Avancier Methods (AM)

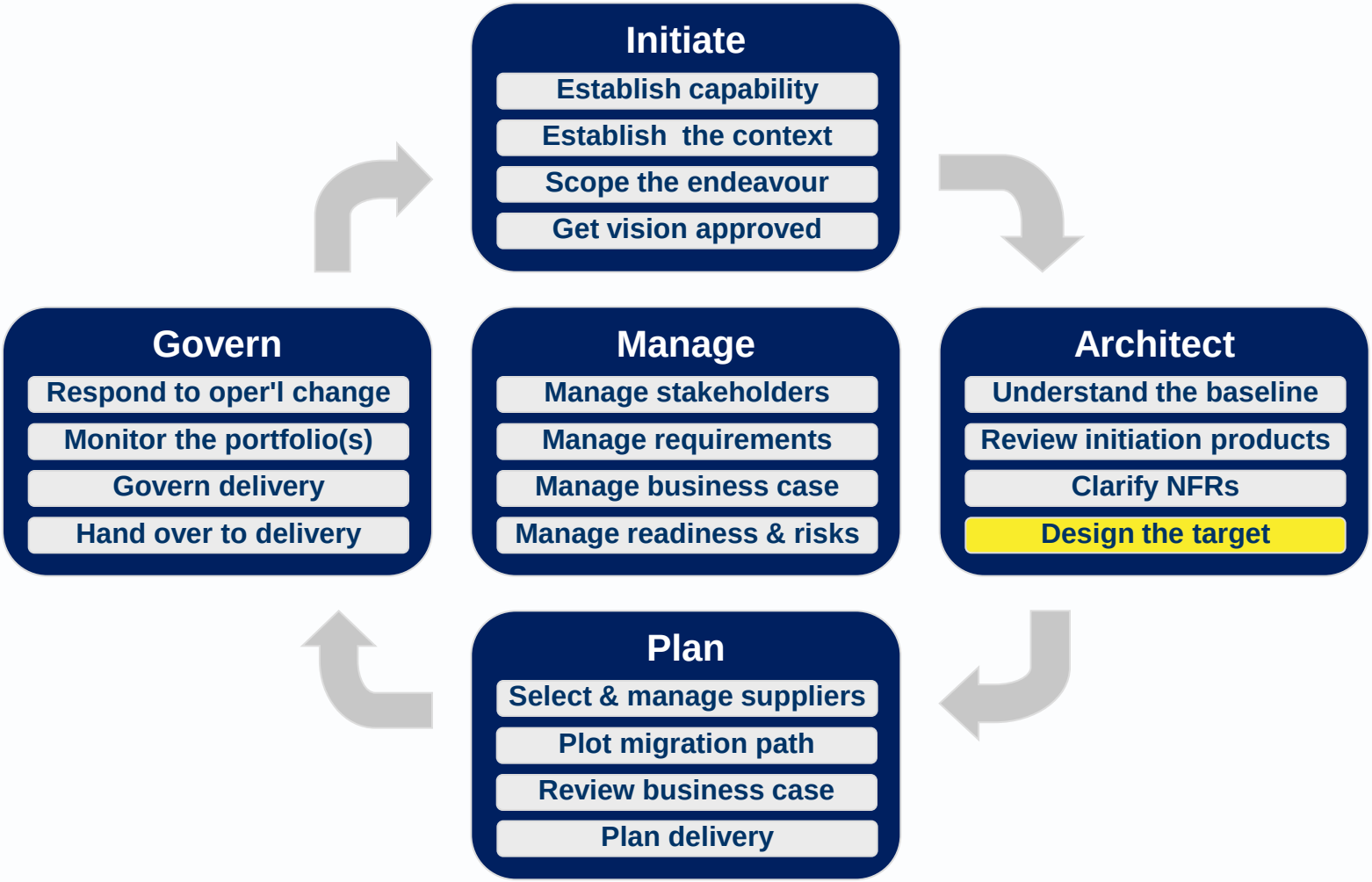
Data Architecture

Define data stores (relational)

The data/information architecture domain/layer

	Passive Structure	*Required Behaviour*	*Logical Structure*	*Physical Structure*
Business		Business Service Business Process	Function Role	Org Unit Actor
Data / Information	**Data Entity**	**Data Flow**	**Log Data Model**	**Data Store**
Applications		IS Service	Application Interface	Application
Infrastructure Technology		Platform Service	Platform Interface	Platform Applicat'n

Design the target (AM level 2)



Design target data architecture (AM level 3 and 4)

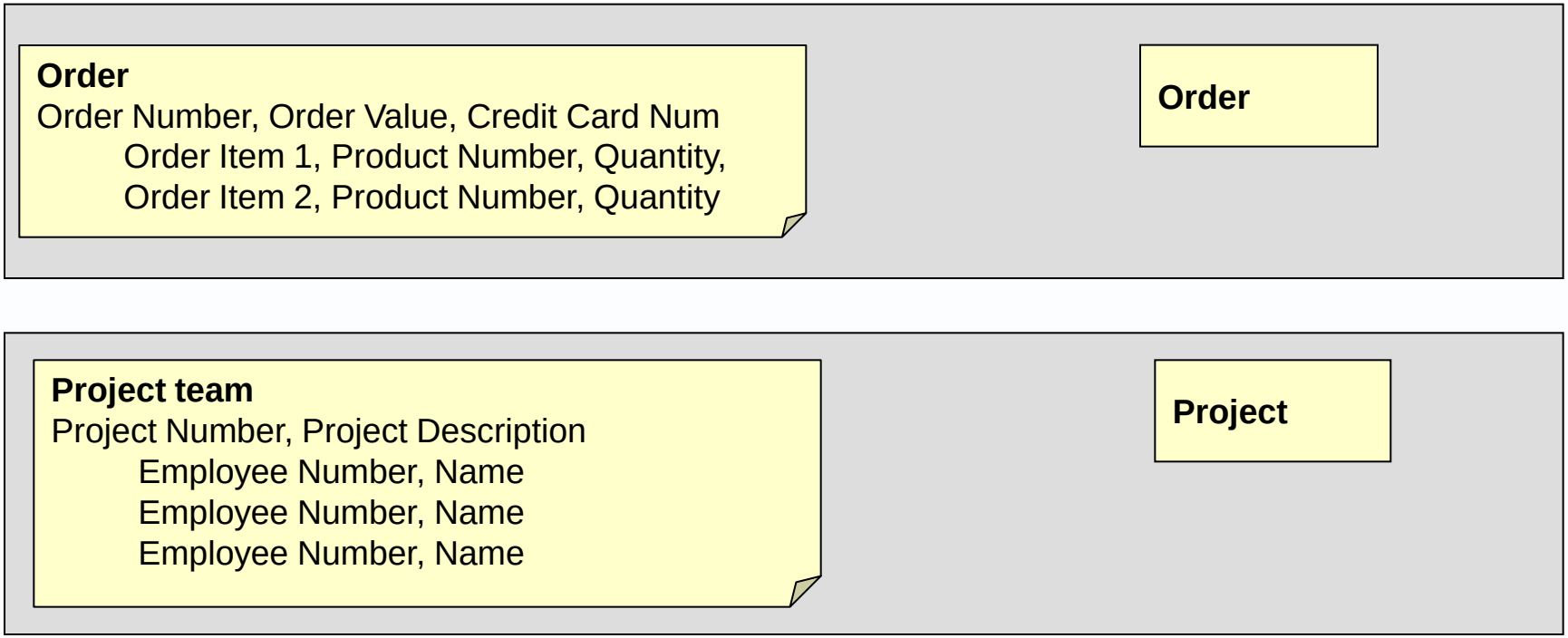
Avancier

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): relational**
    1. **Analyse I/O documents to find “entities” identified by primary keys.**
    2. **Define a logical data model - by “normalising” and relating the entities**
    3. **Code physical data model as database schema using the chosen DBMS.**
    4. **Refine the database design for flexibility and performance**
    5. **Design to ensure the database satisfies CIA and scalability requirements**
5. Address data quality issues

Analyse I/O documents to find “entities” identified by primary keys

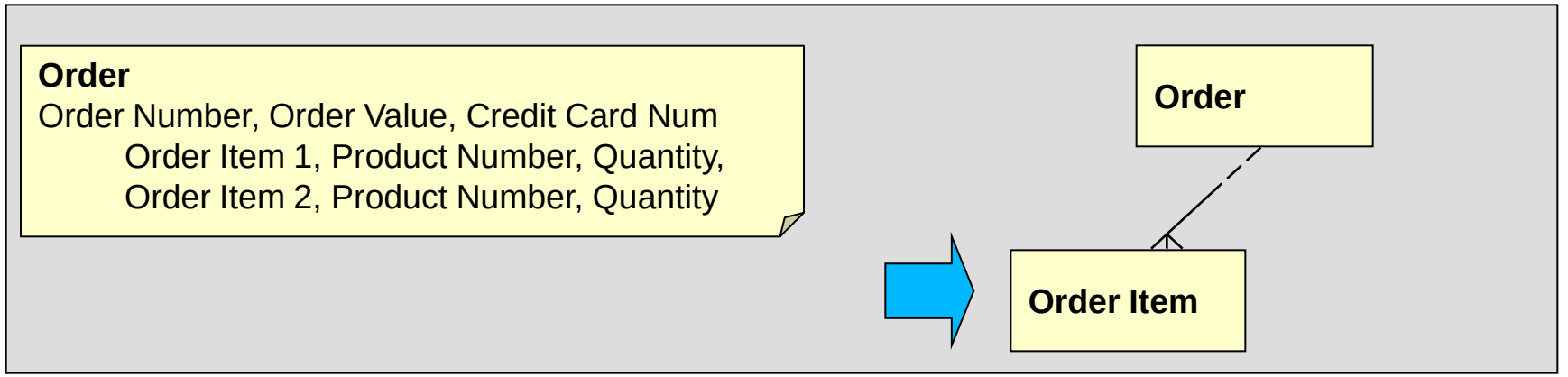
Avancier

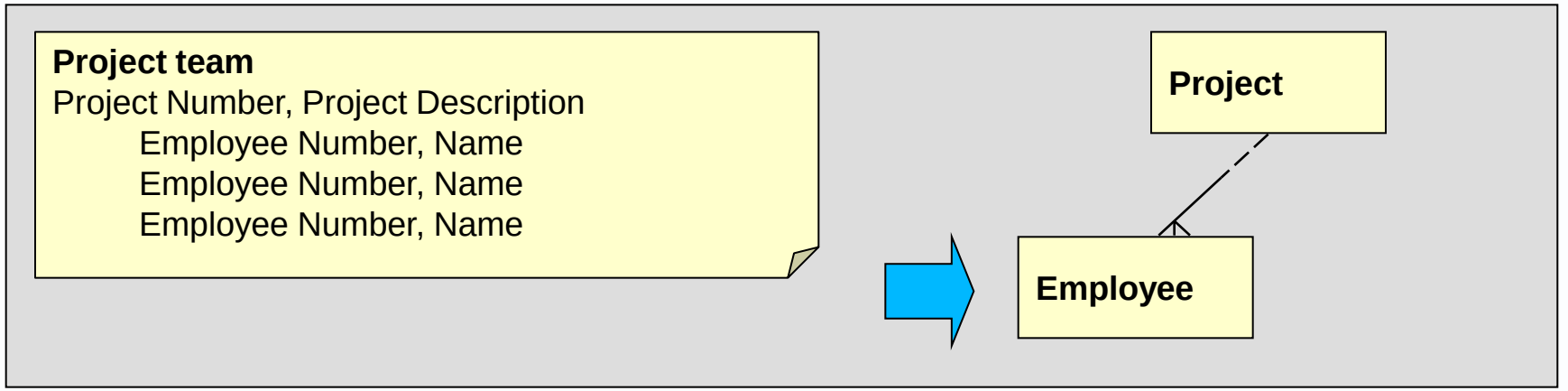
- Look for primary keys used in the business



