Avancier Methods (AM)

Techniques

Logical Domain Models

It is illegal to copy, share or show this document

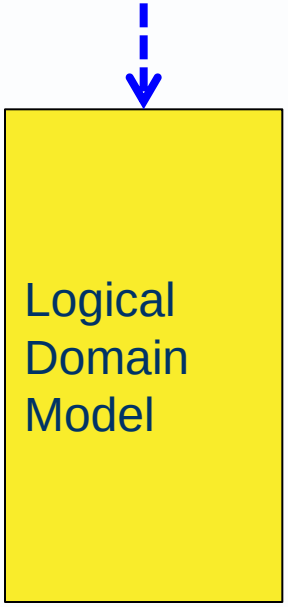
(or other document published at http://avancier.co.uk)

without the written permission of the copyright holder

Copyright Avancier Ltd 2015

A proposal for software architects

- This is a well formed - truly technology independent – logical domain model



Logical Domain Model

Optimise for OOPL

- NFR and technology constraints

Optimise for DBMS

Events and Queries

OO Domain Model

Physical Structural Model

Discussed elsewhere

A layered software architecture

User Interface Layer

Service Layer

Business Logic Layer

Data Abstraction Layer

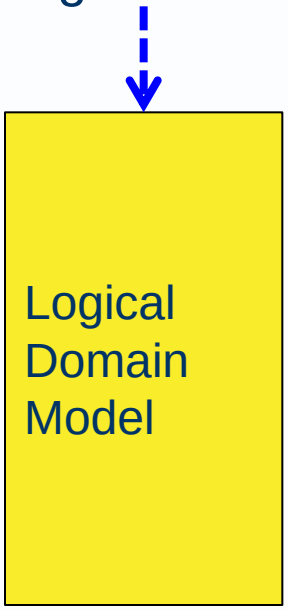
Database Structure

The closer these to the Logical Domain Model
the simpler and cheaper the data abstraction layer, the more likely the database will support new queries

Copyright Avancier Ltd 2015

The question is

- What exactly is a well formed - truly technology independent – logical domain model?



Copyright Avancier Ltd 2015

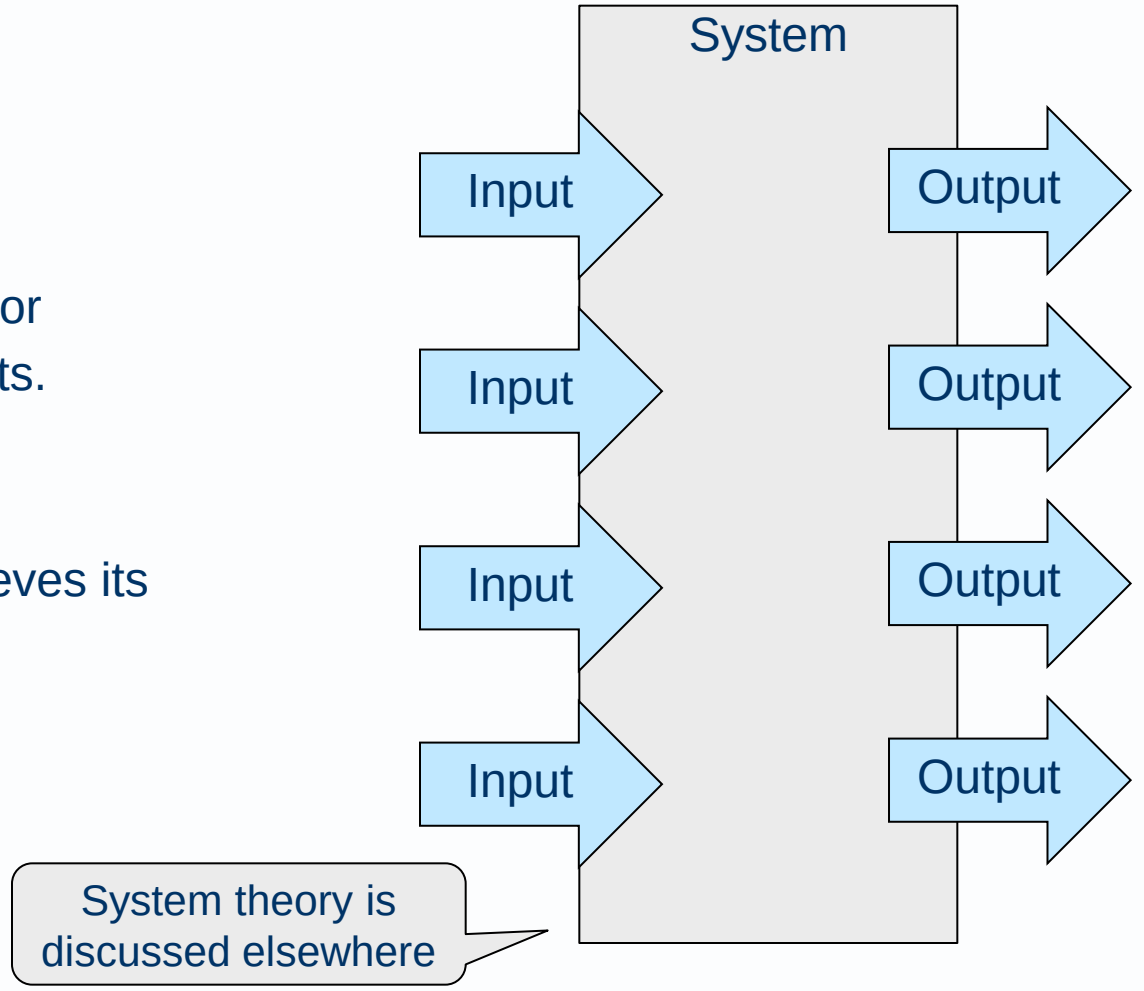
Requirements

- **Requirements**
- Analysis of system input/output
- Data types
- Modelling system state and behaviour
- Aggregate entities
- Analysing association cardinality
- Analysing class hierarchies

Copyright Avancier Ltd 2015

Scoping a software system

- An activity system is composed of
  - components that play
  - roles in
  - processes to achieve
  - desired effects by
  - maintaining system state and/or
  - transforming inputs into outputs.
- A software system
  - is an activity system that achieves its
  - desired effects by supplying
  - outputs to external entities.



