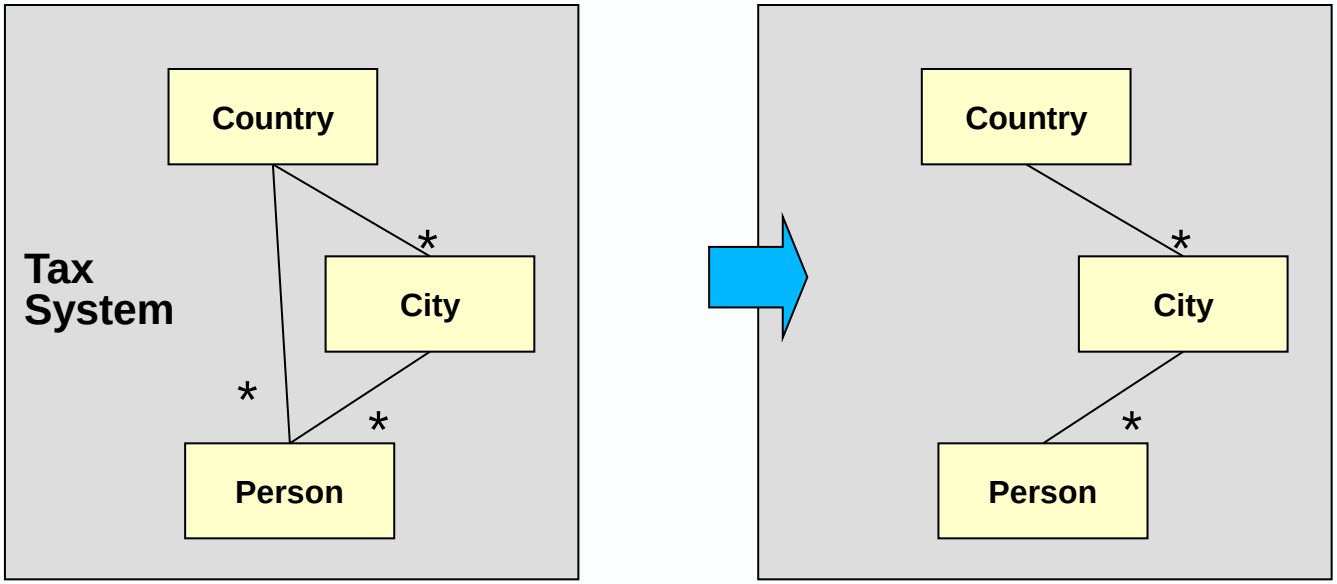
- Is one City the residential location of more than one Person? Yes
- Does one Person reside in more than one City? No



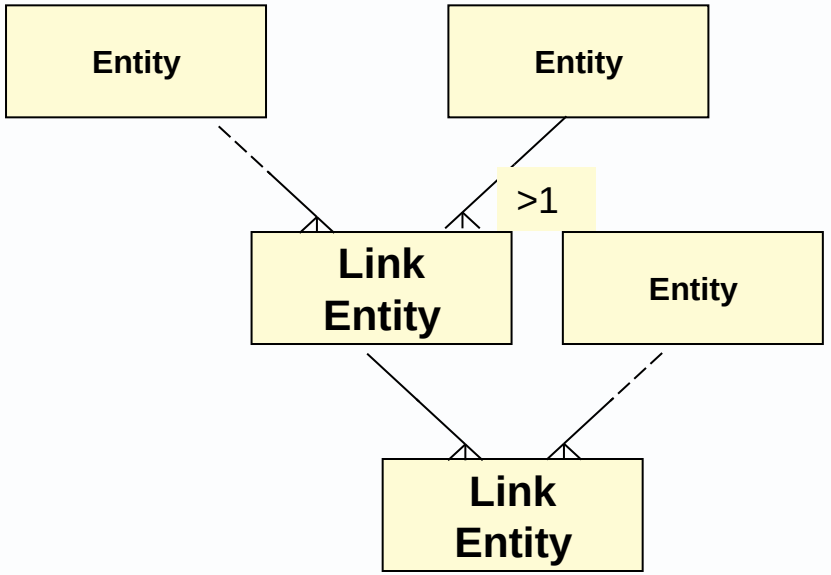
- Does one Country tax more than one Person? Yes
- Does one Person pay tax in more than one Country? No

Look for constraints in triangles - remove redundant relationships

- Are the same Persons found down both long and short relationships? Yes (say)



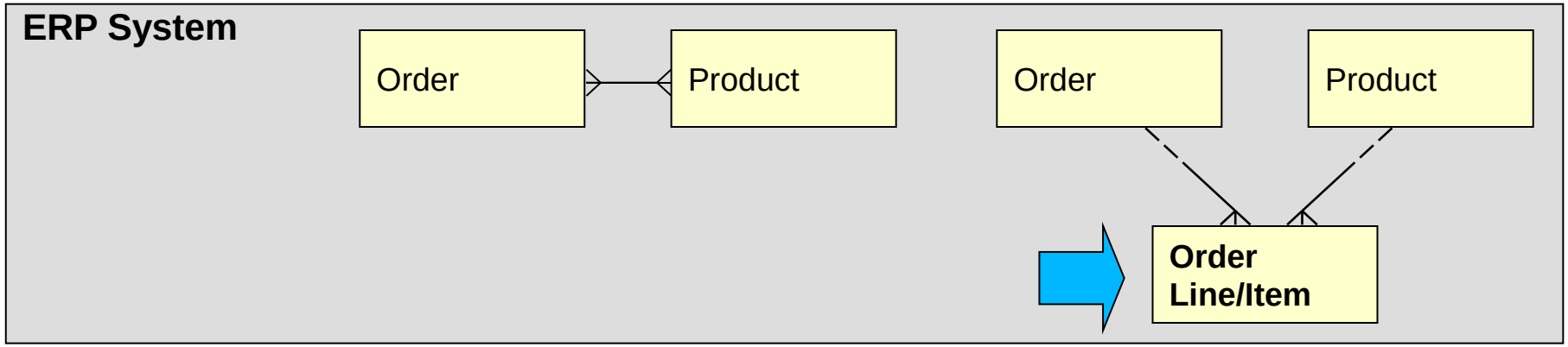
Link Entities

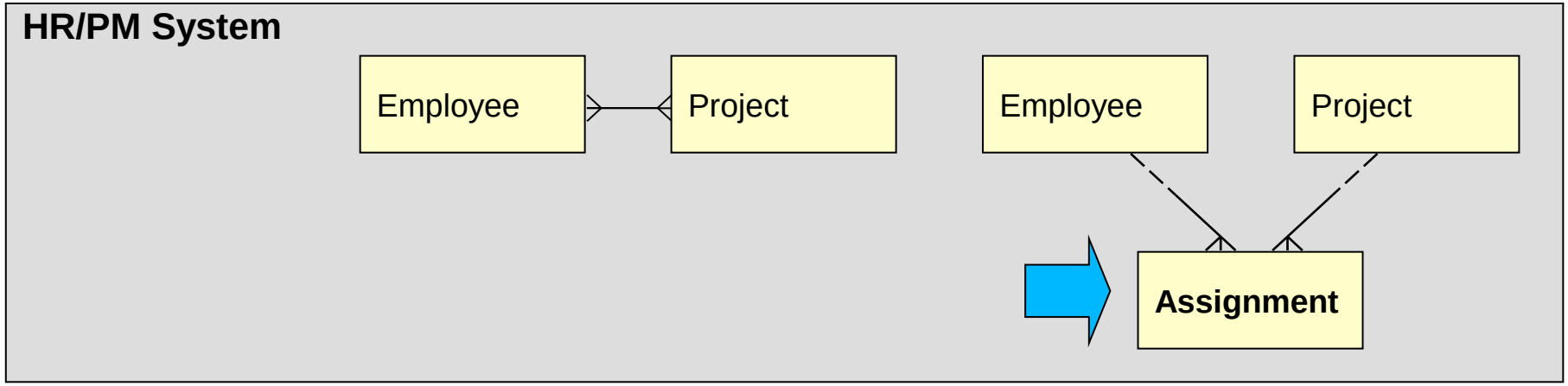


Look for link entities to resolve N-to-N associations

Avancier

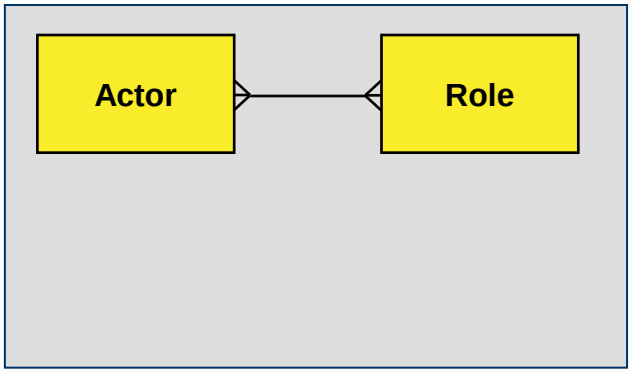
What is the name of the event or the thing that relates one of one entity to one of the other entity?