First normal form

- Separate a repeating group into a detail entity



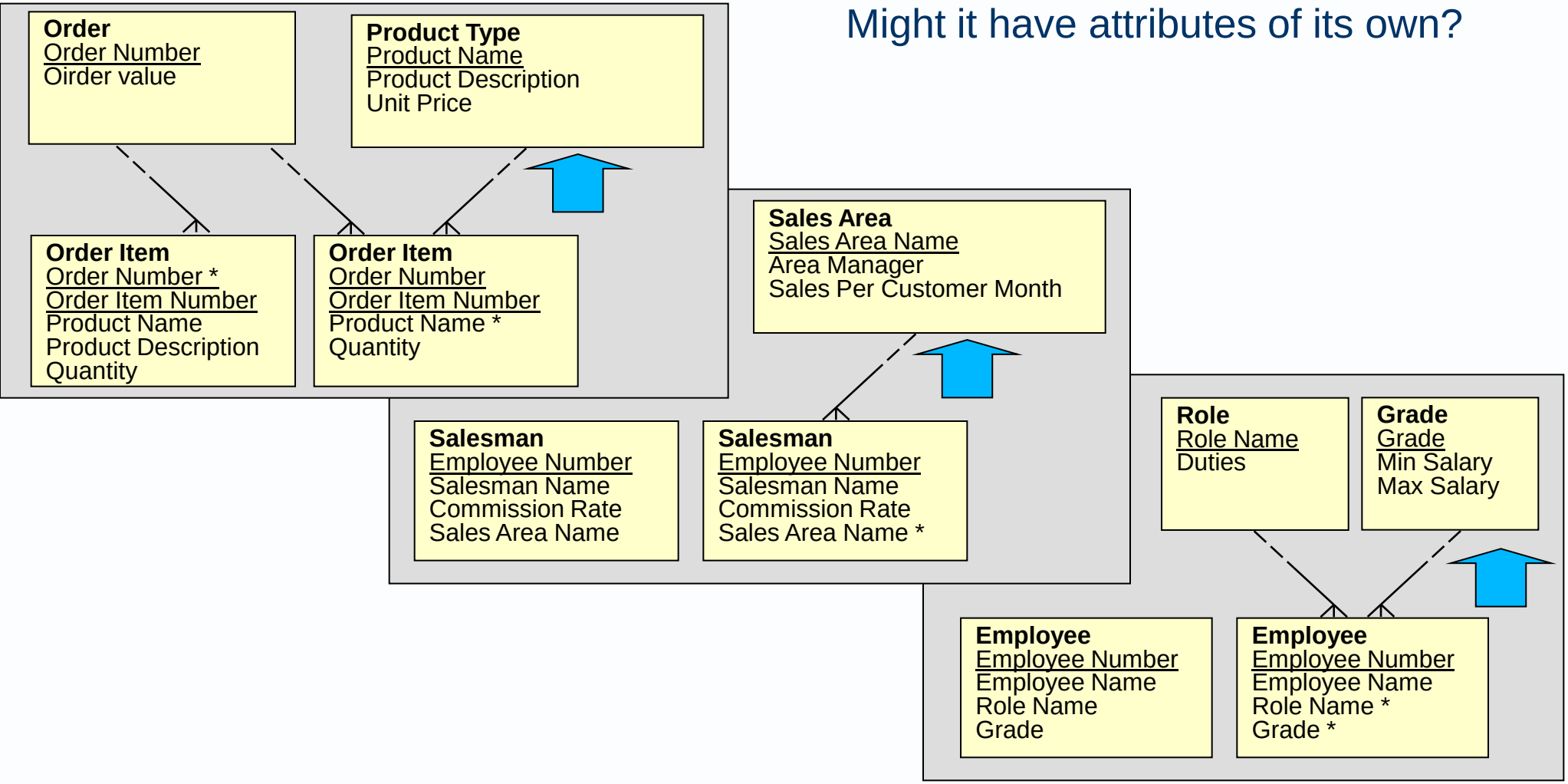


2nd and 3rd normal forms: raise attributes to become an entity

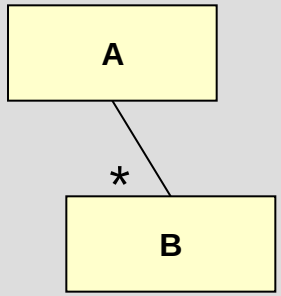
Avancier

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

Might it have attributes of its own?

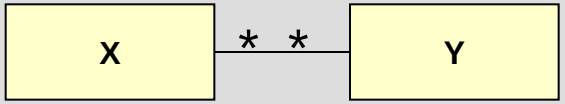


Consider an association from both ends



- 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

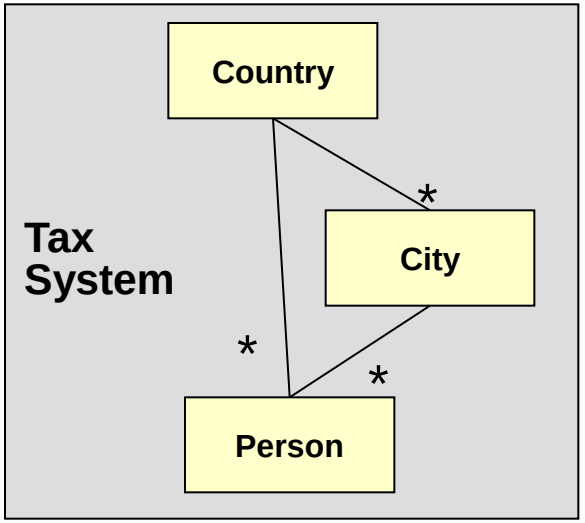


- Does entity Y relate to more than one X? Yes

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

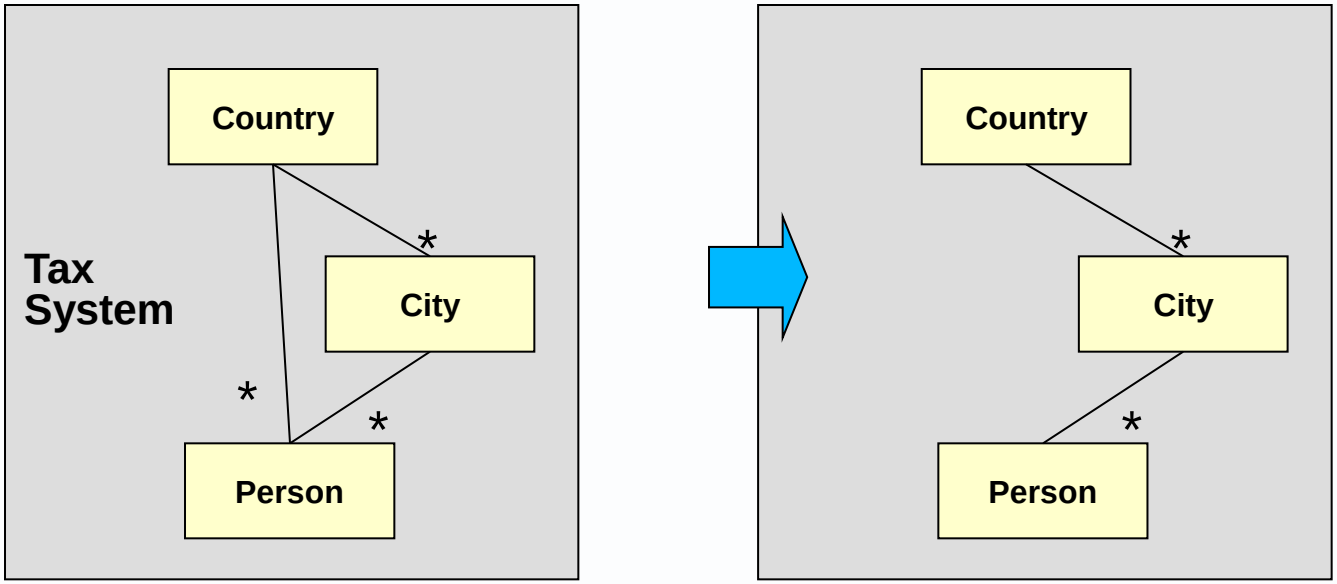


Is there redundant information we could remove?

- 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

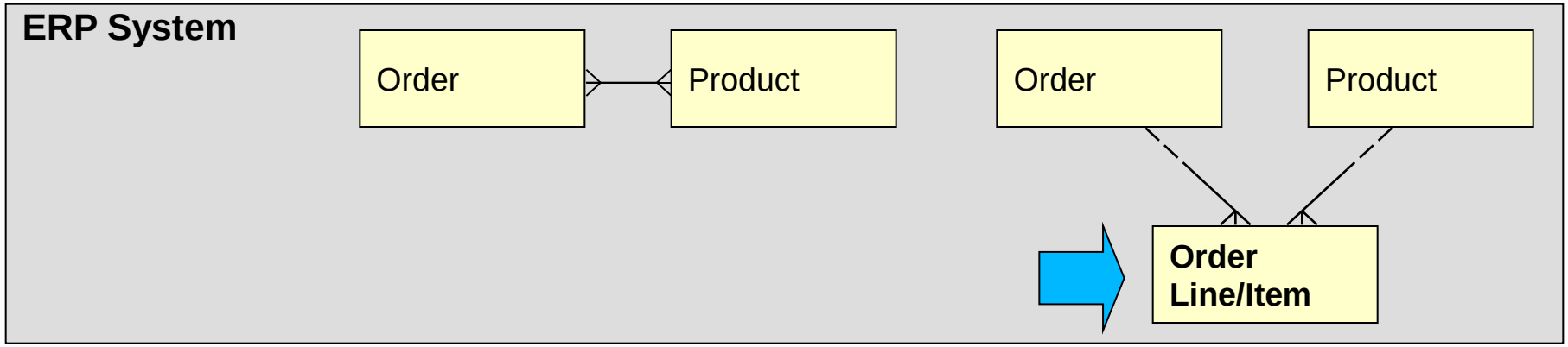
- Are the same Persons found down both long and short relationships? Yes (say)