Copyright Avancier Ltd 2015

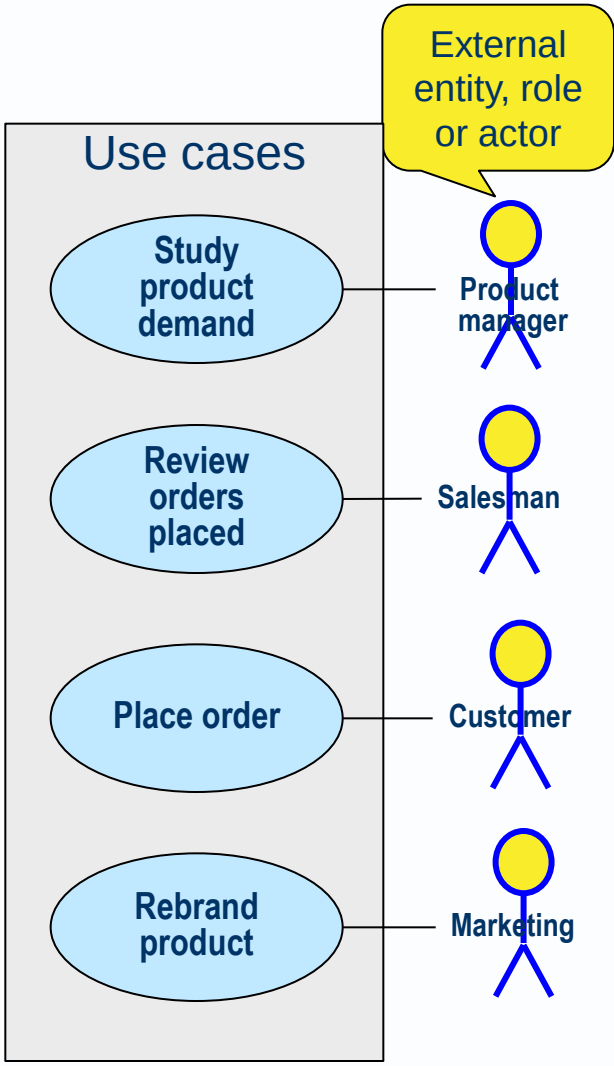
Required system behaviour

- A **use case diagram** shows the processes of a system that external entities are involved in

- A use case is typically a process to
  - Enter an input data structure (via events or commands)
  - Retrieve an output data structure (via queries)

- Each system input/output behaviour can be detailed in the form of a **use case definition**

Discussed elsewhere



Copyright Avancier Ltd 2015

Required system outputs

- Key queries
- A Salesman wants to know what Orders a Customer has placed (new or historical).

Review orders placed

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

- The Product Managers want to know the demand (new or historical) for a Product type.

Study product demand

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

Copyright Avancier Ltd 2015

Events and queries important to the business domain

- Key queries (earlier)
  - Customer order history
  - Product demand
- Key events
  - Customer Registration and Withdrawal
  - Product Introduction and Withdrawal
  - Order Placement and Payment
- Core business transaction
  - Customers submit Orders requesting specified amounts of Products.
  - The preconditions for an Order include that there is enough Product in stock

Copyright Avancier Ltd 2015

From external to internal

- So let us assume
- Some use cases have been defined
  - To enter input data structures (via events or commands)
  - To retrieve output data structures (via queries)
- And some I/O data structures have been defined
- **The focus of this slide show is on how the internal behaviour of the system can be organised around a structural model**

Copyright Avancier Ltd 2015

Analysis of system input/output

- Requirements
- **Analysis of system input/output**
- Data types
- Modelling system state and behaviour
- Aggregate entities
- Analysing association cardinality
- Analysing class hierarchies

Copyright Avancier Ltd 2015

What is the aim?

Avancier

- A *logical* model of an activity system
  - defines business terms, concepts and rules
  - defines events and the entities they affect
  - technology independent - no reference to any IT technology

- An *economical* model
  - contains the data needed to produce required outputs
  - defines entities, attributes and rules important to this business
  - supports known query and event processes

- The most economical model is naturally efficient in some ways
  - But the choice of programming language and data store technology, and design for non-functional requirements is not a driver here

Copyright Avancier Ltd 2015

Analysis of shopping basket to find entities

Avancier

Shopping Basket
- **Customer id**
- **Order id**
- **Order value**
- Products Ordered
  - **Product type**
  - **Product amount**
- Products Ordered End

Shopping Basket

ORDER
Customer id
Order id
Order value

PRODUCT
Product type
Product Amount

- Look for primary keys business people have invented

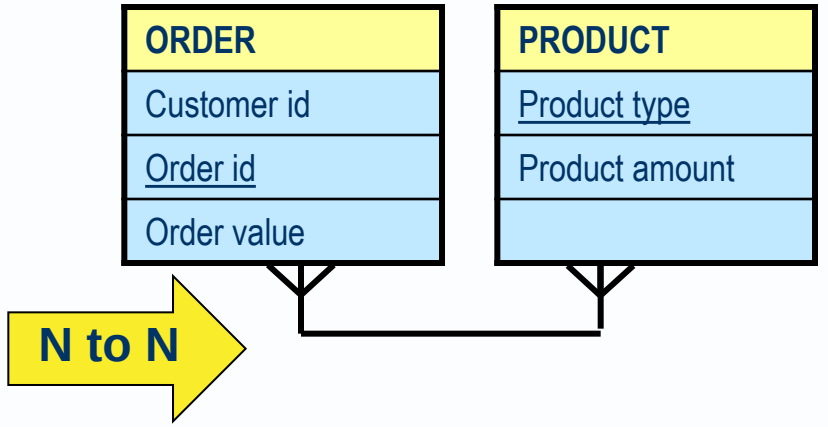
Copyright Avancier Ltd 2015

Analysis of a many-to-many association

Avancier

Shopping Basket
- **Customer id**
- **Order id**
- **Order value**
- Products Ordered
  - **Product type**
  - **Product amount**
- Products Ordered End

Shopping Basket



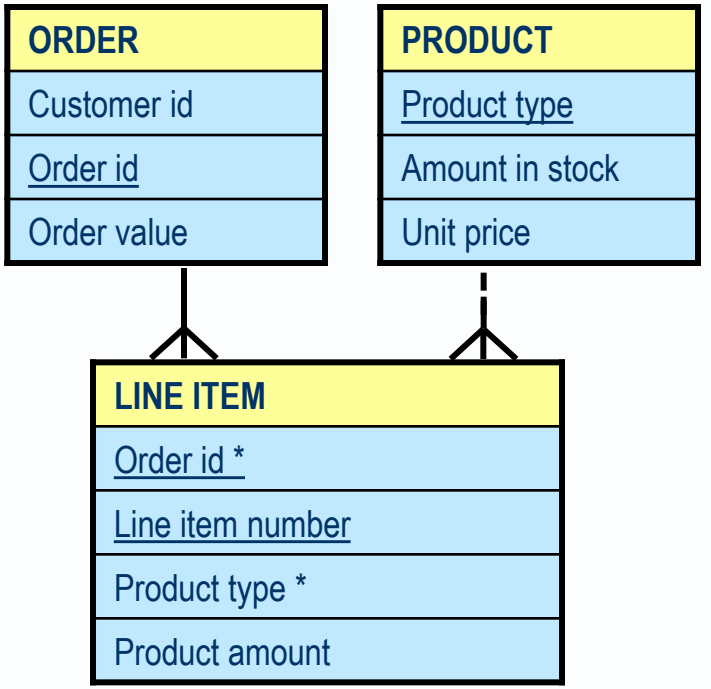
- The modeller identifies a many-to-many association
  - One Order can be for many Products
  - One Product will appear on many Orders

- What is the name of the event or thing that connects one to the other?
- Say, a “Line Item” that identifies the Product amount requested

Copyright Avancier Ltd 2015

More business rules can be added

- A Line Item is a discrete thing whose continuity of existence must be remembered.
- The structural model should define whether
  - an Order can exist without any Line Items.
  - a Line Item can be switched from today's Order to a future "Back Order".
- NOTE: These business rules are nothing to do with computers, except that computers demand we define rules more rigorously than a human system



* If you think showing foreign keys in a domain model implies a relational database, than stop thinking that! For reasons explained in the following slides.