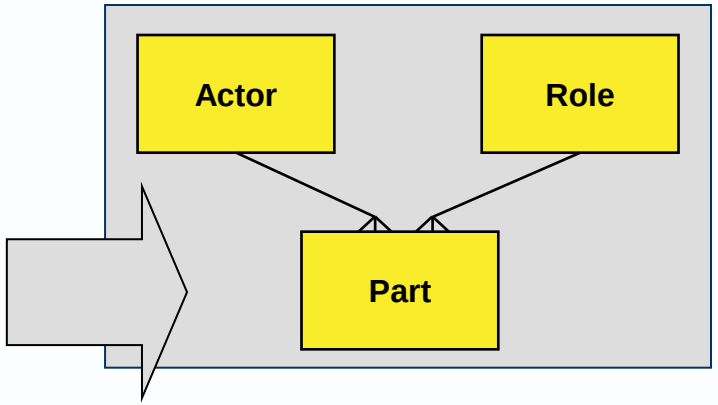


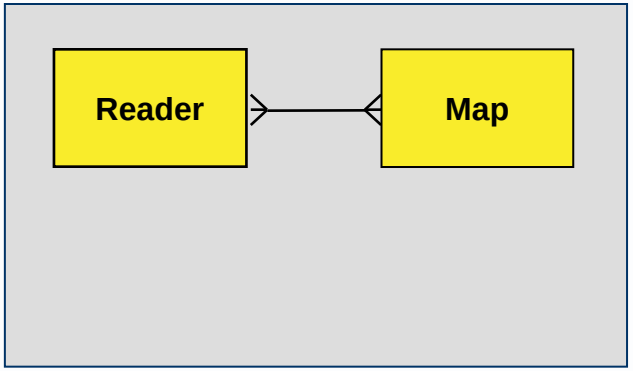


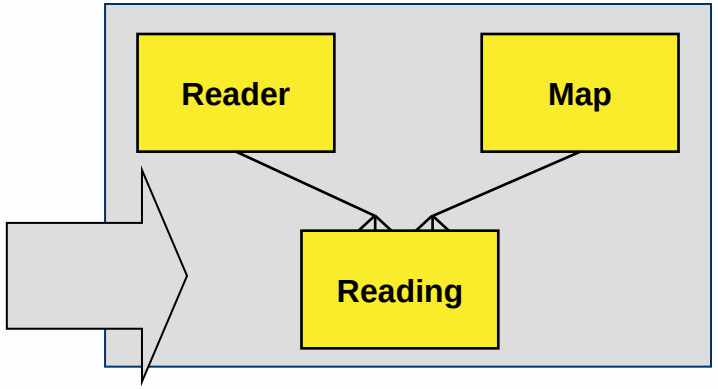
Look for link entities to resolve N-to-N associations

- Is there an entity that links one of one to one of the other?





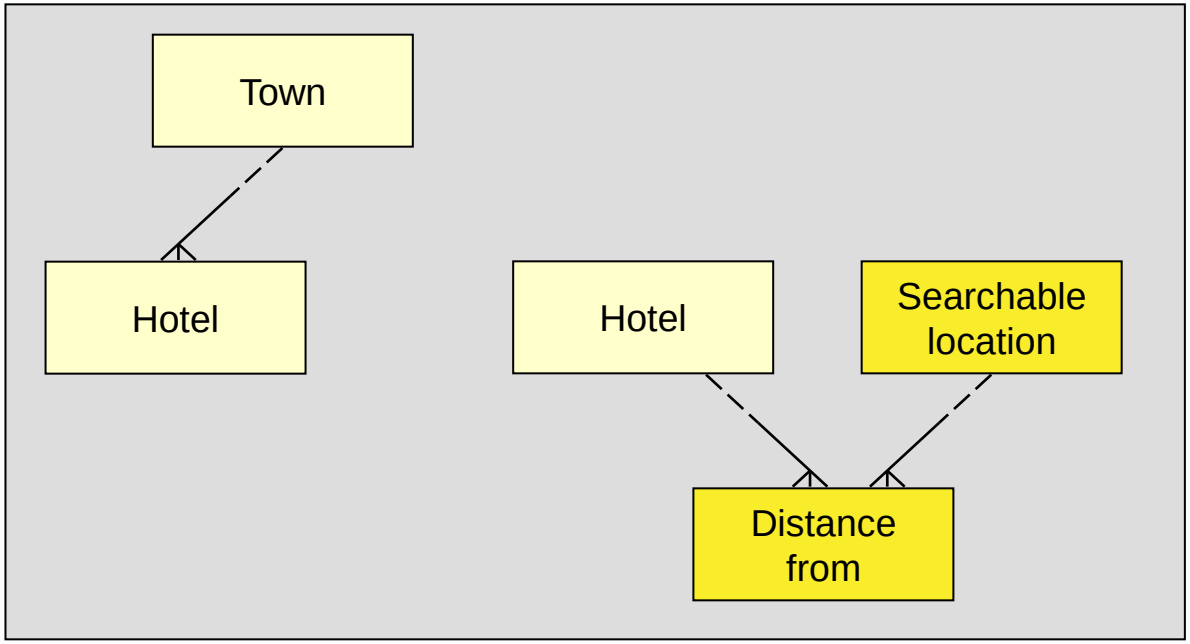




Look for link entities to resolve N-to-N associations

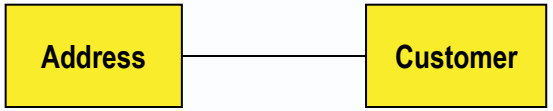
Avancier

e.g. a more flexible search



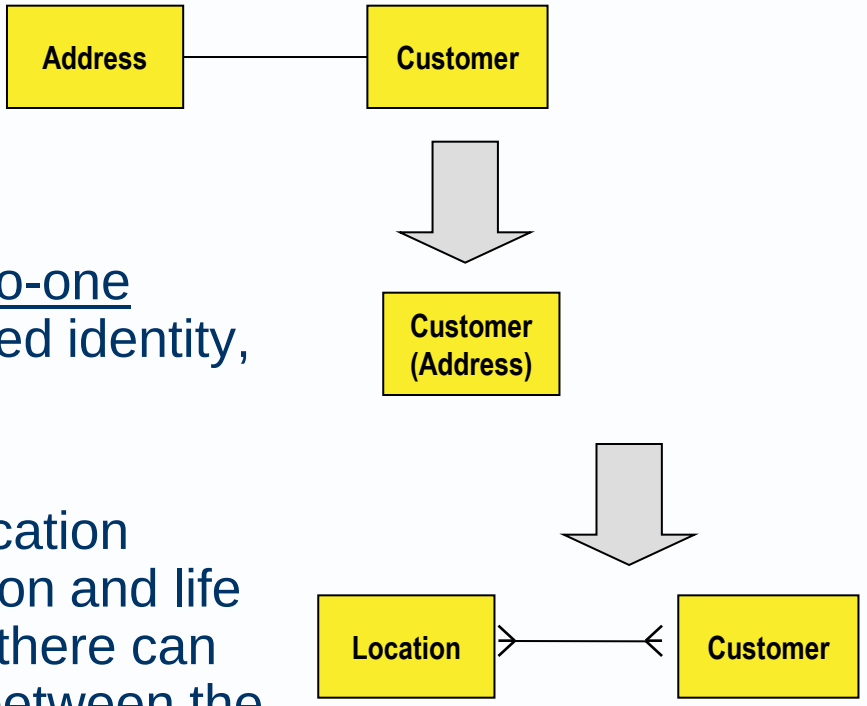
Look for redundancy in 1-to-1 associations

- Suppose we invent an identifier for an address that has a different value from the identifier of the customer that has that address?



- Do we have two entities in 1 to 1 correspondence? Probably not

Look for redundancy in 1-to-1 associations

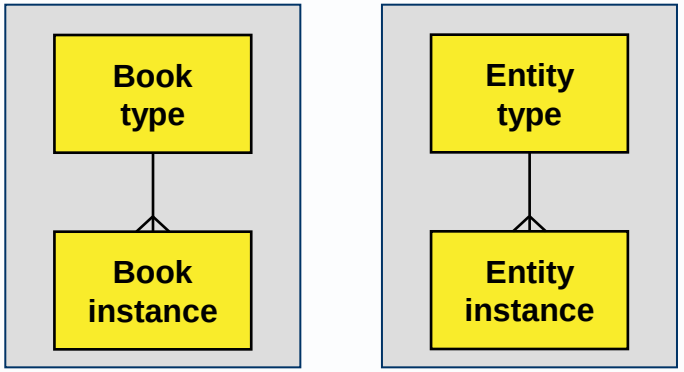


- EITHER the two identifiers are in one-to-one association, so represent a single shared identity, and there is only one entity.
- OR, the address turns into a distinct location entity, with its own properties, description and life history – separate from customer, and there can be an N-to-N association relationship between the two entities.
- *This has nothing at all to do with computers*