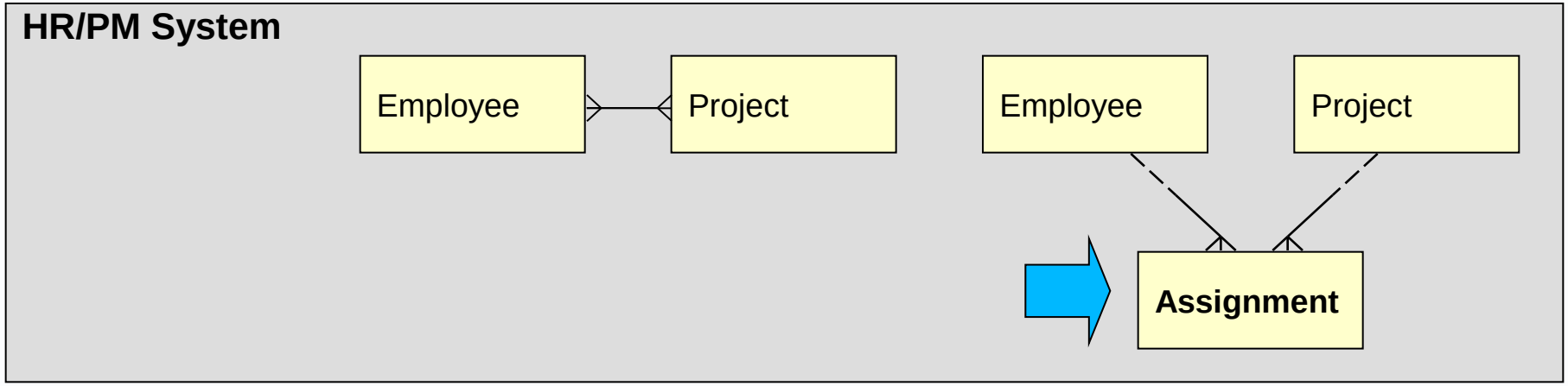


Look for link entities to resolve N-to-N associations

Avancier

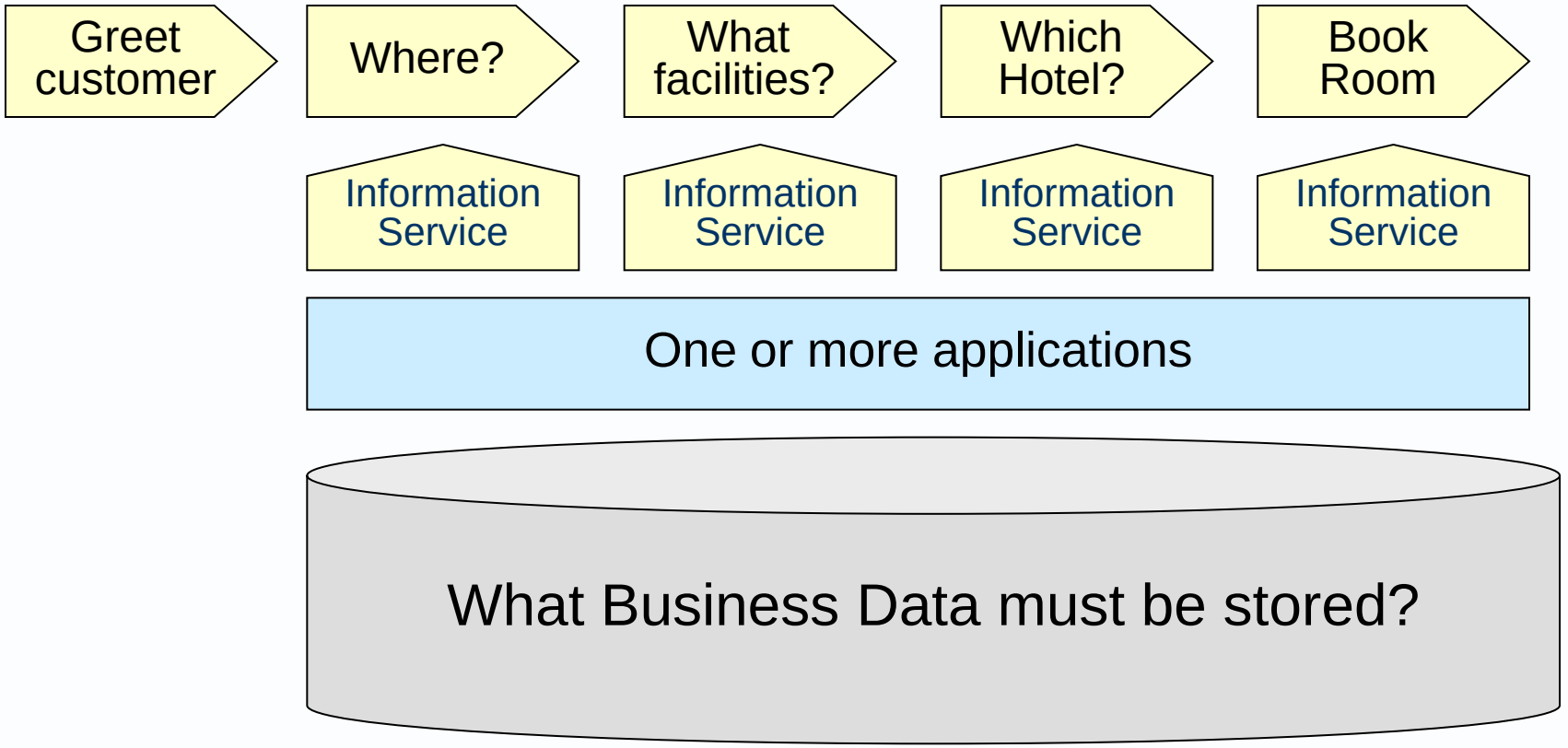
What is the name of the event or the thing that connects one of one entity to one of the other entity?





Hotel booking - value stream - scenario

- We have sales partners and hotel partners (hotel chains)
- The end-to-end business process of a sales agent



Ex 20m

EXERCISE: draw an entity relationship model (c 12 entities)

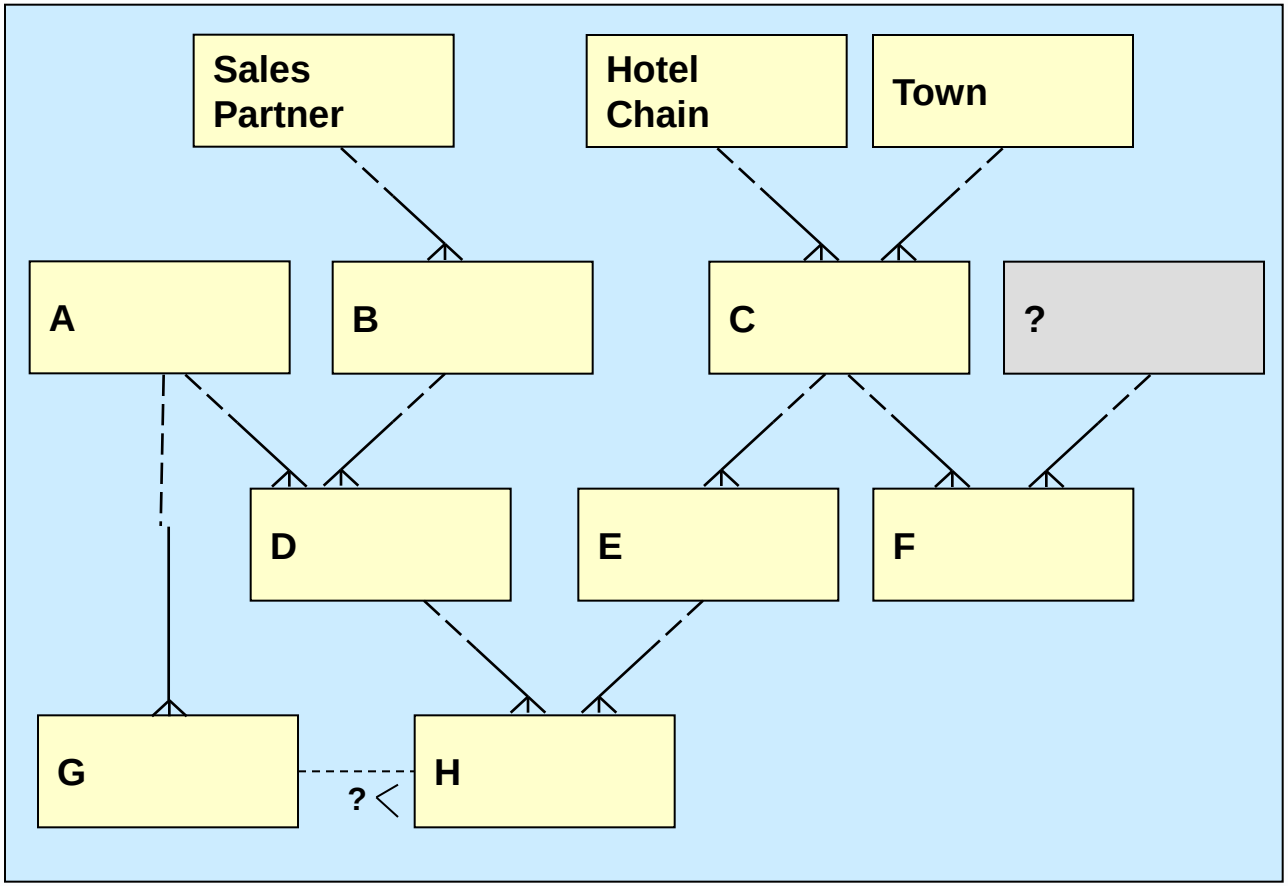
Avancier

Simplified from a real system with 80 entities

- We have sales partners (travel agents)
- We have hotel partners (hotel chains)
- We help sales partners to book rooms in the hotels of those hotel chains
- A sales agent works for a sales partner
- A customer first asks a sales agent to identify a hotel in a specific city or town, and with suitable facilities
- A customer may then ask the sales agent to make a booking in that a hotel
- A booking is composed of one or more room reservations
- A room reservation is the allocation of one room to one booking for a period of one or more days
- A customer may have many email addresses
- A reservation is often associated with a specific email address (where confirmation messages can be sent)