Copyright Avancier Ltd 2015

Aside FYI

- The curious nature of the “Order” entity type is discussed in this short paper called “Order processing in 2500 BC”

- http://grahamberrisford.com/AM%202%20Methods%20support/01Introduction/Order%20processing%20in%202500%20BC.htm

Copyright Avancier Ltd 2015

Entity instances must be identifiable

- An entity is
  - a discrete thing that a business wants to remember
  - distinguishable from other instances of the same type.
- External actors must be able to
  - recognise an entity's continuity of identity, so that they can
  - apply events and queries to the right entity instance
  - monitor and/or direct the behaviour of each entity instance.
- To these ends, each entity instance must be uniquely identifiable in some way.

Copyright Avancier Ltd 2015

Primary keys

- When you find an identifier (name or number) used by humans, you have found an entity type whose instances are important to humans. *This has nothing to do with computers.*

- You can uniquely identify a thing in the world by
  - pointing at it, or
  - naming it (uniquely within a name space).

- As businesses evolved to handle many instances of a type, they introduced identifying codes and numbers. So primary keys (and foreign keys) existed in business systems long before computers.

- But IT people invent further identifiers, and may use a Globally Unique Identifier (GUID), a 128-bit integer number, to identify resources.

Copyright Avancier Ltd 2015

Creating primary keys

- When a new business system is created, newly defined entity types need unique identifiers

CUSTOMER
Customer id
Customer n & a

- "Kernel" entities
  - are given their own primary keys (GUID?)

LINE ITEM
Line item GUID
Order id *
Product type *
Product amount

- "Link" entities
  - may be given their own primary key
  - and/or a compound identifier.

LINE ITEM
Order id *
Product type *
Product amount

- "Detail" entities of an aggregate
  - may be given their own primary key
  - and/or hierarchical (composite) identifier.

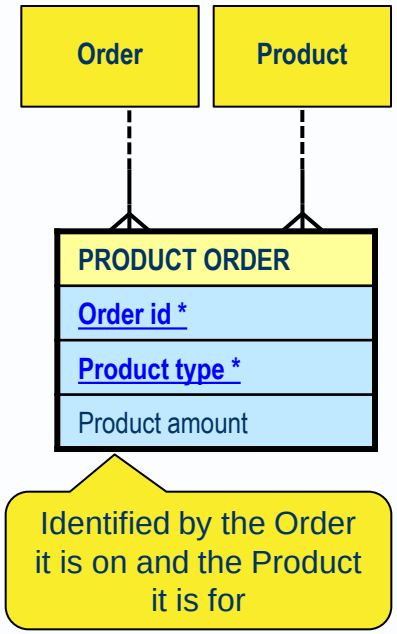
LINE ITEM
Order id *
Line item number
Product type *
Product amount

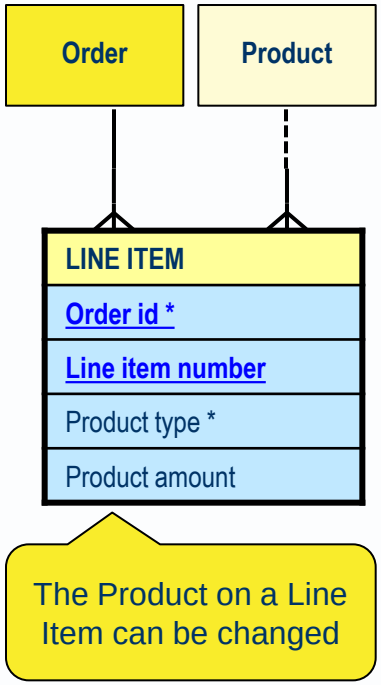
Copyright Avancier Ltd 2015

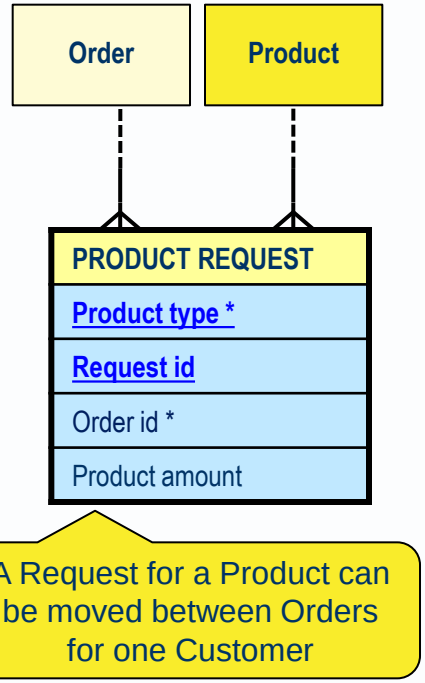
How foreign keys specify rules for coupling between entities

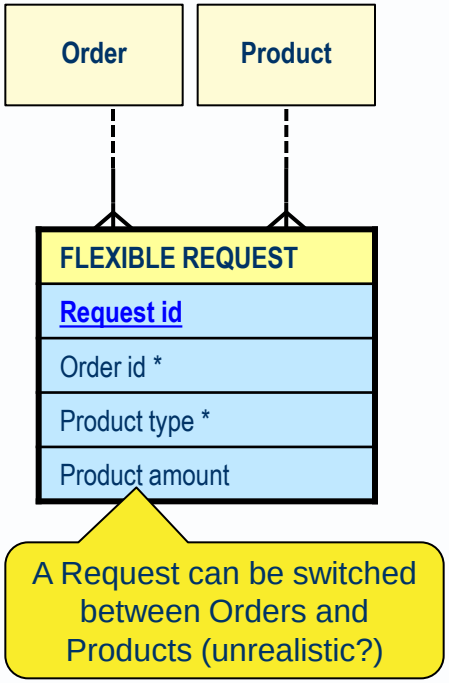
Avancier

Giving every entity type a GUID is fine, so why do the models below show other primary and foreign keys? Not because an RDBMS implementation is assumed. But because the domain model exists to specify business terms, concepts and rules; and the keys do specify what system behaviour is allowed.