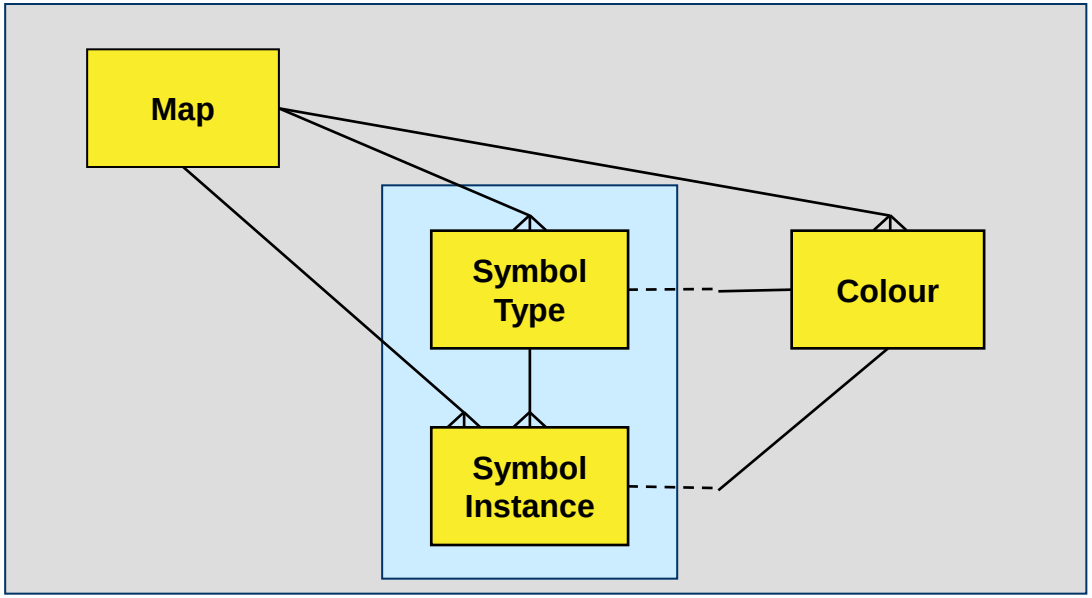
Look to separate a type from its instances

Narrative text is ambiguous

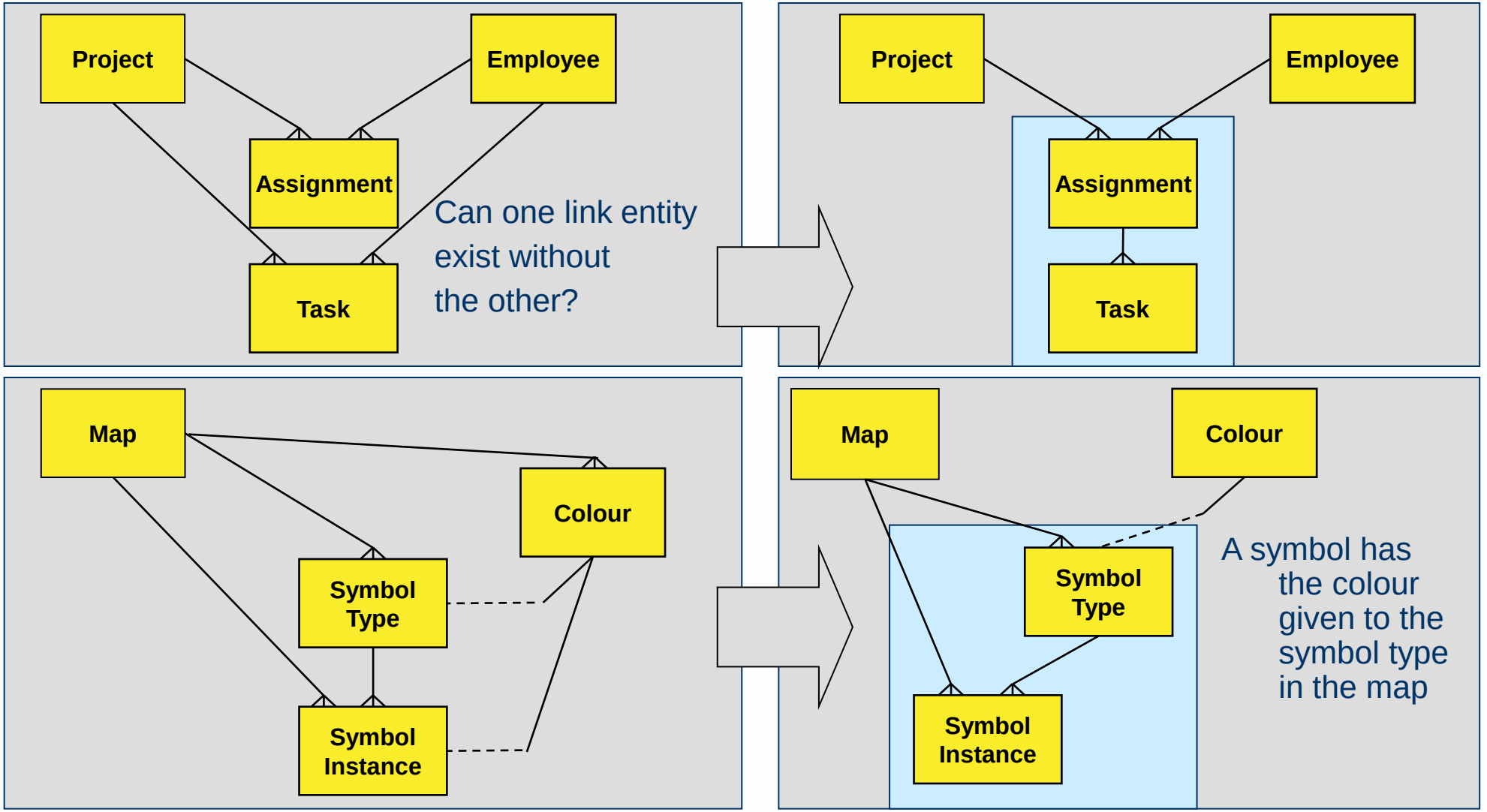
- I publish a book (type)
- I read a book (instance)



- We should separate the symbol types in the key from the symbol instances in the representational part of the map



Look for constraints in double V associations



6: Validate the model

1 Analyse required I/O

2 Analyse data stores

3 Build a data dictionary

4 Define data entities

5 Analyse relationships

6 Validate the model

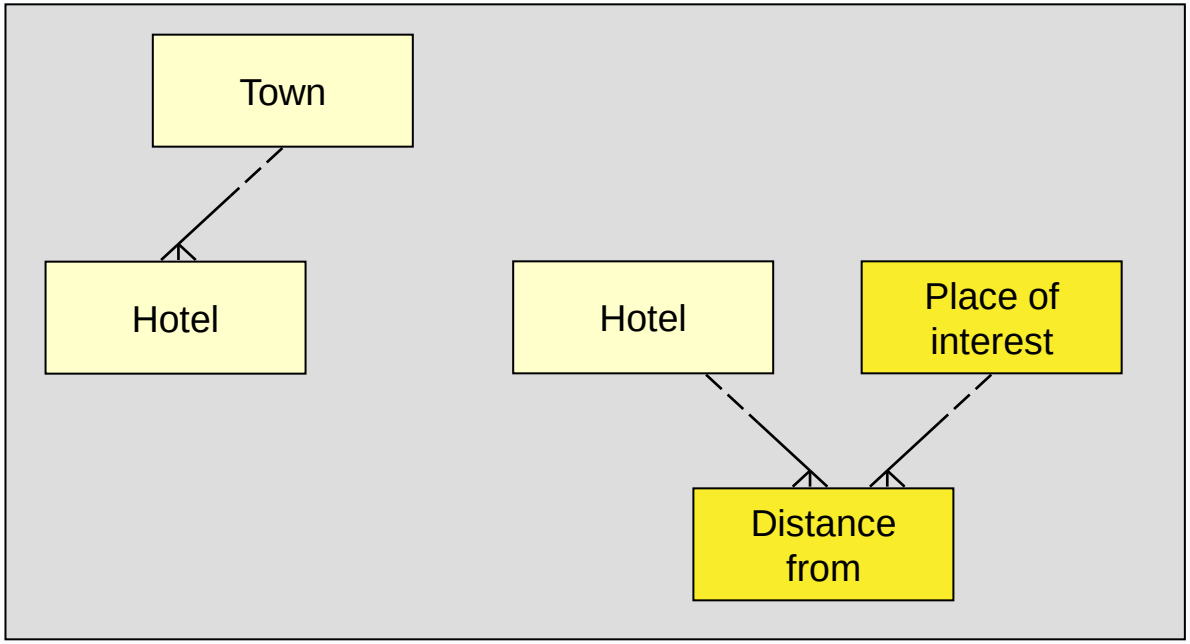
7 Facilitate access paths

1. Is it flexible enough to meet user needs?
   - Perhaps some generalisation is needed?
2. Does it record enough events?
   - Does the business need to remember transient events (e.g. credit and debit transactions) as well as persistent entities (e.g. current accounts)?
3. Does it support processes - performantly?
   - If not, then you may optimise the *physical* data model in various ways

Validation 1: Is it flexible enough to meet user needs?