A starter for you

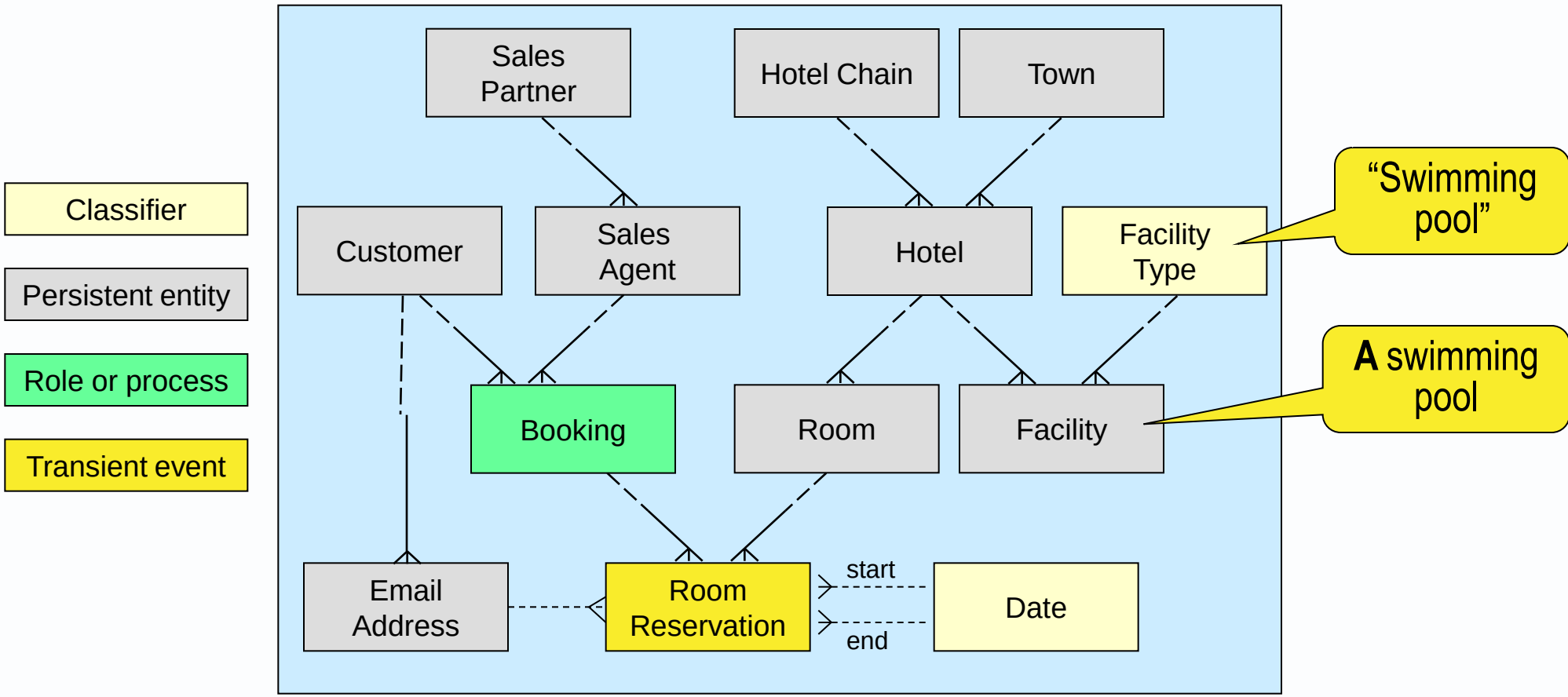
- What is A? B? etc.



A POSSIBLE answer

Avancier

- Attributes like Facility Type and Date are sometimes drawn as entities in their own right - especially if they are important entry points for business enquiries



See data modelling techniques for notes on…

- 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 redundancy in 1-to-1 associations
- Look for link entities to resolve N-to-N associations
- 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

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): relational**
    1. **Analyse I/O documents to find “entities” identified by primary keys.**
    2. **Define a logical data model - by “normalising” and relating the entities**
    3. **Code physical data model as database schema using the chosen DBMS.**
    4. **Refine the database design for flexibility and performance**
    5. **Design to ensure the database satisfies CIA and scalability requirements**
5. Address data quality issues

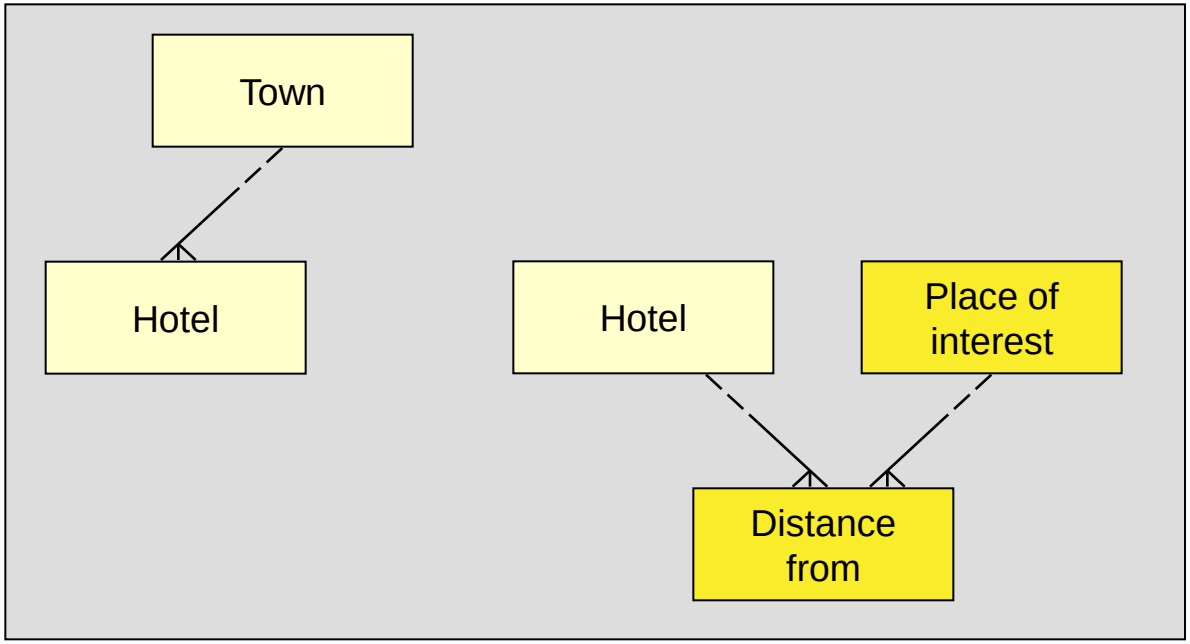
Validation 1: Is it flexible enough to meet user needs?

Avancier

Is some generalisation needed?

Do we search for hotels only within a town?

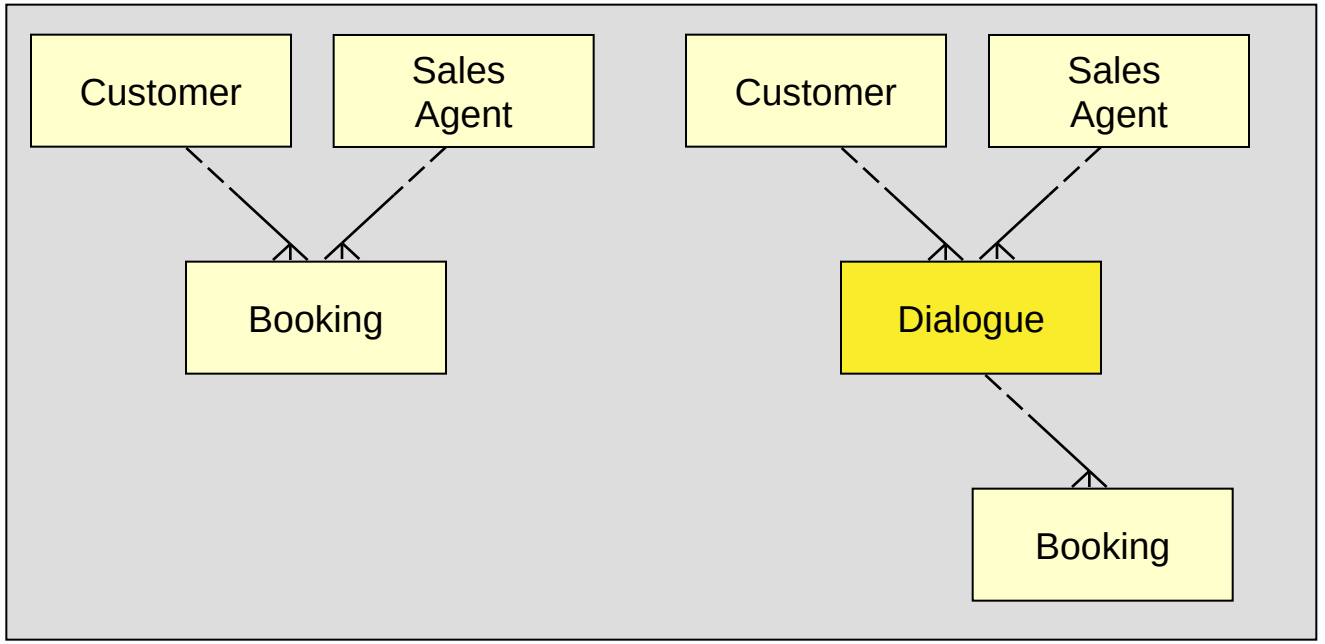
A more flexible search introduces an N-to-N association



Validation 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)?

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

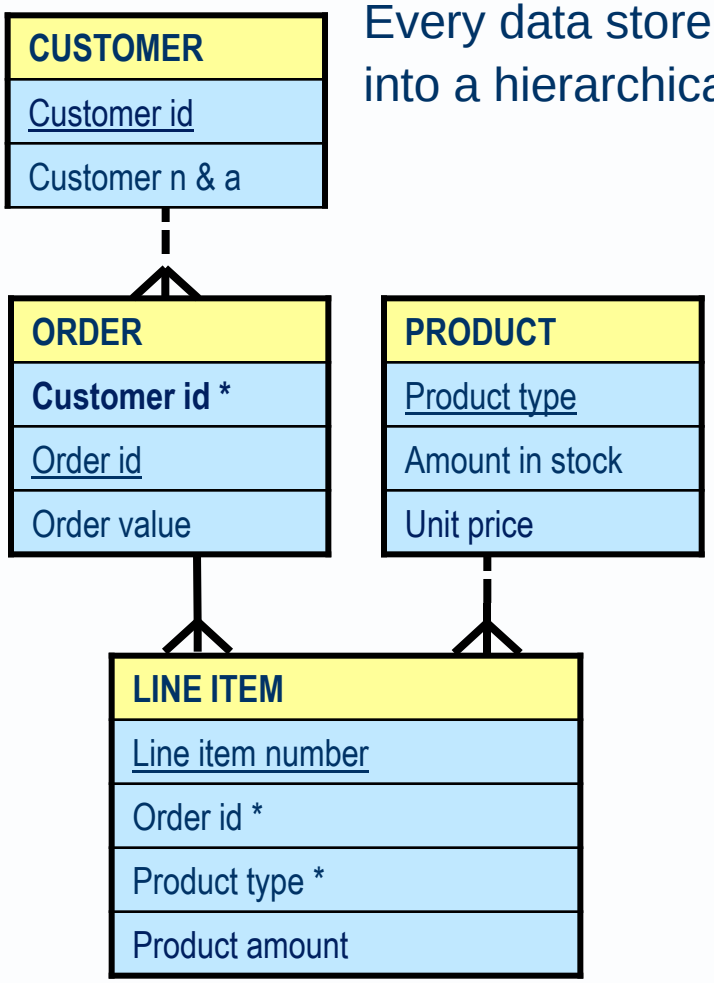


Argos record enquiries for out-of-stock items as well as orders

Refine the database design for flexibility and performance

- Identify processes to be supported, especially
  - batch input and output processes
  - predicted queries and required reports
  - processes that are frequent or have long access paths
- Ensure the data model contains data needed by those processes
- Facilitate process access paths
  - Do access path analysis
  - Add derivable data
  - Add derivable relationships
  - Otherwise denormalise the data structure

Consider how data will be serialised into a required data flow



Every data store can be serialised into a hierarchical / sequential data flow

Serialisation

Customer Order History
- **Customer id**
- **Customer name and address**
- Orders Placed
  - **Order id**
  - **Order value**
  - Products Ordered
    - **Product type**
    - **Product amount**
  - Products Ordered End
- Order Placed End