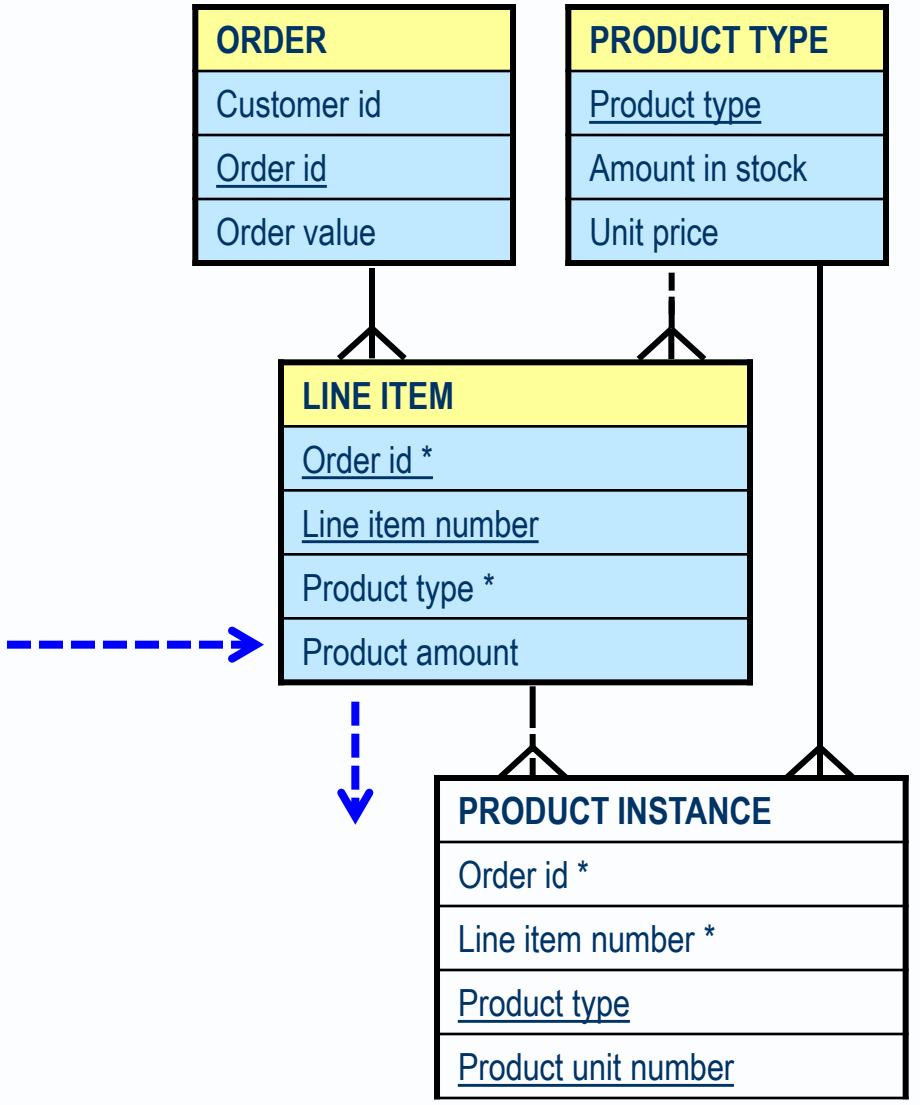


A foreign key that is *not* specified as a part of a primary key can be changed.
If the foreign key can be null, then the association is optional.
If past foreign key values must be remembered, then the association becomes N-to-N.

Copyright Avancier Ltd 2015

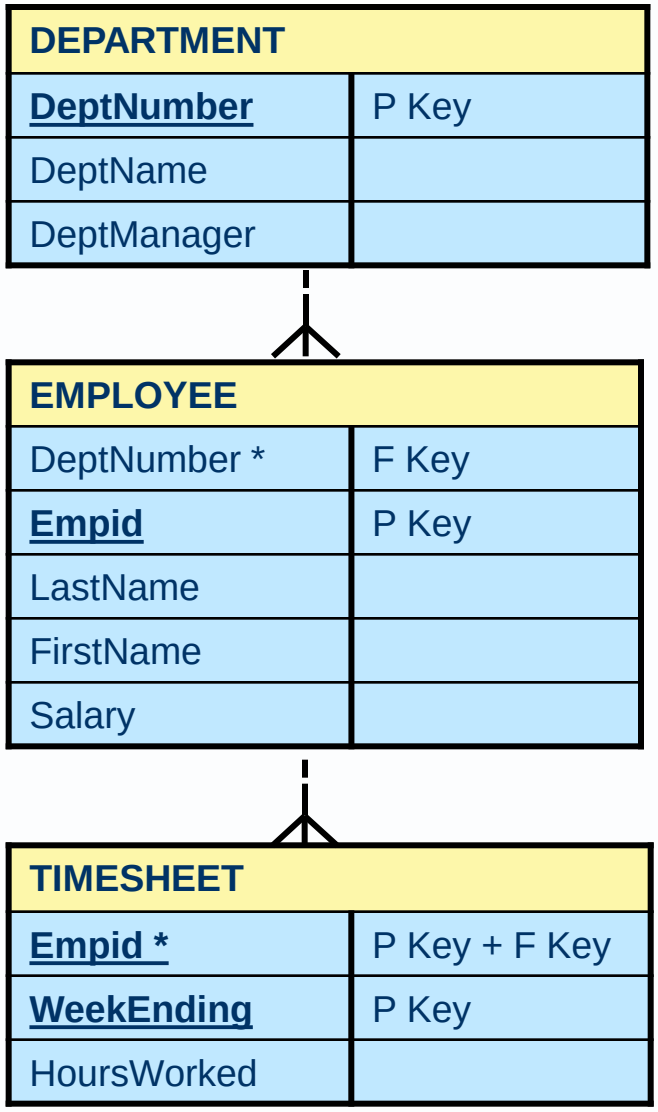
Totals as abstractions from detail entities

- A cardinal number (Product amount) can imply the existence of a more detailed entity, which might be of interest in the domain (Product instance)



Copyright Avancier Ltd 2015

Logical structural model (relational style representation)



- The internal state of a software system is definable in a structural model composed of entities, attributes and relationships
- It describes the data that must persist for the processes of a business system to work.
- It defines business terms and concepts of importance to a business.
- **This is not a database schema – or an OOP class diagram – but they might be designed to match it.**

Copyright Avancier Ltd 2015

Data types

- Requirements
- Analysis of system input/output
- **Data types**
- Modelling system state and behaviour
- Aggregate entities
- Analysing association cardinality
- Analysing class hierarchies

Copyright Avancier Ltd 2015

Atomic data types

- The atomic elements of a structural model are the data types that appear as attributes of entities.
- In most business systems, most of the data types can be found in the required input and output data structures.
- A few data types may be known only within the system, either derived from input data, or used to derive output data.

Shopping Basket
    Customer id
    Order id
    Order value
    Products Ordered
        Product type
        Product amount
    Products Ordered End
Shopping Basket

Copyright Avancier Ltd 2015

Data dictionaries

- It is often assumed that name of a data type (say Age) is sufficient to convey its meaning.

- But Age OF Customer could mean
  - years since Date OF Birth, or
  - years since Date OF Customer Registration.

- So it helps to have some kind of dictionary which describes the meaning of each data type.

Copyright Avancier Ltd 2015

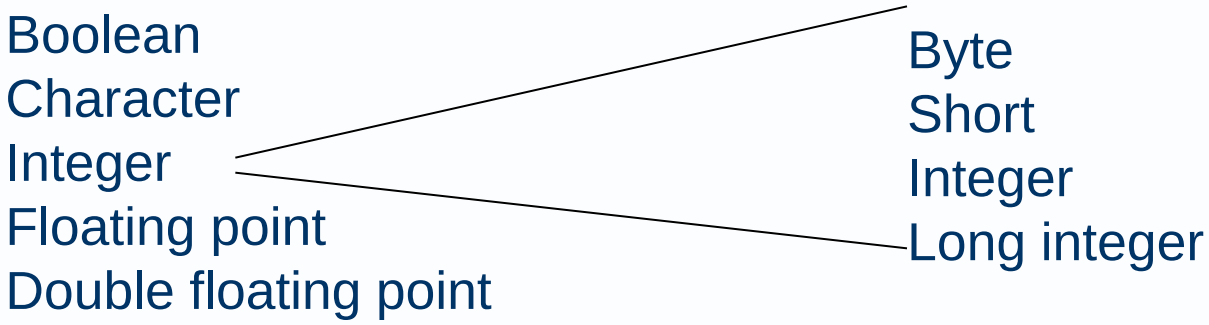
Simple data dictionary (Salesforce.com)