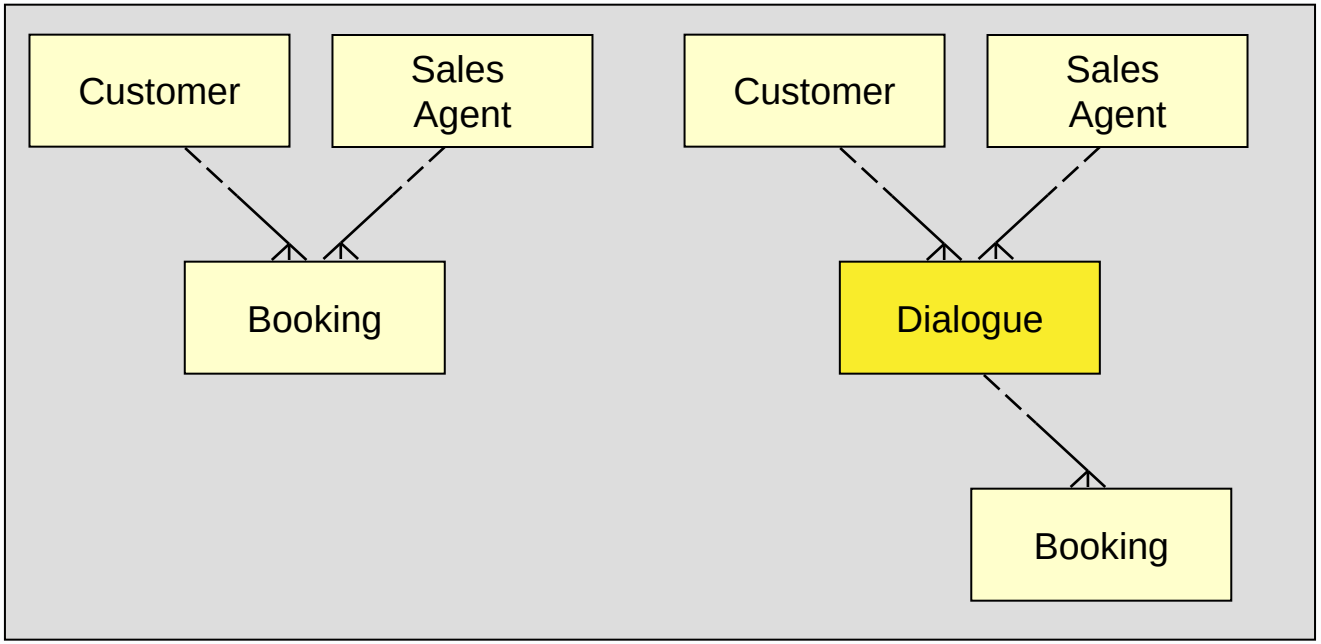
Search only by town?

A more flexible search introduces an N-to-N association



Validation 2: Does it record enough events?

Consider **recording enquiry interactions** as well as updates



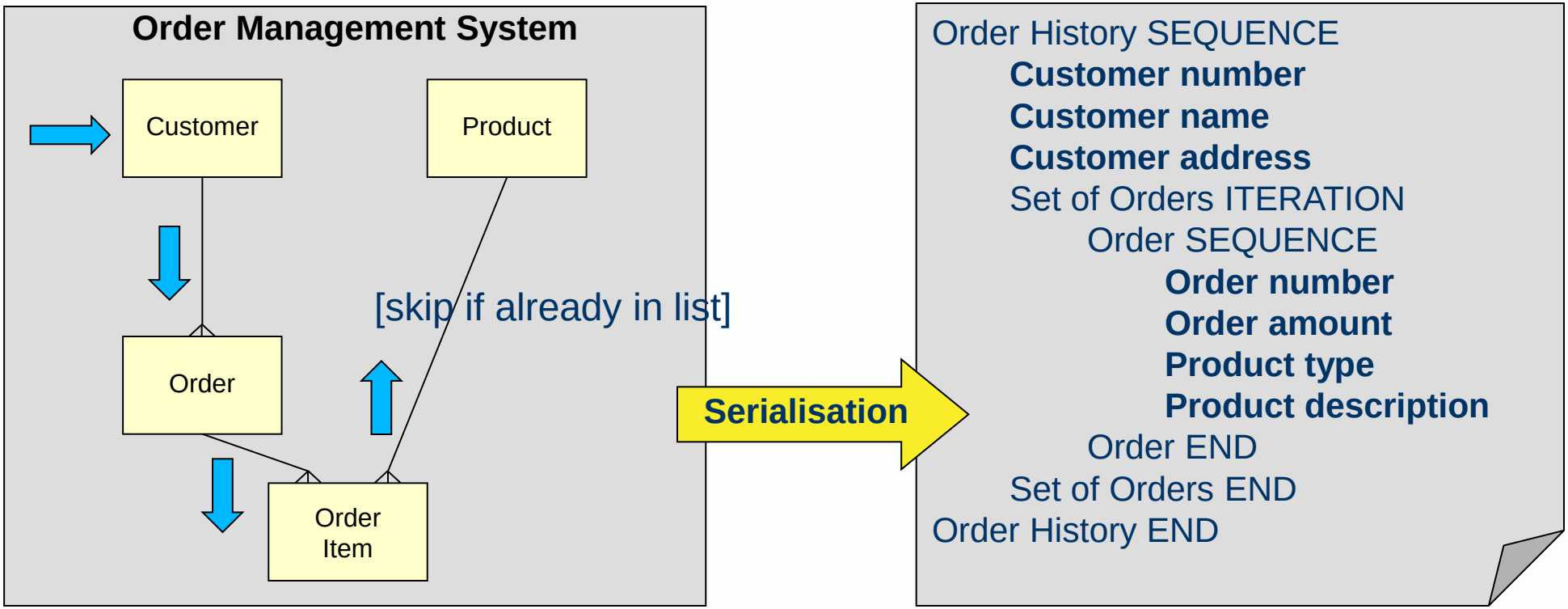
Argos record enquiries for out-of-stock items as well as orders

Validation 3: Does it support processes - performantly?

- Identify processes that are frequent or have long access paths
- Ensure the data model contains data needed by
  - Required business processes
  - Required reports

Do access path analysis – is it feasible?

- "List all the products ordered by a customer"
- First, is it feasible? YES



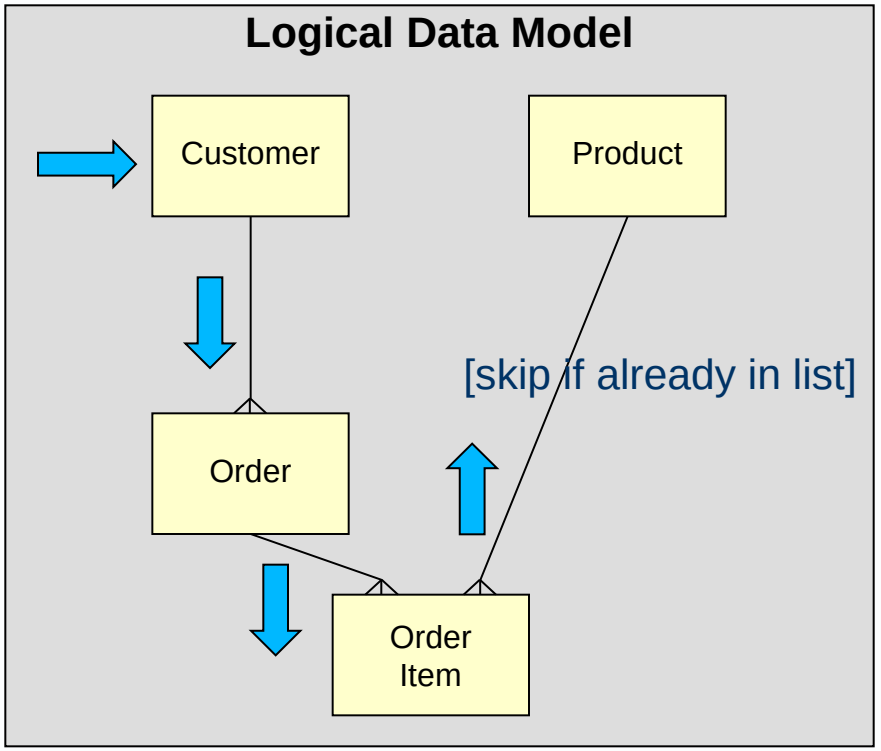
Facilitate access paths

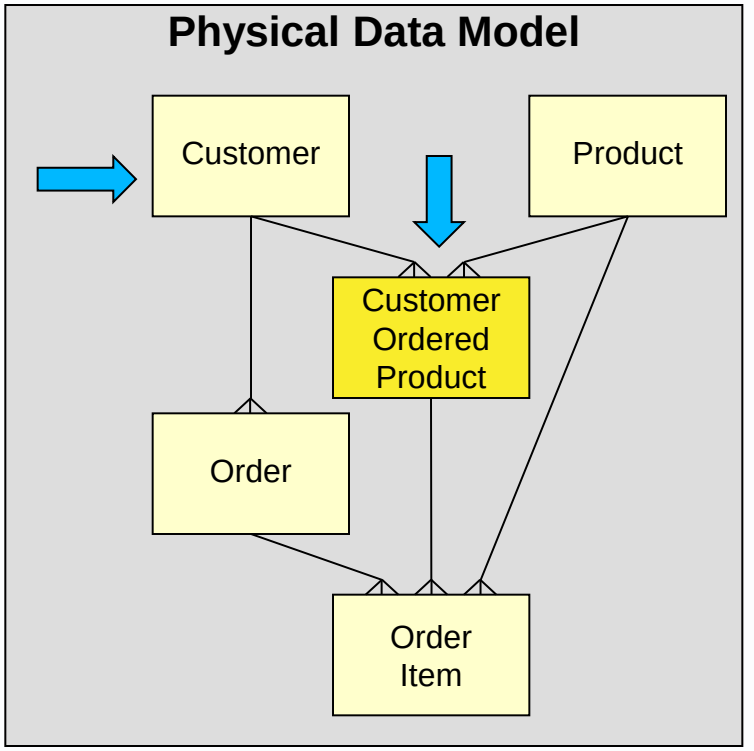
- Make sure critical processes can readily access the data they need
- Do access path analysis
- Add derivable data
- Add derivable relationships
- Otherwise denormalise the physical data model

Do access path analysis – is it fast enough?