Customer Order History END

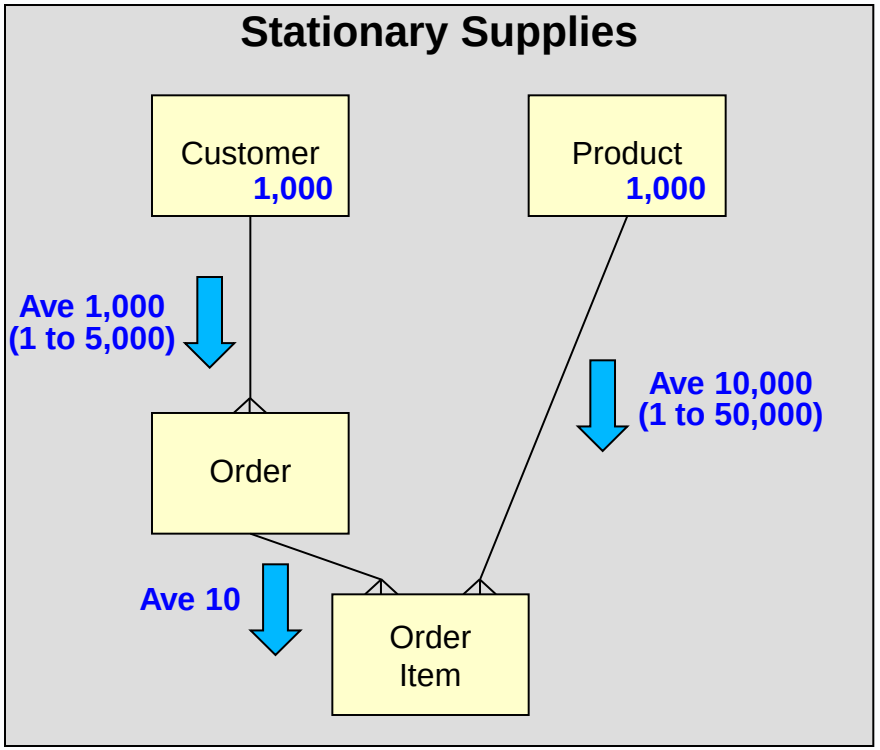
Serialisation

Product Demand
- **Product type**
- **Amount on hand**
- Products ordered
  - **Product amount**
  - **Order id**
- Products Ordered End

Product Demand End

Don’t forget the numbers: school stationary supplies

- Given a Logical Data Model, define
  - the volumes of kernel entities,
  - the population of each relationship,
  - expected growth rates.

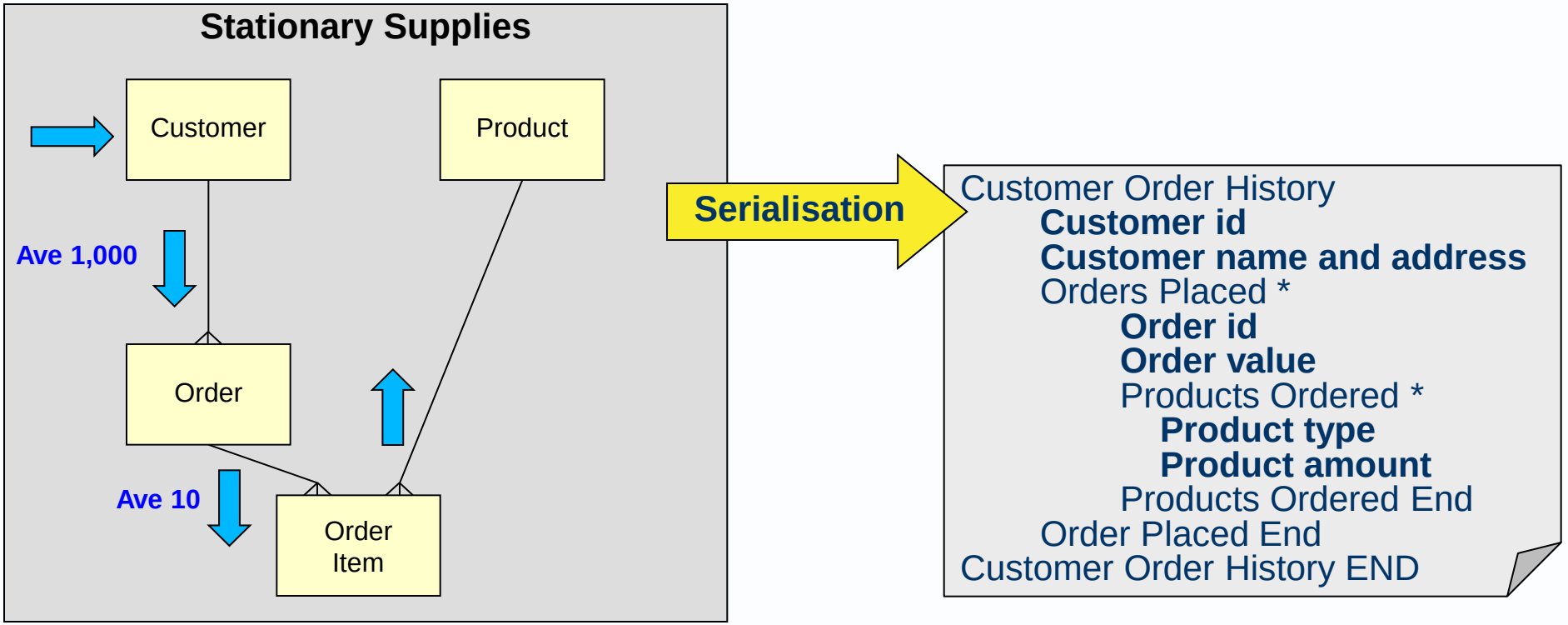


The typical customer (a school)
has placed 1,000 orders,
each with 10 items,
for our stationary products

These numbers influence
both system usage
and physical design

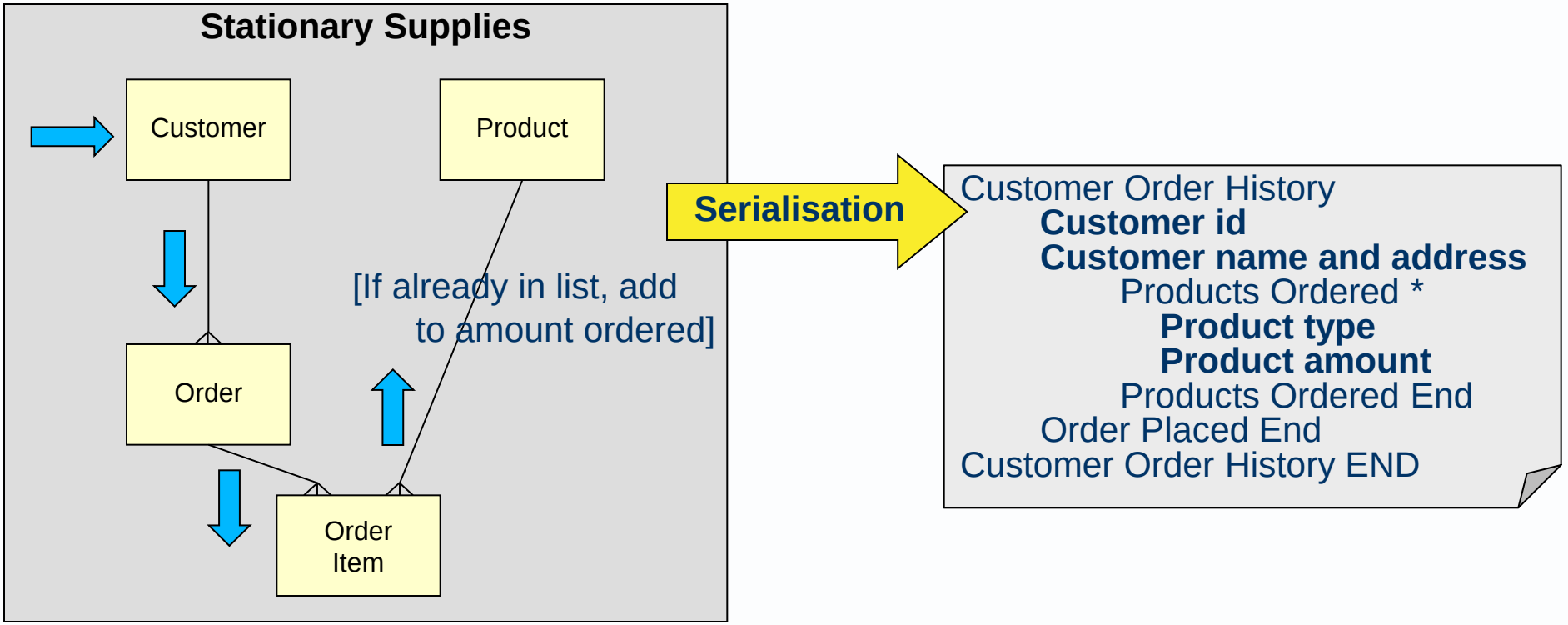
The first system release

- Before calling, a salesman requests a customer's order history
- The application prints out a list of 10,000 order items.
- "That's no good! I only want a list of products the customer has ordered!
  - (on average, 15 product types)



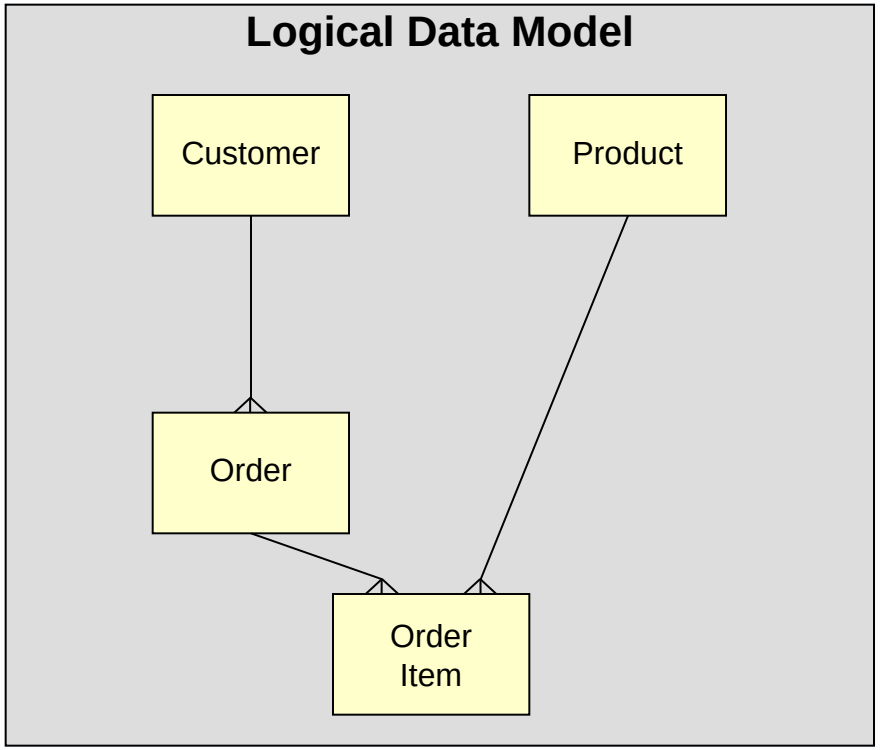
System change request

- Query: “List the products ordered by a customer, with total amounts”
- First, is it feasible? Yes.