Avancier

Object	Description
AcceptedEventRelation	Represents invitees with the status Accepted for a given event.
Account	An individual account, which is an organization involved with your business (such as customers, competitors, and partners).
AccountContactRole	The role that a given Contact plays on an Account.
AccountFeed	Represents a single feed item on an account record detail page. This object is available in API version 18.0 and later.
AccountHistory	Represents the history of changes to the values in the fields of an account. This object is available in versions 11.0 and later.
AccountProductSharingRule	A rule that grants access to an account to users other than the owner.
AccountPartner	A relationship between two Account objects, such as partnerships or subsidiaries.
AccountShare	A sharing entry on an Account.
AccountTag	Associates a word or short phrase with an Account.
AccountTeamMember	A User who is a member of an Account team.
AccountTerritoryAssignmentRule	A rule that assigns accounts to territories.
AccountTerritorySharingRule	Rules for sharing an account within a territory.
ActivityHistory	Information about tasks and events related to an object.
AdditionalNumber	An additional phone number for a CallCenter.
AllowedEmailDomain	Represents an allowed email domain for users in your organization.
Approval	An approval request for a Contract.
Asset	An asset (such as product previously sold and installed) owned by an Account or Contact.
AssetTag	Associates a word or short phrase with an Asset.
AssignmentRule	An assignment rule associated with a Case or Lead.
AsyncApexJob	Represents an individual Apex sharing recalculation job, a batch Apex job, or method with the future annotation.
AttachedContentDocument	This read-only object contains all ContentDocument objects associated with an object.
Attachment	A file that a User has uploaded and attached to a parent object.
AuthSession	The AuthSession object represents an individual user session in your organization.
Bookmark	A link between two opportunities.
BrandTemplate	Letterhead for email templates.
BusinessHours	Specifies the business hours of your support organization. Escalation rules are run only during these hours.
BusinessProcess	A business process.
CallCenter	A single computer-telephony integration (CTI) system instance in an organization.
Campaign	A marketing campaign, such as a direct mail promotion, webinar, or trade show.

Copyright Avancier Ltd 2015

Primitive data types in Java



Type	Contains	Default	Size	Range
boolean	true or false	false	1 bit	NA
char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF
byte	Signed integer	0	8 bits	-128 to 127
short	Signed integer	0	16 bits	-32768 to 32767
int	Signed integer	0	32 bits	-2147483648 to 2147483647
long	Signed integer	0	64 bits	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	0.0	32 bits	±1.4E-45 to ±3.4028235E+38
double	IEEE 754 floating point	0.0	64 bits	±4.9E-324 to ±1.7976931348623157E+308

You tube story

Copyright Avancier Ltd 2015

Primitive data types in IBM's "OF" Language

Avancier

Classes (primitive data types)

- Amount
- Code
- Constant
- Control
- Count
- Date,
- Flag (e.g. Boolean?)
- Name
- Number
- Percent
- Text
- Title

- How to name a business data type
  - Class Connector Modifier
  - Connectors include OF and THAT IS

- In "OF" Language
  - Date OF Birth
  - Amount OF Money
  - Amount THAT IS Discount OF Product THAT IS Retail

- In "Reverse OF" Language
  - BIRTH-DATE
  - MONEY-AMOUNT
  - RETAIL-PRODUCT-DISCOUNT-AMOUNT
  - CUST-ACT-NUMBER

The vocabulary of one enterprise was c1,600 terms defined in this form

Copyright Avancier Ltd 2015

A data dictionary capturing business rules

Avancier

More elaborate than is usually recorded

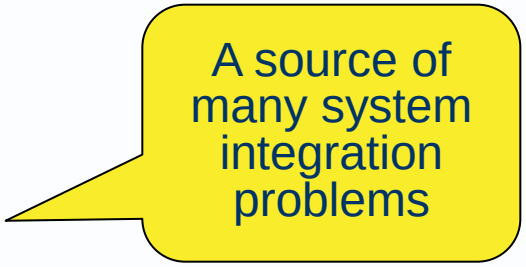
Few manage to define a dictionary at this level of detail

Term	Currency Code
Facts	Currency Code [abbreviates] Currency [which denominates a] Value
Constraints	Currency [is a] 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] Number [in the range] 0000.00 to 9999.99
Derivation	Order Value = sum of (Item Values for an Order) - Discount

Copyright Avancier Ltd 2015

Complex data types (simple data structures)

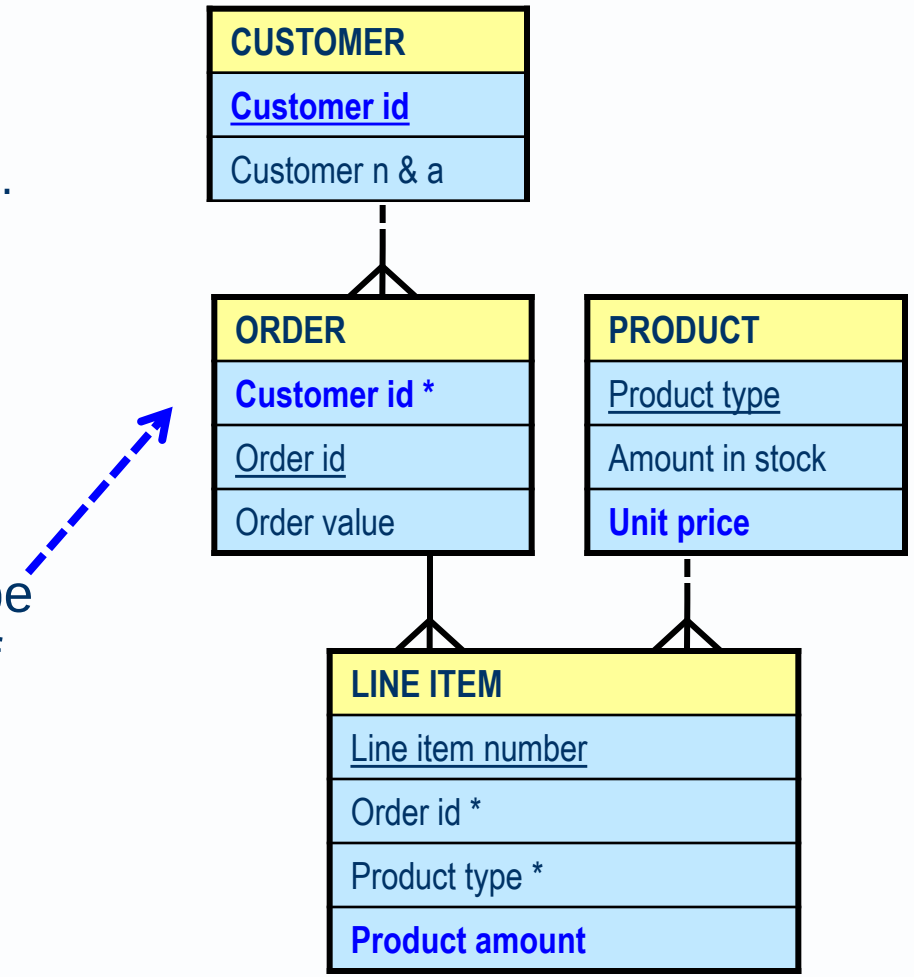
- Date
  - DD
  - MM
  - YYYY
- Person
  - Title
  - First name
  - Last name
- Address
  - Address Line 1
  - Address Line 2
  - Address Line 3
  - Address Line 4
  - County/State
  - Postcode