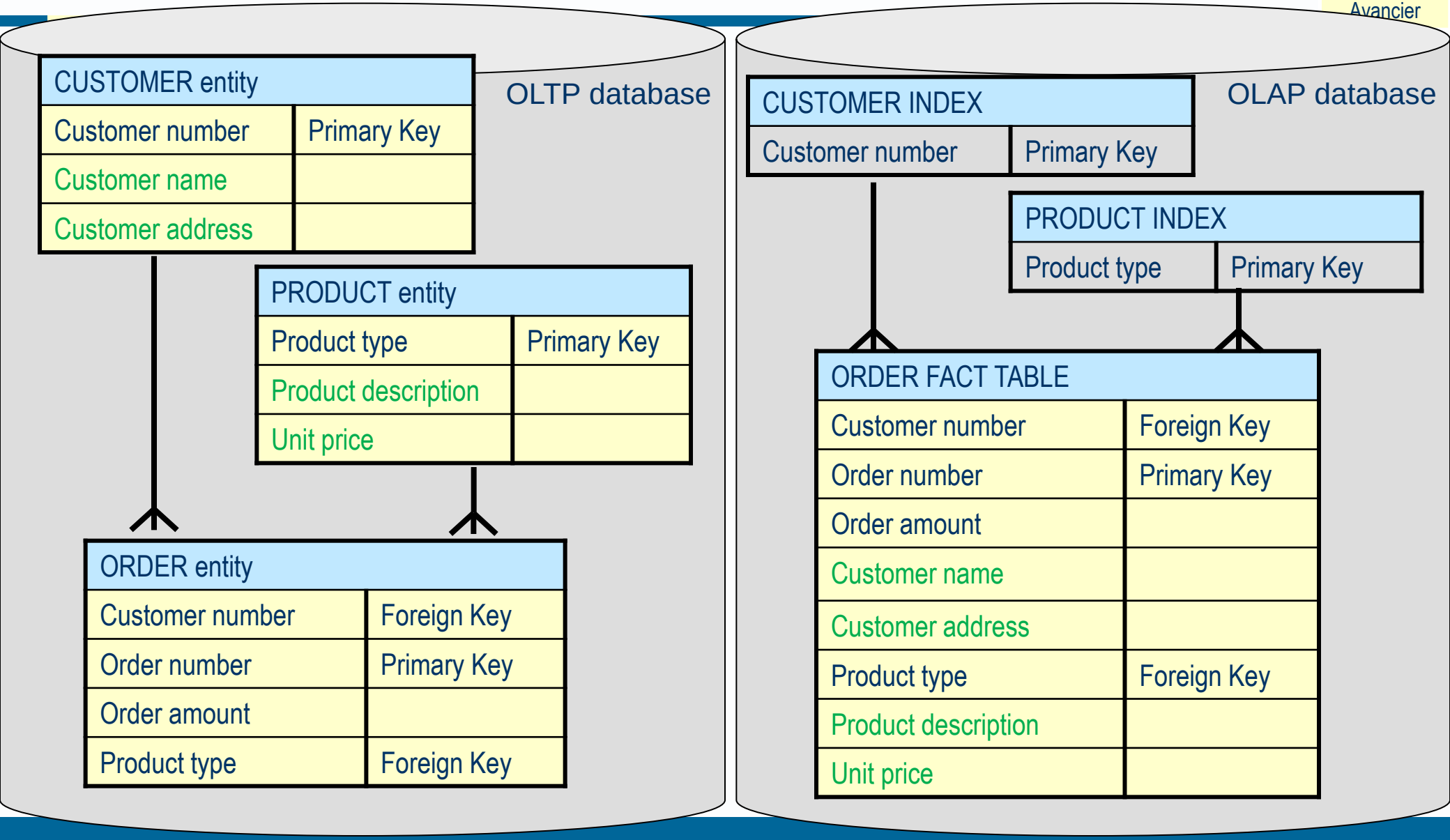
- The typical customer (a school) has placed 1,000 orders, each with 10 items, for our stationery products
- But have ordered only 15 products

- You can remove redundant accesses by adding redundant data or redundant relationships





Denormalisation for faster enquiry/report processes (naive picture)



Part two: advanced

- Consider the wider and longer-term perspective
- Establish data history and versioning requirements
- Beware class hierarchies in models of persistent data
- Use the business's natural primary keys as a guide
- Consider exclusion arcs in place of subtypes
- Don't invent super types just because you can
- Minimise multiple inheritance
- Don't invent concepts you don't need
- Don't map all class hierarchies to tables in the same way
- Consider separating type from instance
- Consider roles in place of sub types

Consider the wider and longer-term perspective

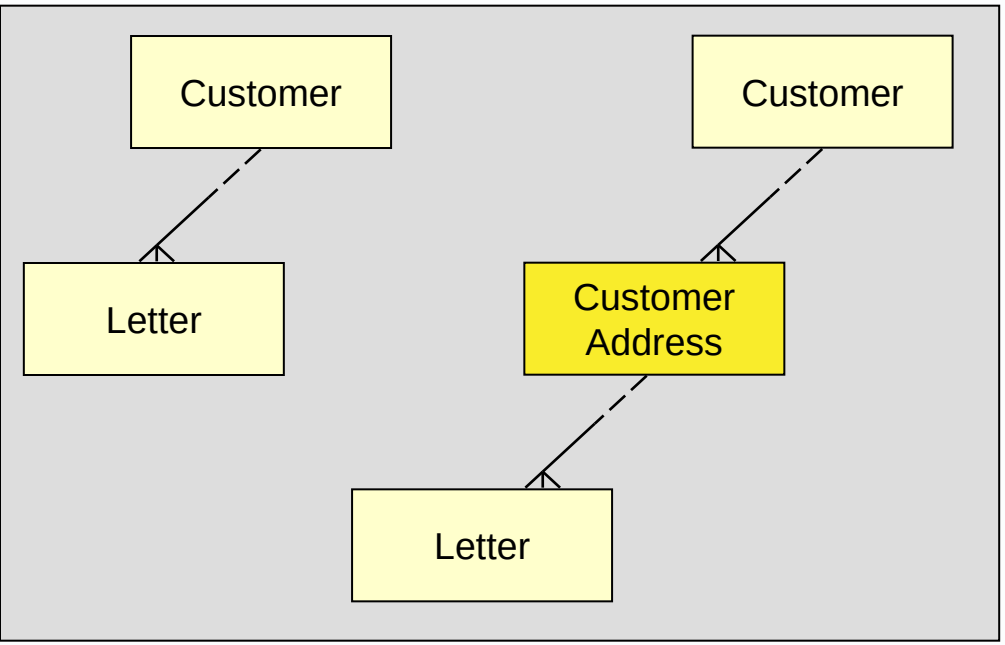
- Define how things change during the behaviour (life history) of entities over time
  - Every entity in a structural model has a life history
  - Entities are not all born at the same time
  - Entities change state
  - Entities become associated with each other
  - Entities break and swap associations with each other
  - For every 1-to-1 association
    - Can A exist without B and vice-versa?
    - Does time change the 1-1 cardinality into 1-to-N?
  - For every 1-to-N association:
    - Is there a time when the Parent has zero or one children?
    - Can a Child be born before the Parent?
    - Does time change the 1-N cardinality into N-to-N?

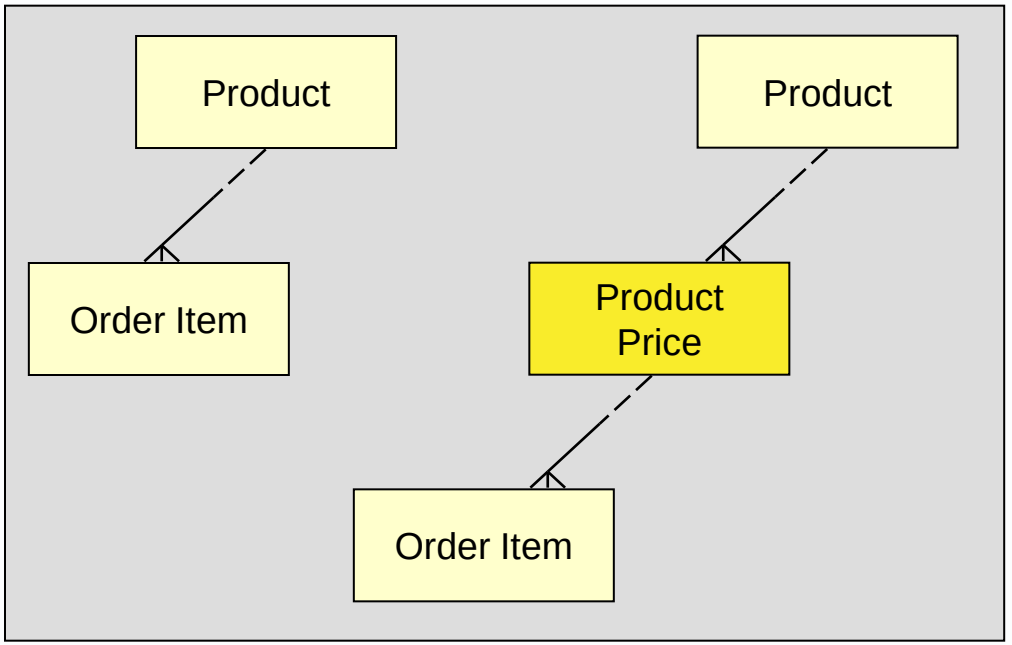
Establish data history and versioning requirements

- Do you need to record a history of attribute values?
  - History of product prices
  - History of customer addresses
- Do you need to record the version number or time span of a value?
- Do you need to record *every* past value?
  - Corrections of mistakes as well as true updates?
- What is the impact of a value change on related entities?
- Do other entities relate to past or current value?
  - Order relates to current or past product price?
  - Order relates to current or and past customer address?