Facilitate access paths

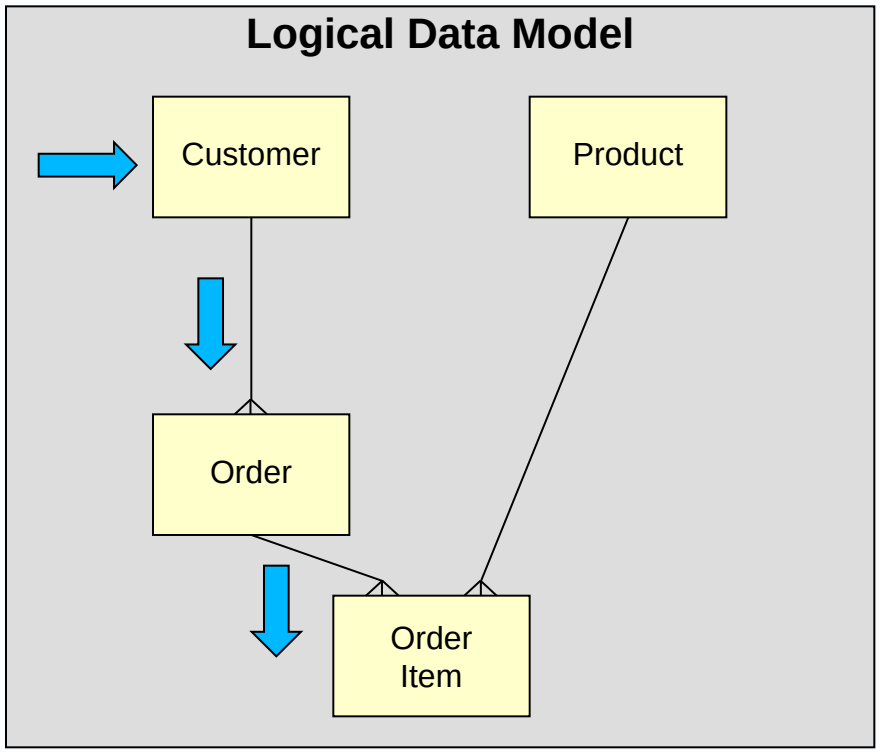
- Make sure critical processes can readily access the data they need
- Do access path analysis
- Add derivable data and/or relationships
- Otherwise denormalise the data structure

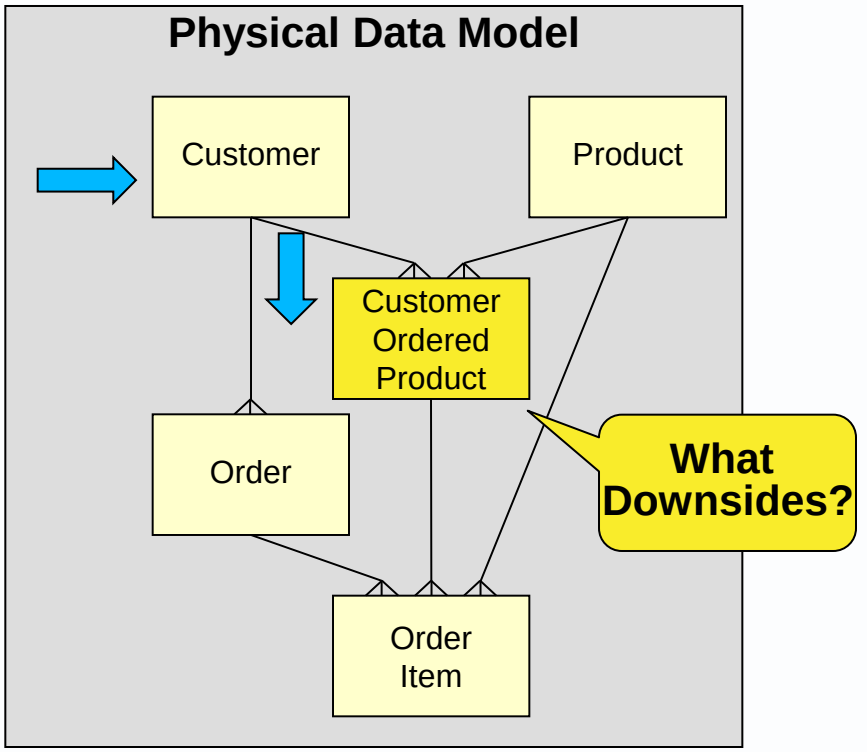


Do access path analysis – is it fast enough?

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

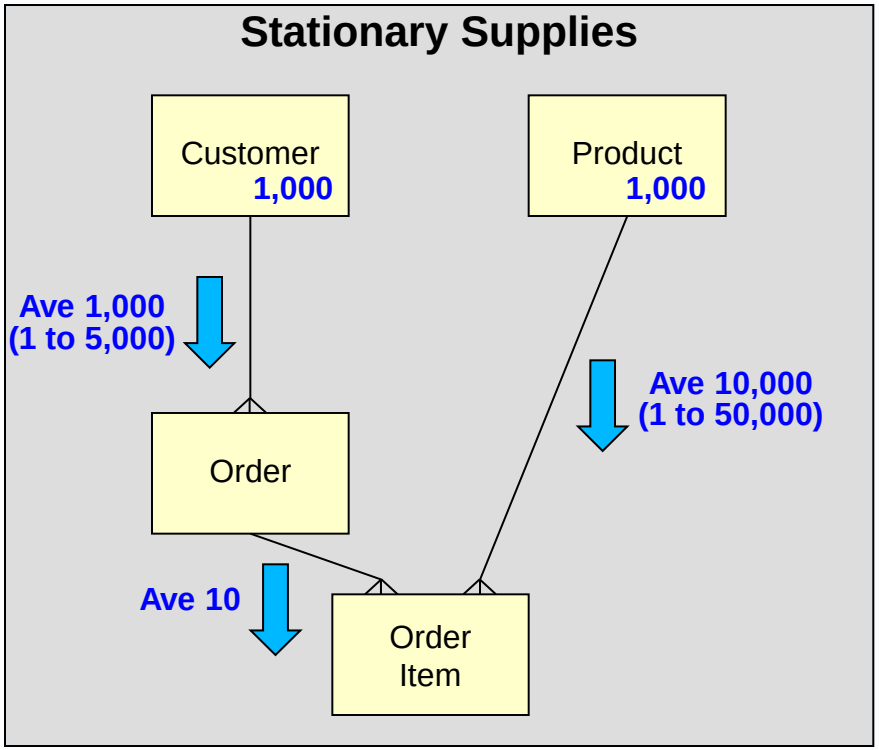
- You can remove redundant accesses by adding redundant data or redundant relationships





Don’t forget the numbers

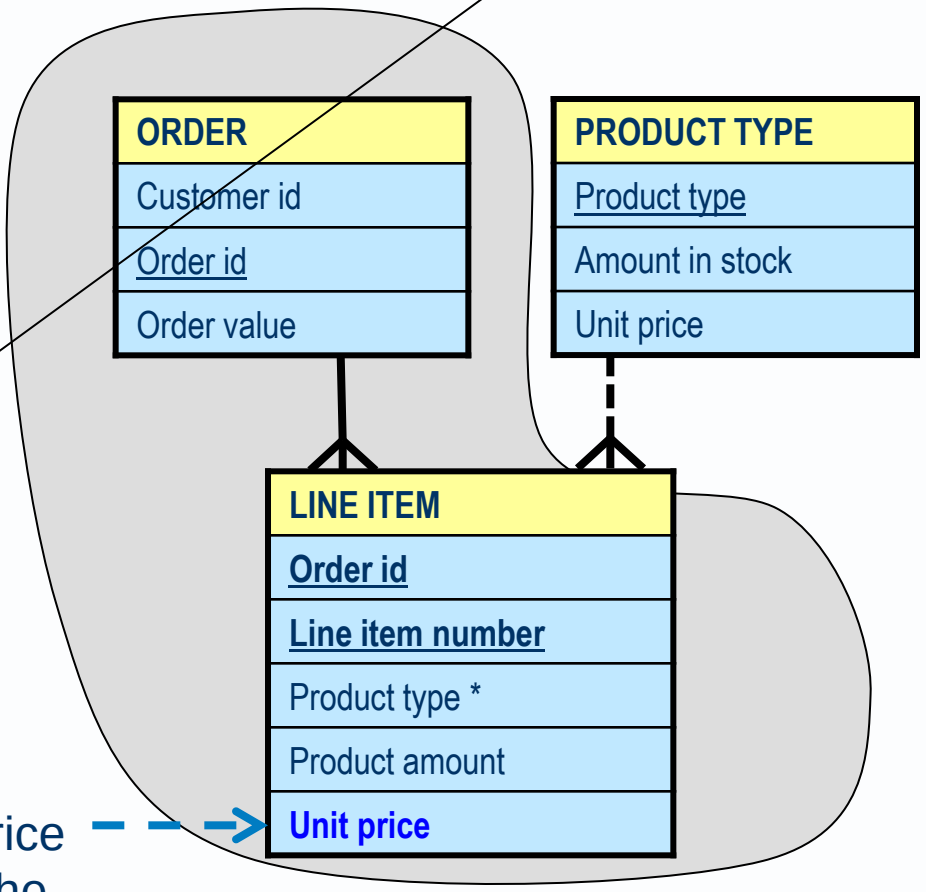
- Given a Logical Data Model, define
  - the volumes of kernel entities,
  - the population of each relationship,
  - expected growth rates.



The numbers influence both system usage and physical design

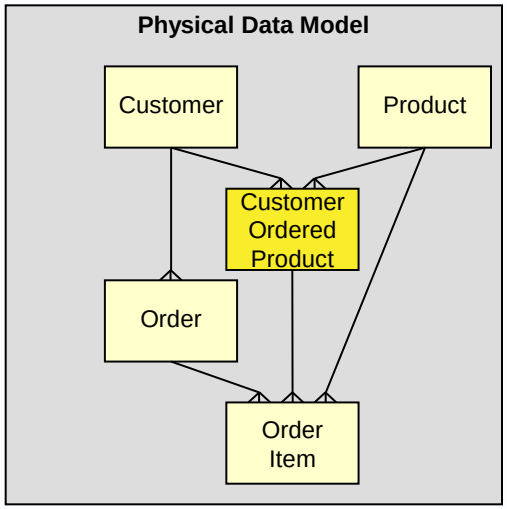
Clustering related data on the same block/page

- Old-fashioned optimisation technique: cluster detail entities with *one* of several parents
- Rule of thumb: “the least dependent occurrence rule”
  - An Order has few Order Lines
  - A Product has many
  - So cluster the Order Lines on the same block/page as the Order
- Copying the price stops the price change transaction affecting the aggregate, which is probably the business rule anyway