Copyright Avancier Ltd 2015

Turning an attribute into a foreign key

- An attribute instance
  - has a value
  - represents a fact about an entity instance.
- An attribute type
  - is a data type associated with
  - an entity type.
- Sometimes an attribute of one entity type is “raised” to become the primary key of another entity type

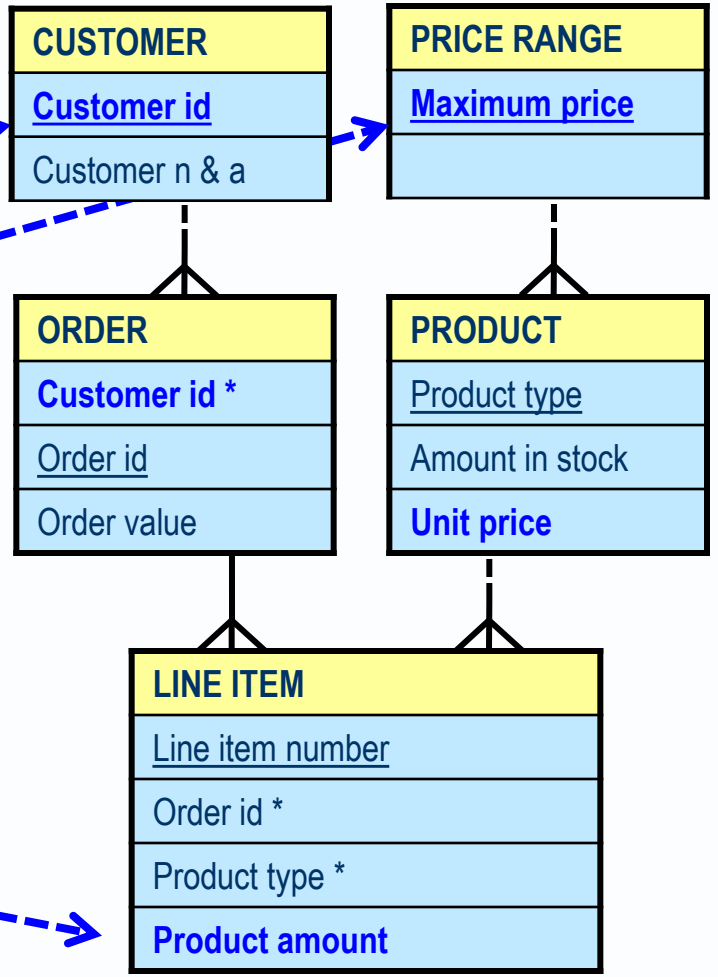


Copyright Avancier Ltd 2015

When do attributes become entities?

Avancier

- *Every* attribute might be regarded the key of an entity type whose instances are attribute values.
- Foreign key data types
  - The primary key of related entity
- Reference data types
  - E.g. values for currency, colour or price range
  - Instances created by domain experts to constrain attribute values, or facilitate access to them
  - Often modelled as entities in a structural model
- Ad hoc values
  - E.g. values for name, address or product amount
  - Instances exist only in attribute occurrences.
  - **Not** usually modelled as entities in a structural model,



Copyright Avancier Ltd 2015

When do attributes become entities?

- There is a degree of arbitrariness about deciding which attributes are represented as entities: e.g. is Currency or Address an entity?

- The challenge is to uncover what best fits the user requirements in a specific domain/context - because that is the only truth there is.

Copyright Avancier Ltd 2015

Modelling system state and behaviour

- Requirements
- Analysis of system input/output
- Data types
- **Modelling system state and behaviour**
- Aggregate entities
- Analysing association cardinality
- Analysing class hierarchies

Copyright Avancier Ltd 2015

We are modelling *activity* systems

- There is no “right” model of the real world per se, independent of context or requirements.

- Every model is context/domain specific
- built to support *the required behaviour* of a system.
- a system driven by discrete events.

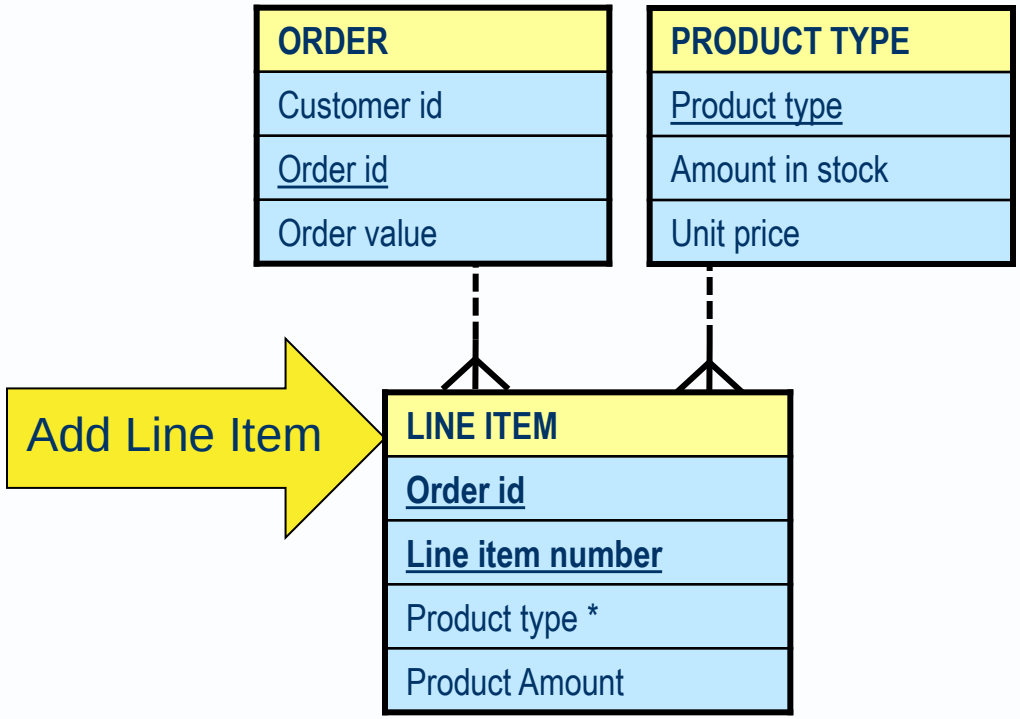
- Note: the Unified Modelling Language (2.1) is based on two premises:

1. all behavior in a modeled system is caused by actions executed by so-called “active” objects (cf. entities or components).
2. behavioral semantics only deal with event-driven, or discrete, behaviors (c.f. processes).

Copyright Avancier Ltd 2015

Discrete Event Driven Systems (DEDS)

- A system is composed of structural components (objects or actors) that
  - cooperate in event-driven processes
  - maintain information about the state of those processes.