Establish data history and versioning requirements

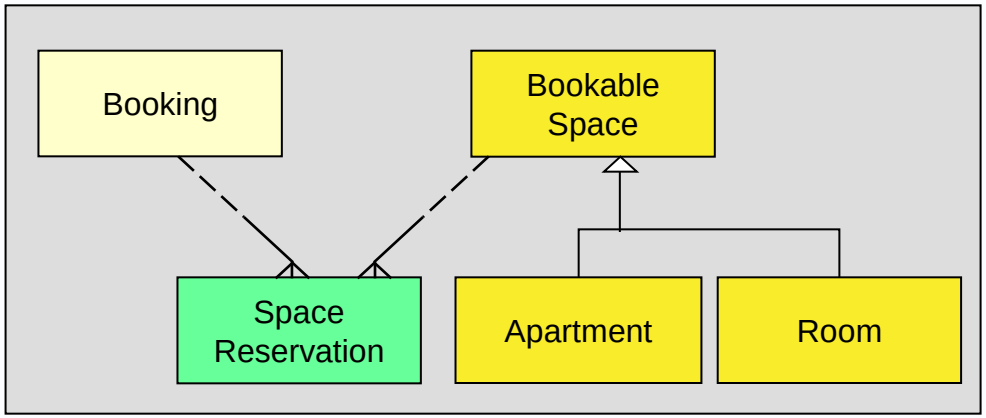
- Do you need to record a history of attribute values?
  - History of product prices
  - History of customer addresses

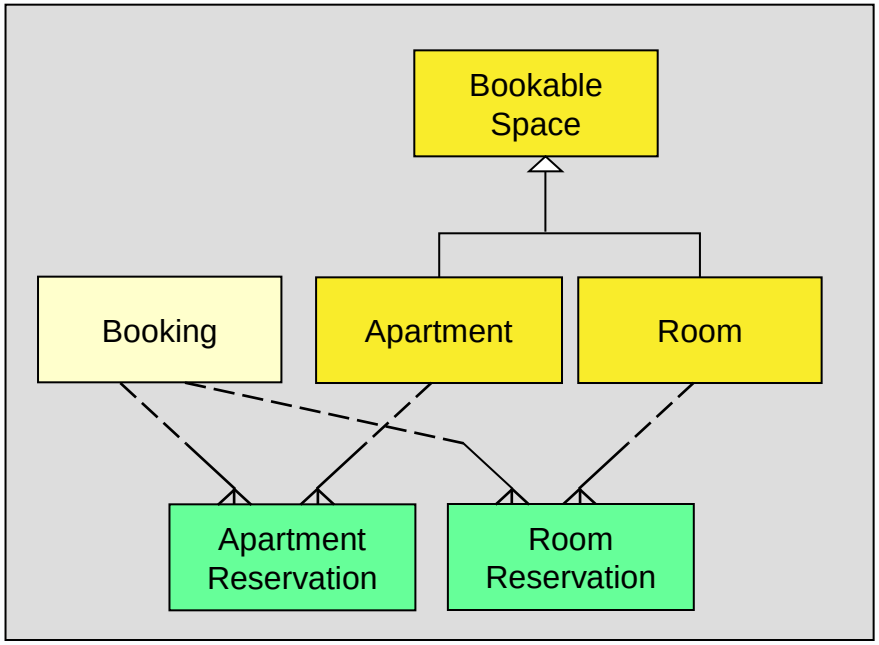


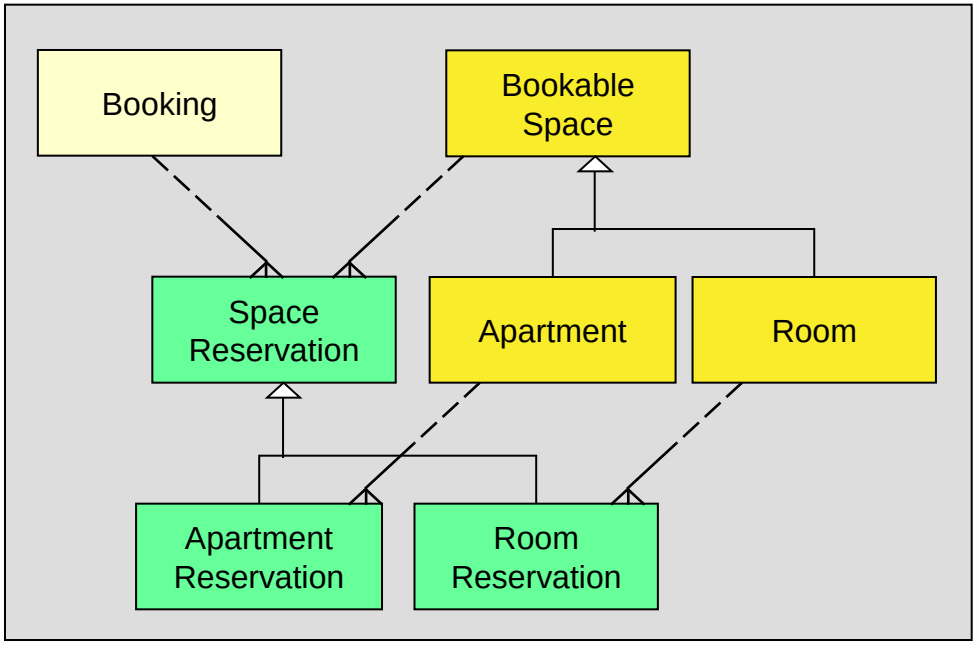


Beware class hierarchies in models of persistent data

- Do you relate to
  - the supertype?
  - the subtypes
  - both?