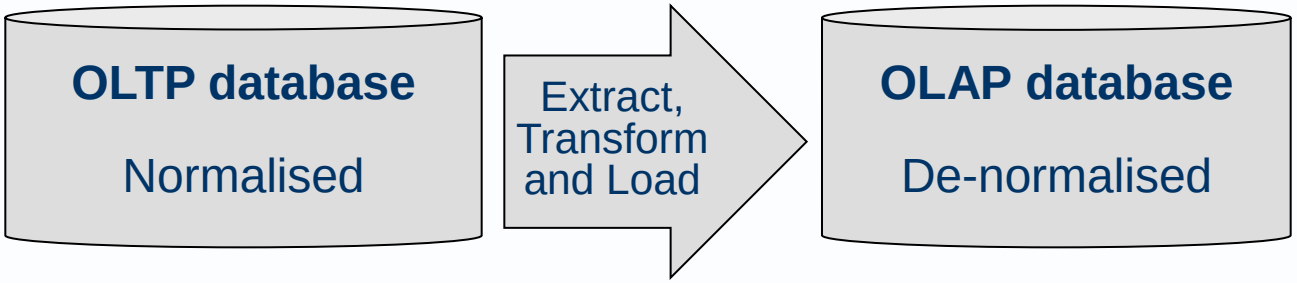


Separation of update and query data stores

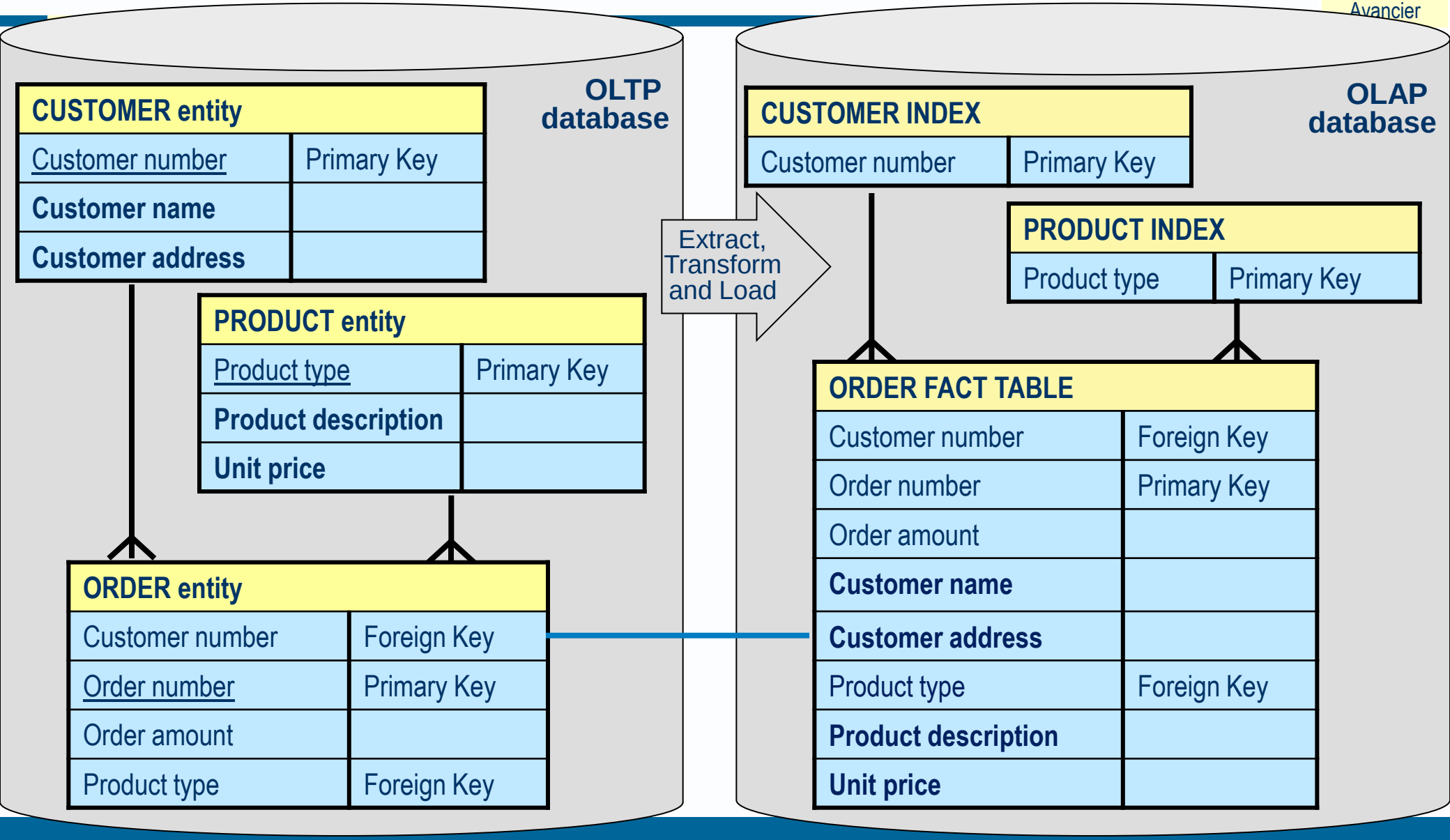
- You can
  - reduce redundant data accesses by
  - adding redundant data, relationships and indexes.



- Where that is not enough, you can
  - Separate update data store from query data store
  - And de-normalise the query data store



Denormalisation for faster enquiry/report processes (naive picture)



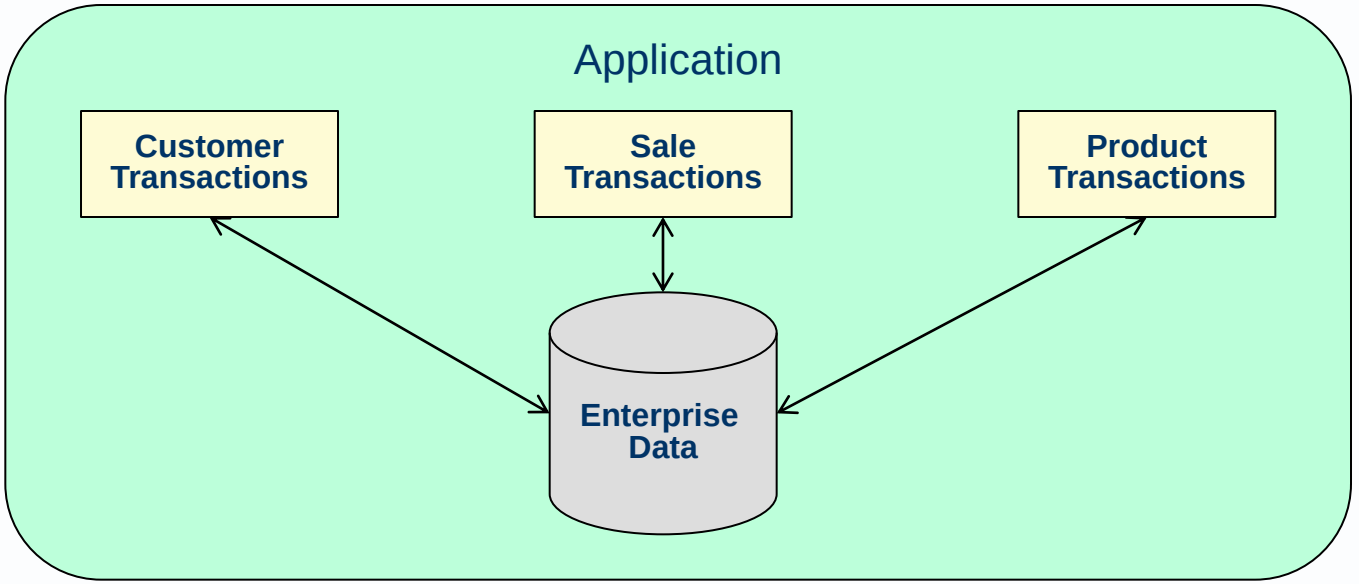
Design target data architecture (AM level 3 and 4)

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): relational**
    1. **Analyse I/O documents to find “entities” identified by primary keys.**
    2. **Define a logical data model - by “normalising” and relating the entities**
    3. **Code physical data model as database schema using the chosen DBMS.**
    4. **Refine the database design for flexibility and performance**
    5. **Design to ensure the database satisfies CIA and scalability requirements**
5. Address data quality issues

Design for integrity

- You may consolidate an enterprise's business data into a large database.
  - "It is not hard to speculate about, if not realize, very large, very complex systems implementations, extending in scope and complexity to encompass an entire enterprise." John Zachman, 1987

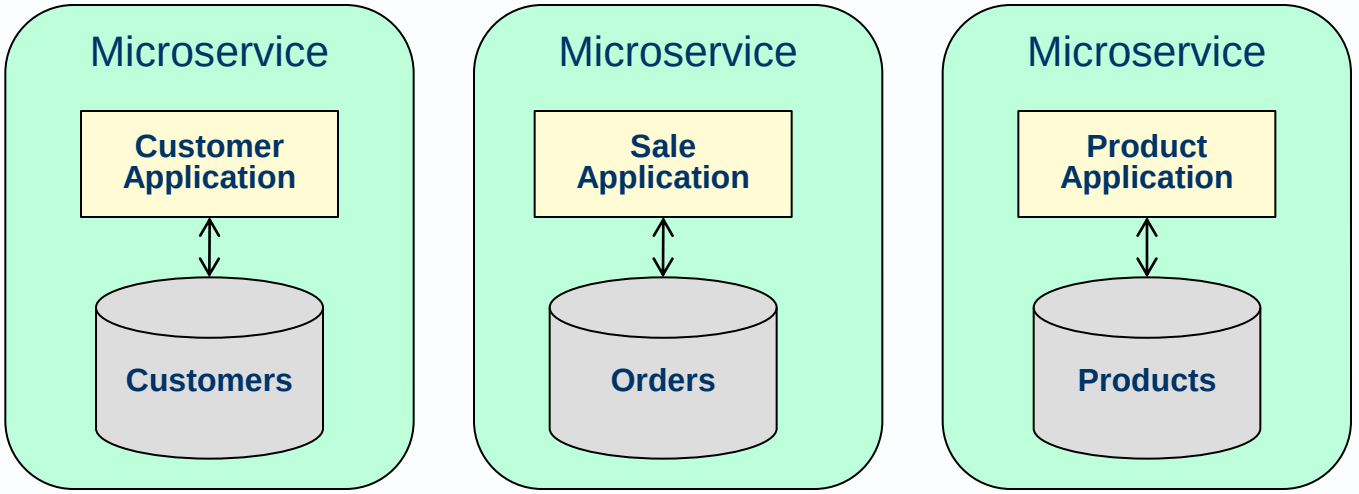


- And it appears SAP pursued this strategy for many years.