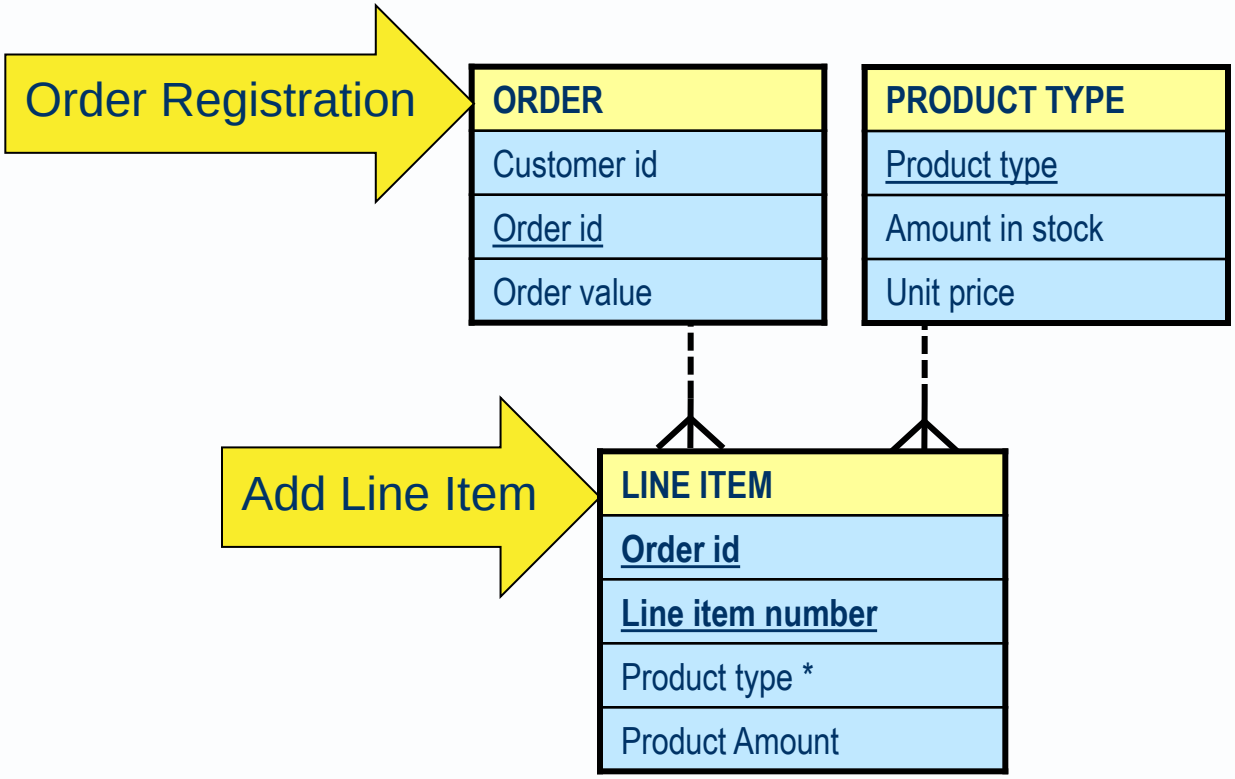


- The state of the system is driven forward by discrete events
- Each event changes the state of one or more entities
- Entity birth events and death events are especially significant.

Copyright Avancier Ltd 2015

Behavioural (event-oriented) models

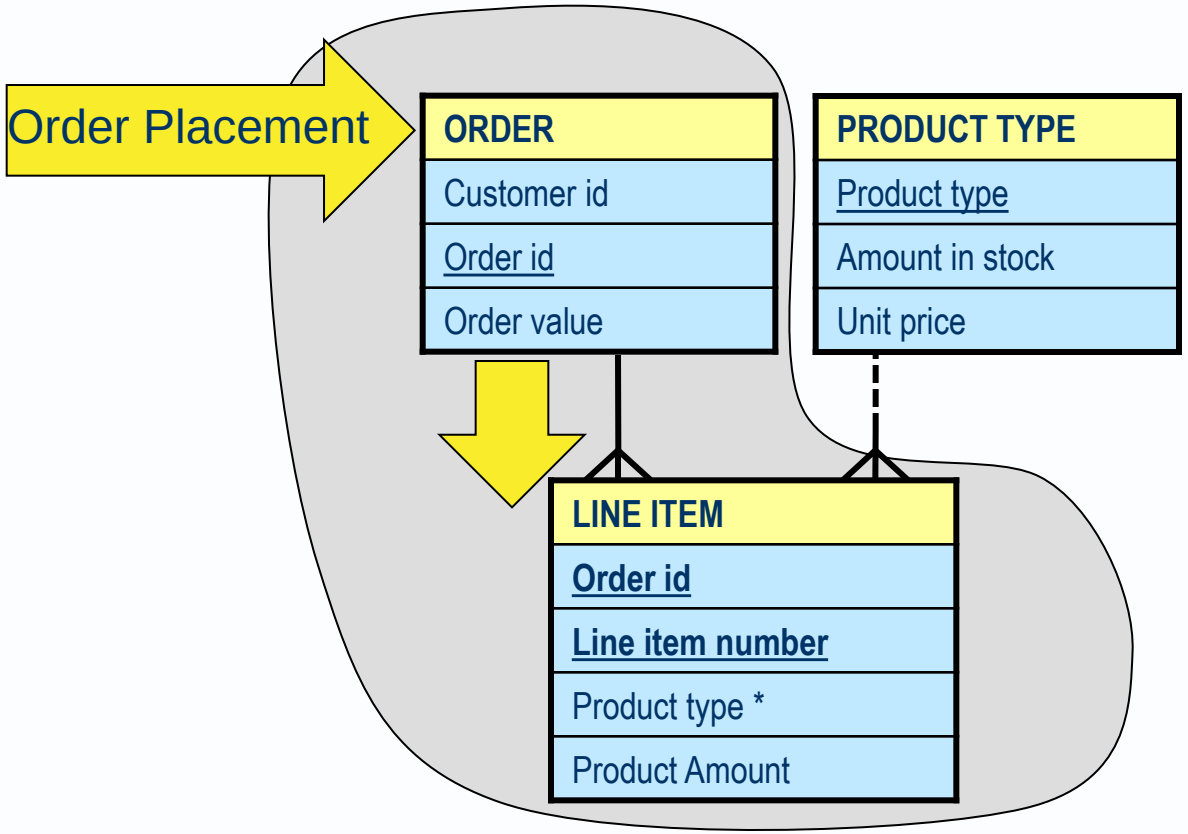
- If a structural model shows a Order entity can exist with no Line Items.
- Then this implies there are two discrete entity birth events



Copyright Avancier Ltd 2015

Aggregate birth event

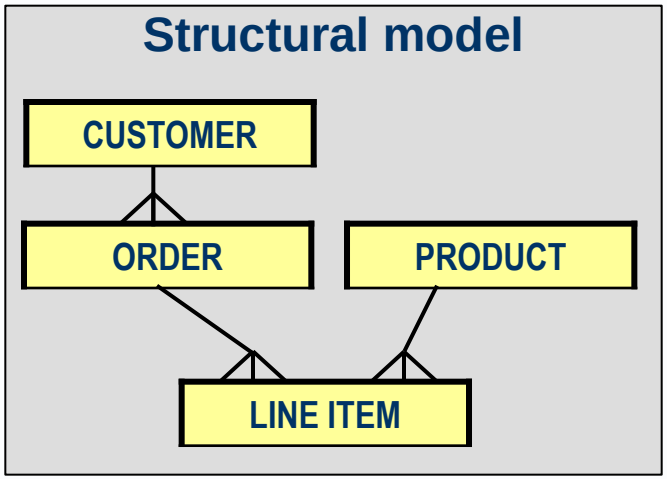
- If a structural model shows an Order entity cannot exist without one or more Line Items.
- Then there must be a discrete Order Placement event that carries the aggregate data structure into the system.
- (The DBA might still allow the database to hold an Order without Line Items, leaving the birth event rule to be imposed by software).