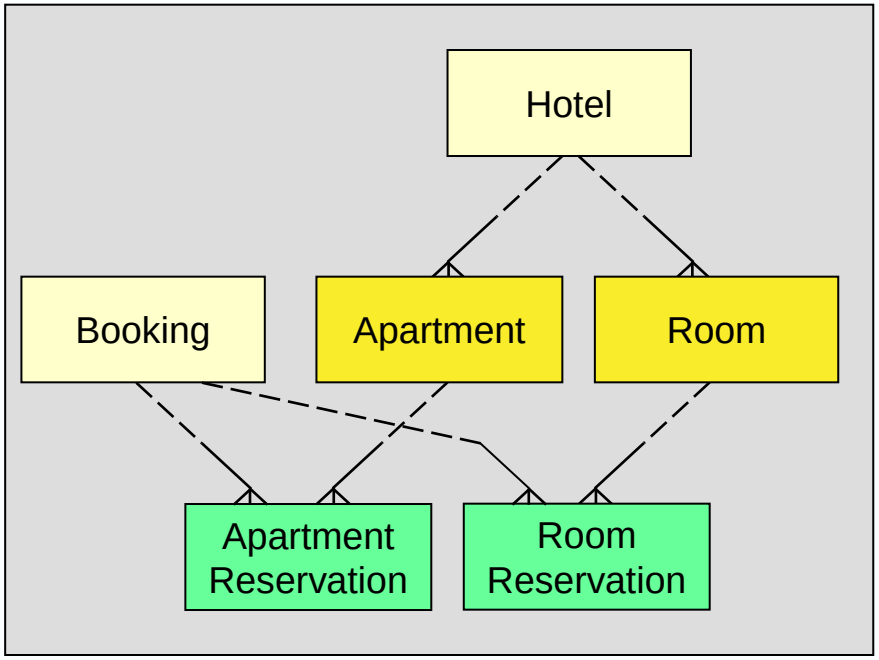




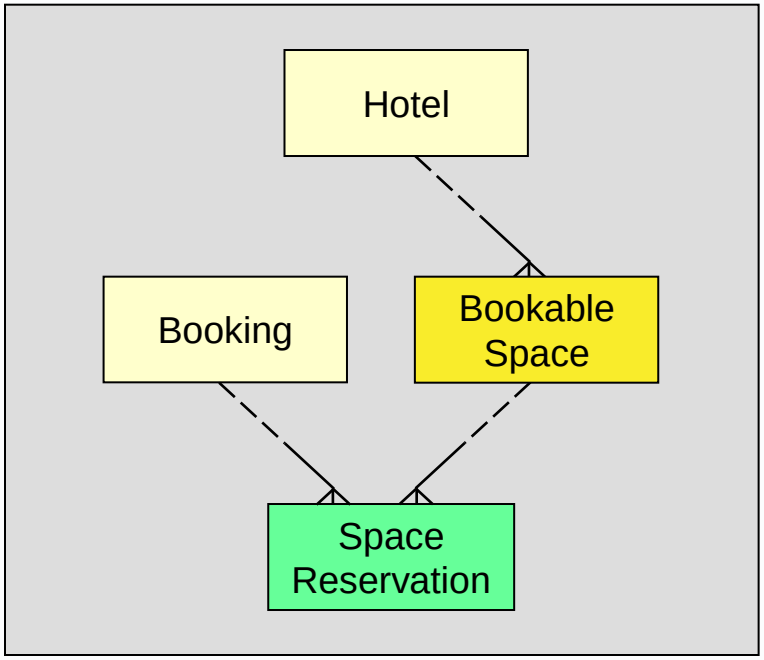


Use the business's natural primary keys as a guide

- Business has different number ranges for apartments and rooms

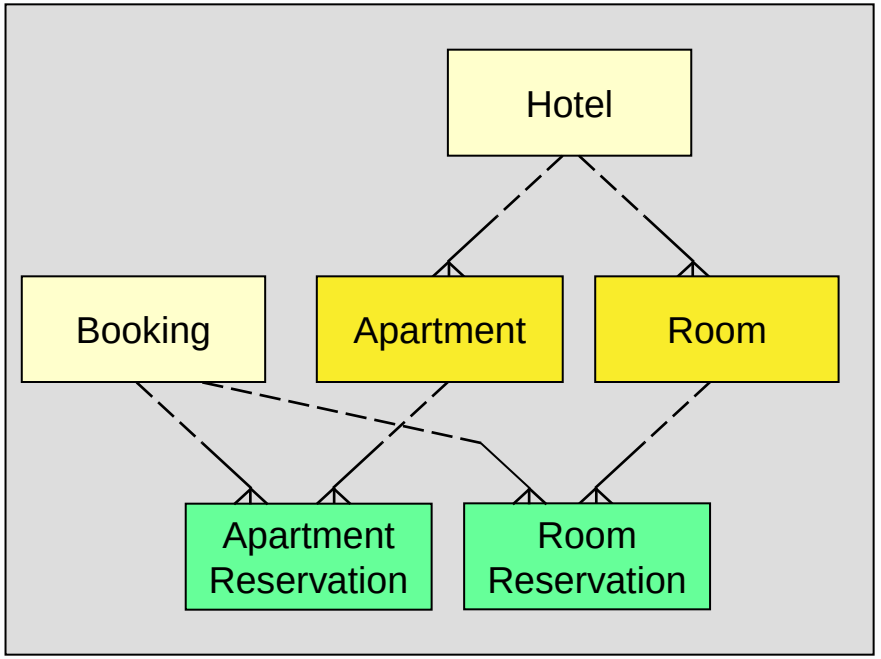


- Business has one number range for apartments and rooms

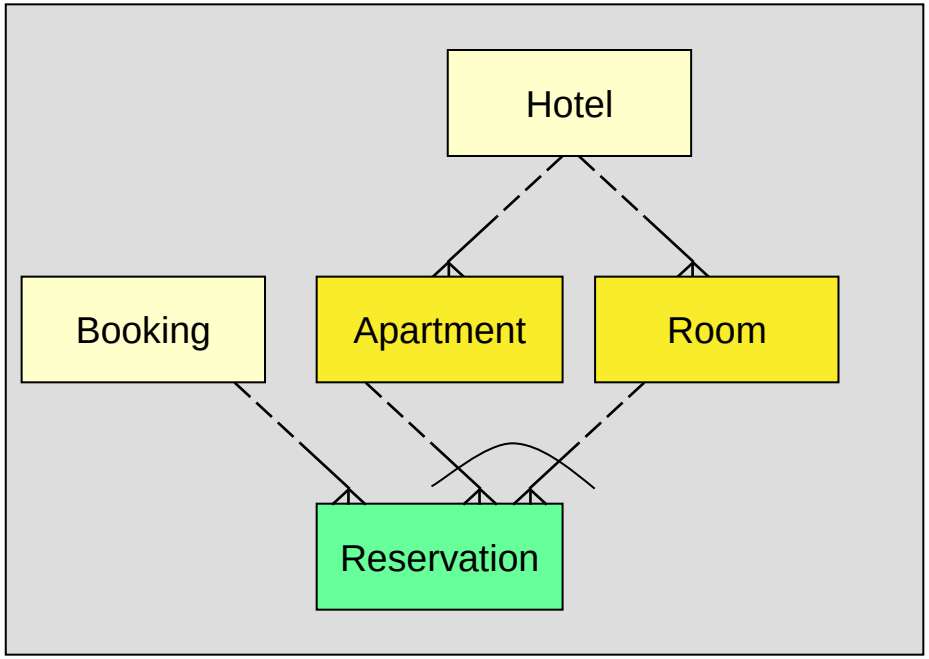


Consider exclusion arcs in place of subtypes

- The original idea



- Simpler data structure?

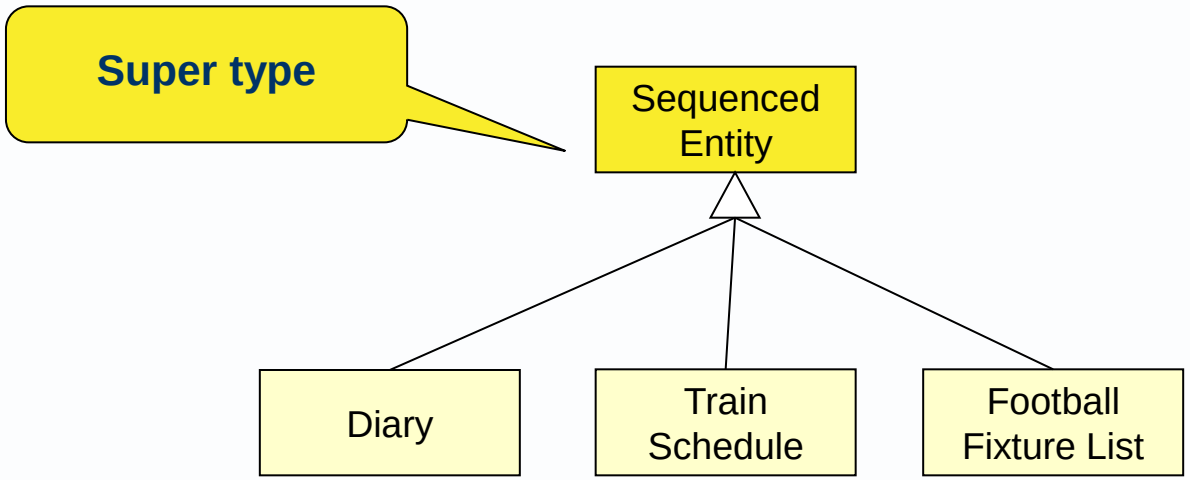


Model with a purpose (again)

- Do not get carried away with “conceptual” modelling
- You are not modelling the “real world” for its own sake
- You are only modelling that tiny, tiny, tiny part of the real world that it is your business to monitor and perhaps control

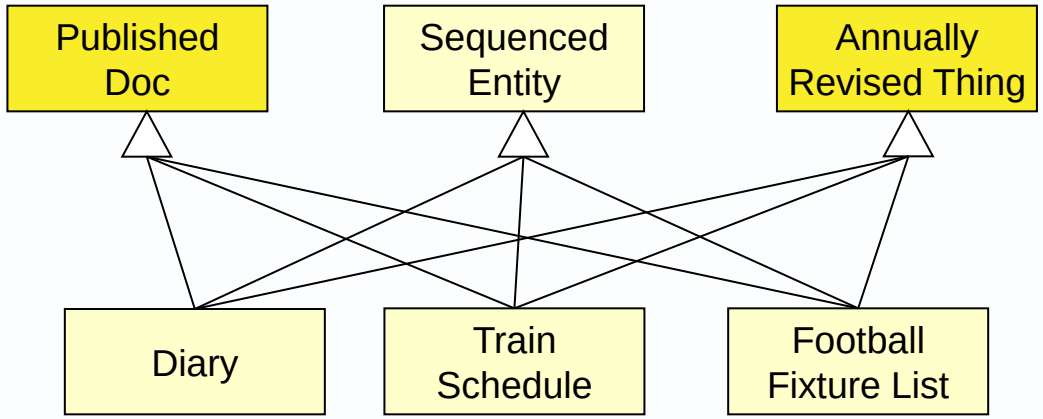
Don’t invent super types just because you can

- Don’t invent super types just because you can



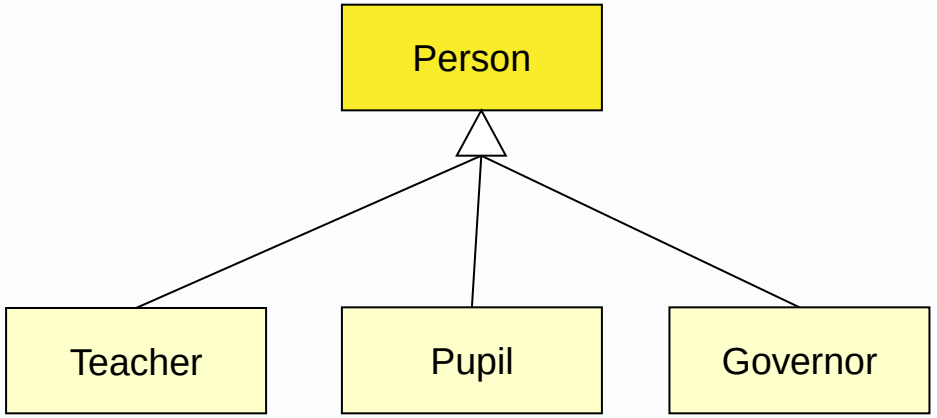
Minimise multiple inheritance

- Multiple inheritance is a fact of life
- But don’t make life harder by creating abstract super types of little or no practical benefit