Design for scalability

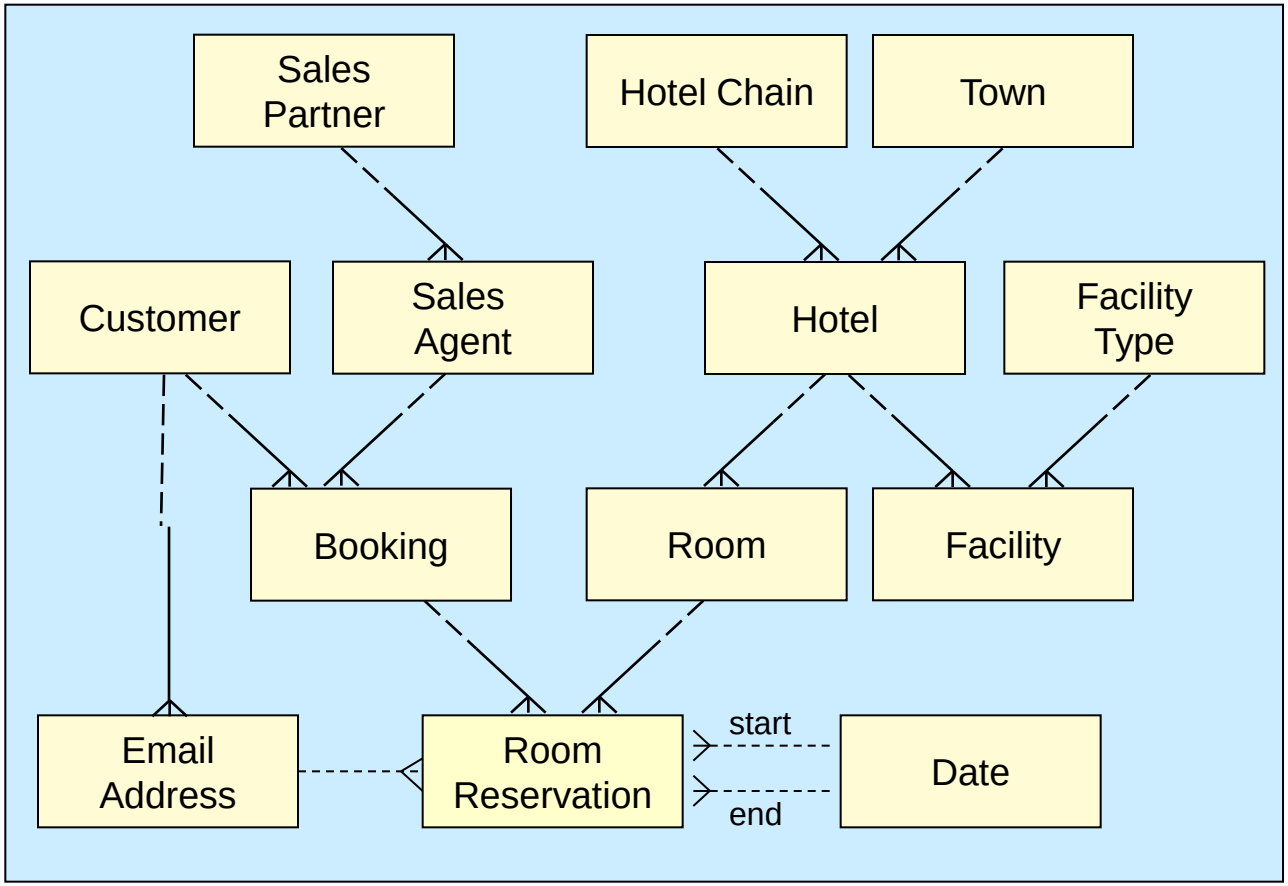
- “Distribute data management" to so-called “micro services” that each maintain part of the cohesive data structure

- They may be deployed separately, but are *still coupled*



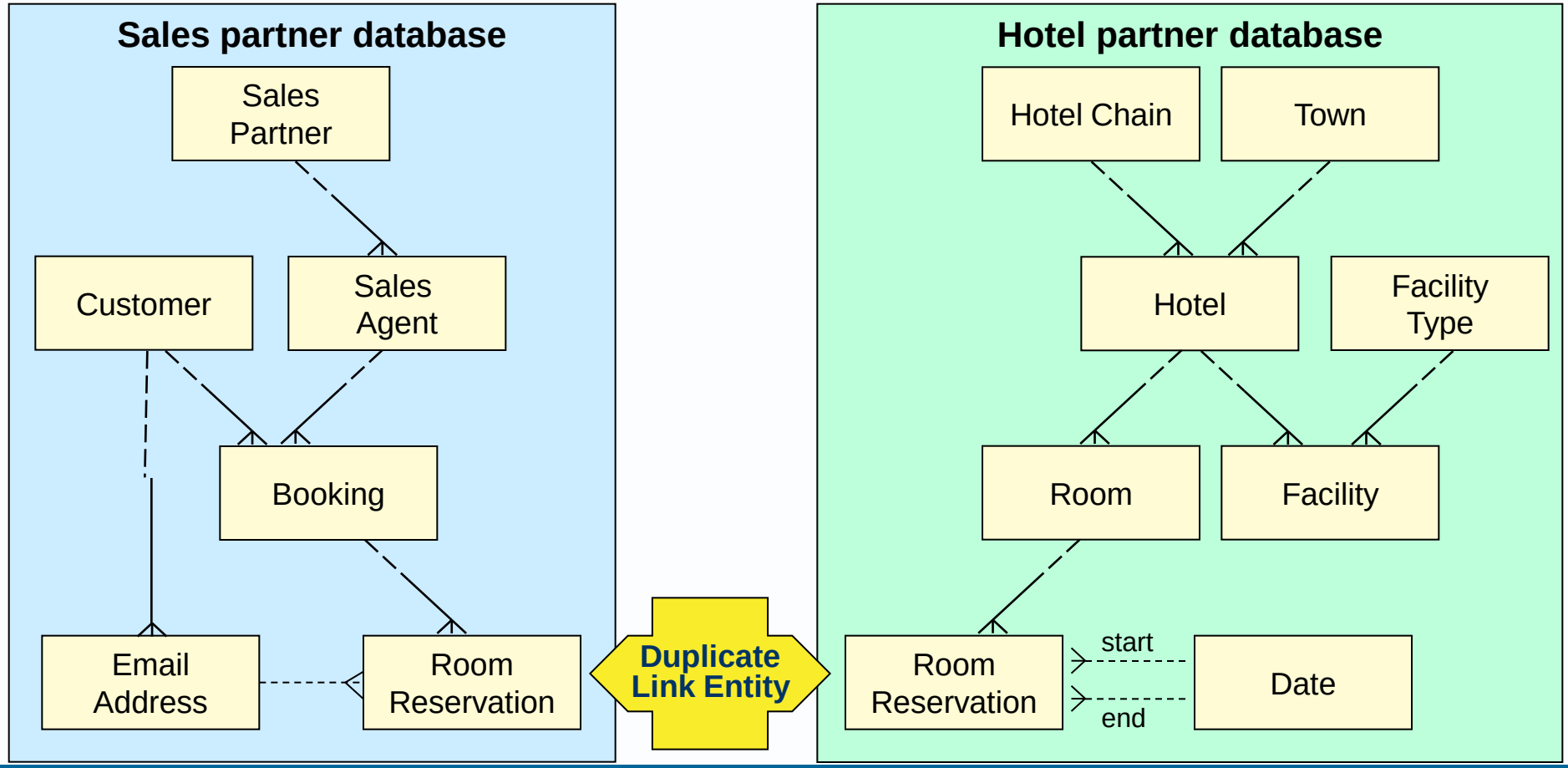
Partitioning a data store for availability and scalability

- How would you divide this logical data model between two data stores?



Horizontal partitioning into two data stores

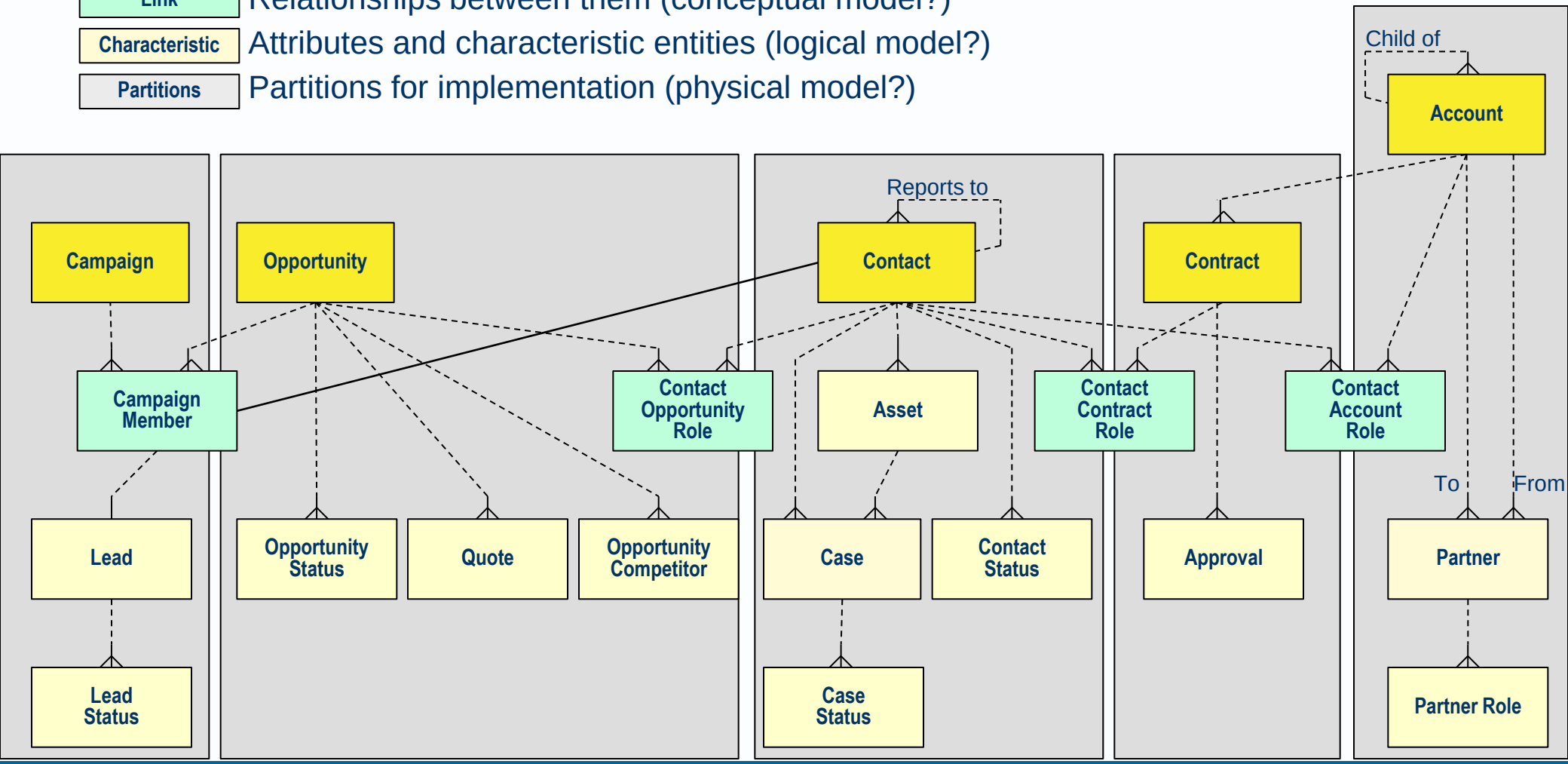
- Divide the network structure into hierarchies: has implications for both software architecture and business architecture… tbd



Salesforce.com domain model redrawn (cf. ZF column 1?)

Avancier

- **Kernel** — Kernel entities
- **Link** — Relationships between them (conceptual model?)
- **Characteristic** — Attributes and characteristic entities (logical model?)
- **Partitions** — Partitions for implementation (physical model?)



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): relational**
    1. **Analyse I/O documents to find “entities” identified by primary keys.**
    2. **Define a logical data model - by “normalising” and relating the entities**
    3. **Code physical data model as database schema using the chosen DBMS.**
    4. **Refine the database design for flexibility and performance**
    5. **Design to ensure the database satisfies CIA and scalability requirements**
5. Address data quality issues