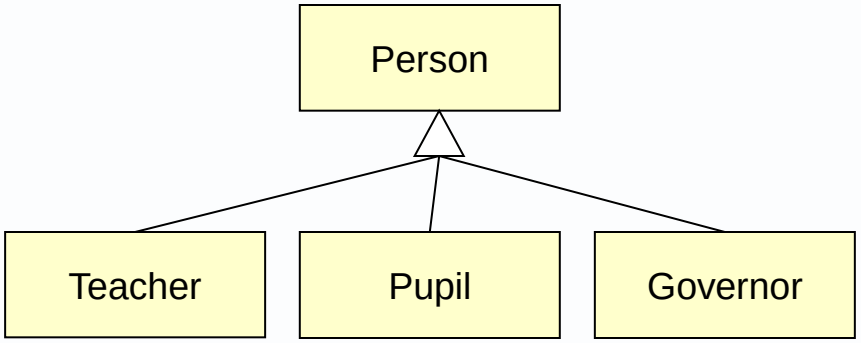


Don’t invent concepts you don’t need

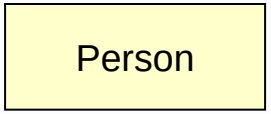
- Have you a way to uniquely a identify a person?
- Are you interested in a person outside of the discrete roles they play?



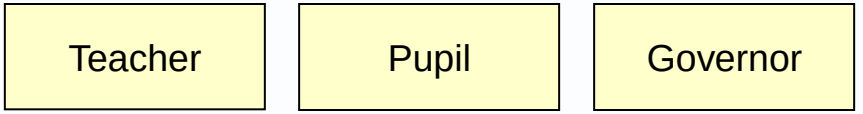
Don’t map all class hierarchies to tables in the same way



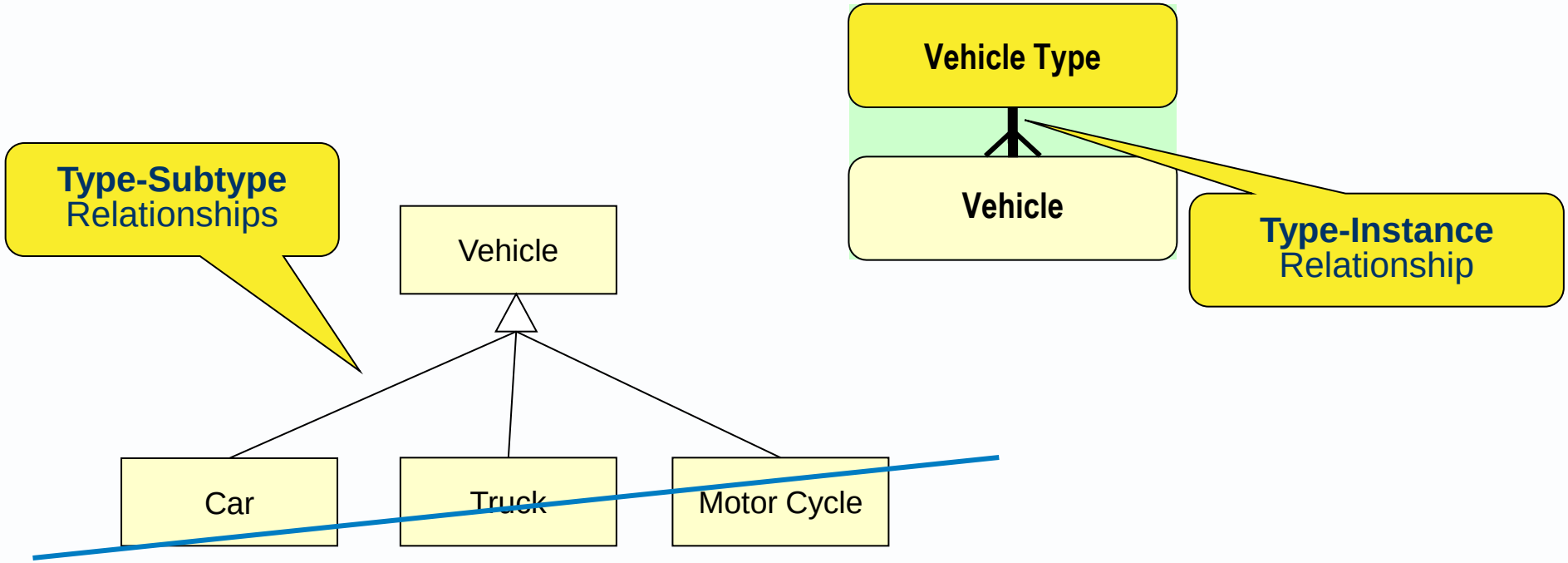
- If there is a unique person identifier, you might roll up into a table for the super type



- Else, probably better to duplicate super type attributes in a table for each subtype

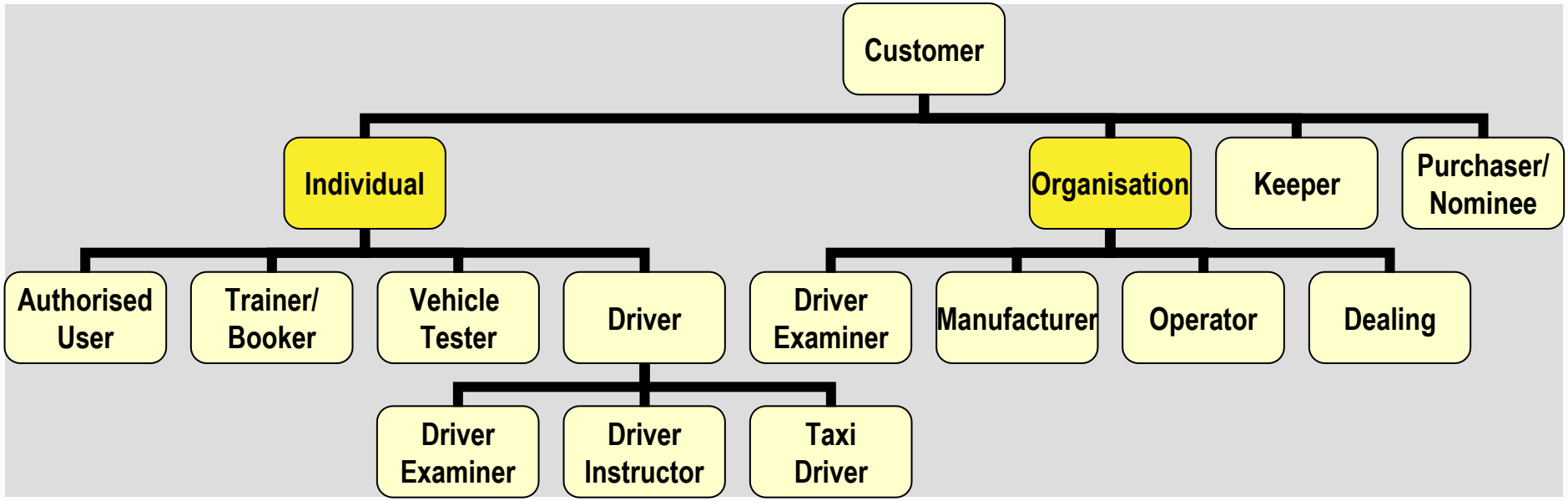


Consider separating type from instance

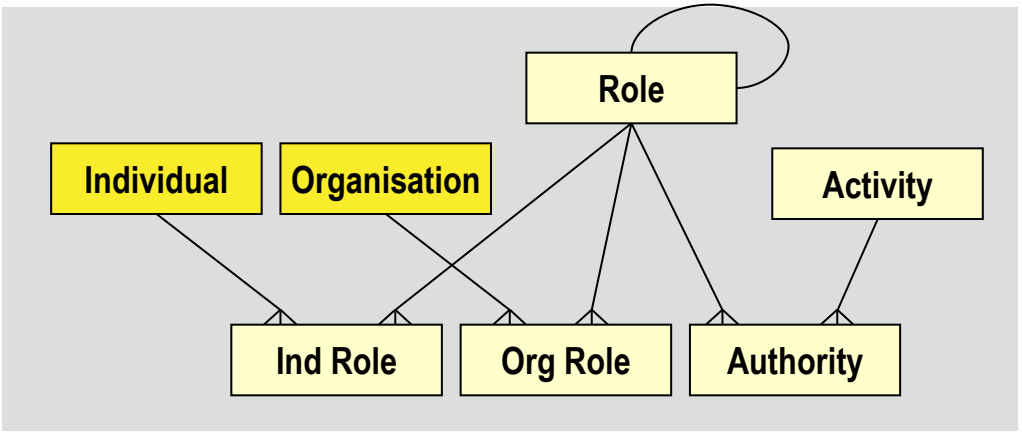


- Subtypes are volatile

Consider roles in place of sub types



- Subtypes may become roles



Strategic data architecture concerns

- Data risk assessment
- Data risk categories
  - Data quality,
  - Data security,
  - Data governance,
  - Data infrastructure,
  - Data complexity,
  - Metadata
- Governance framework
  - Policies
  - Standards & procedures
  - Evidence
- Data quality solutions
  - Manual
  - Automated
  - integrated.