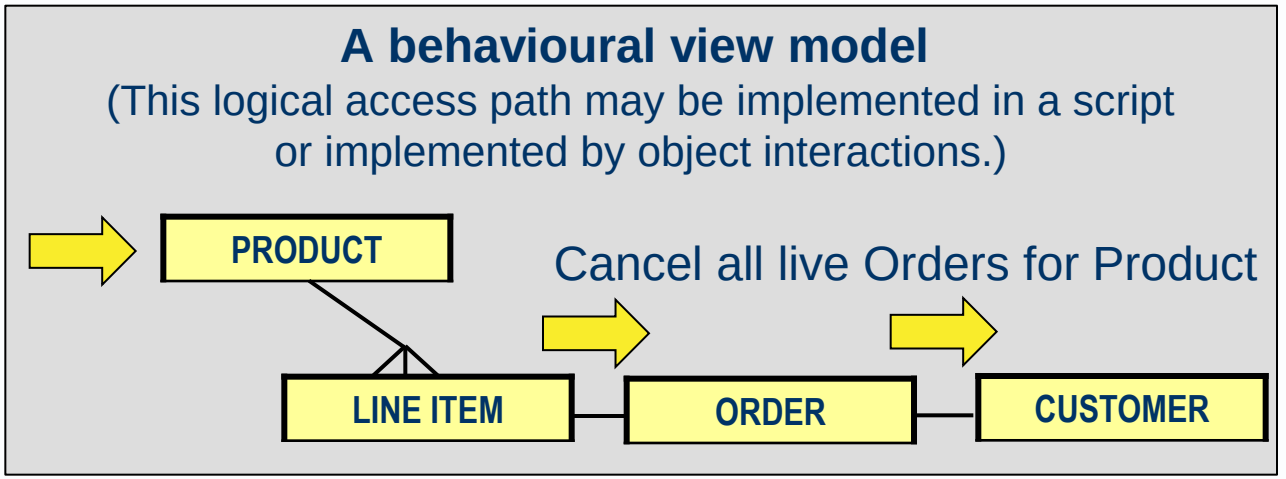


Copyright Avancier Ltd 2015

A “behavioural view model” for one event/transaction

- When caching data to process one event, you need only a *view* of the full structural model, not the whole thing





- Note
- N-to-1 associations in the full structural model become 1-to-1 associations in the “view model” for one event/transaction
- The view models for different events/transaction will overlap
- Caching data *may* imply locking it in the data store, for consistency

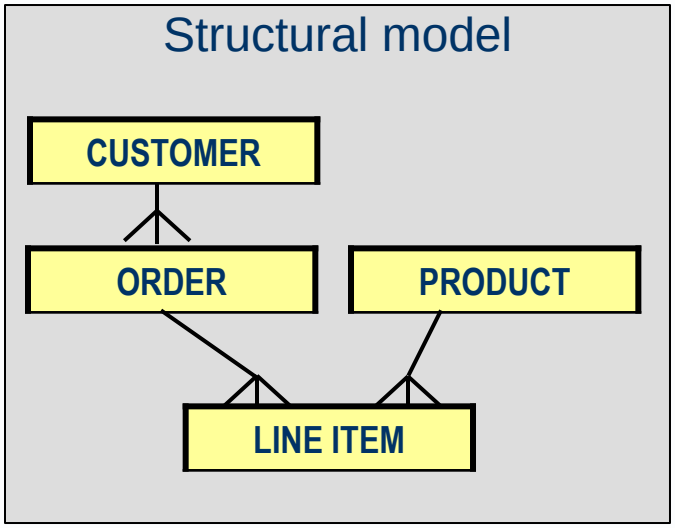
Copyright Avancier Ltd 2015

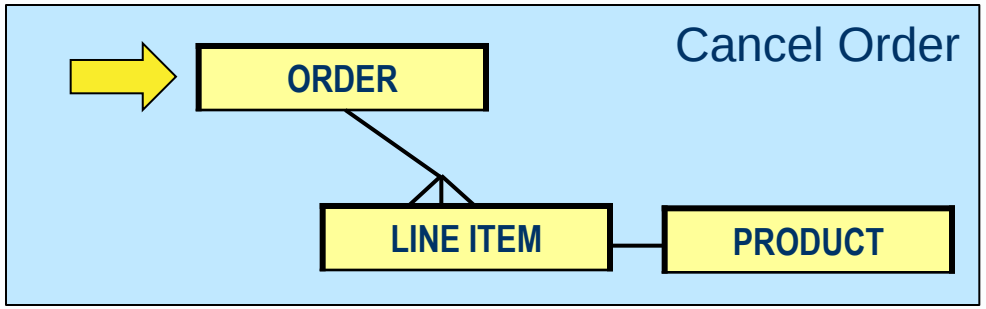
Normalisation of behaviour models
Factoring out super events (common transaction scripts)

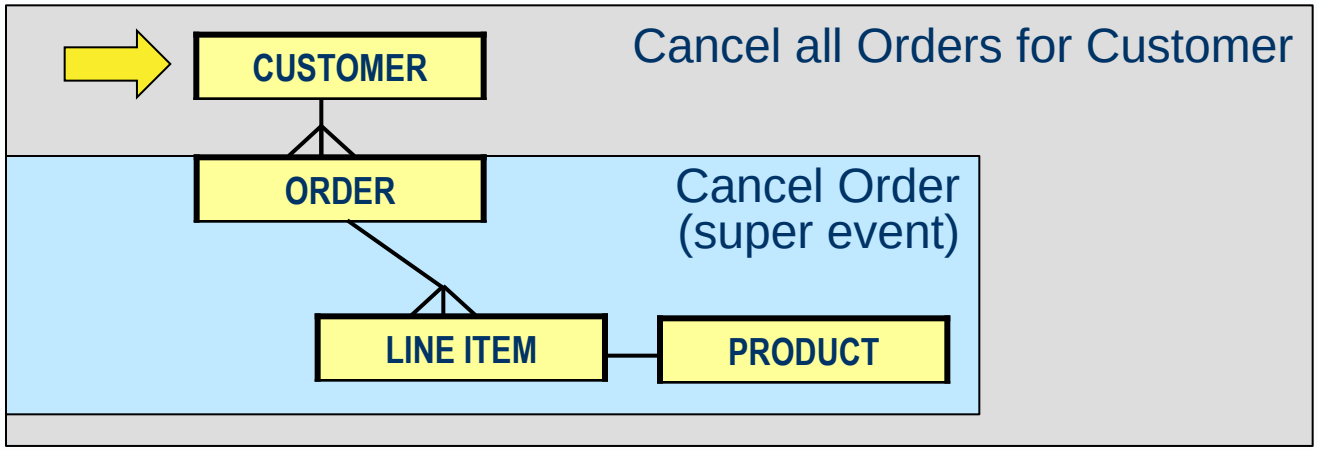
Avancier

One transaction script can inherit/extend another.

The logical/economical should grasp the opportunity for reuse







Copyright Avancier Ltd 2015

Structure and behaviour go hand in hand

- Analysis of the events that affect entities is vital
- *The access path of an event is a part of the domain model*

- But this slide show focuses more on the structural model

Copyright Avancier Ltd 2015

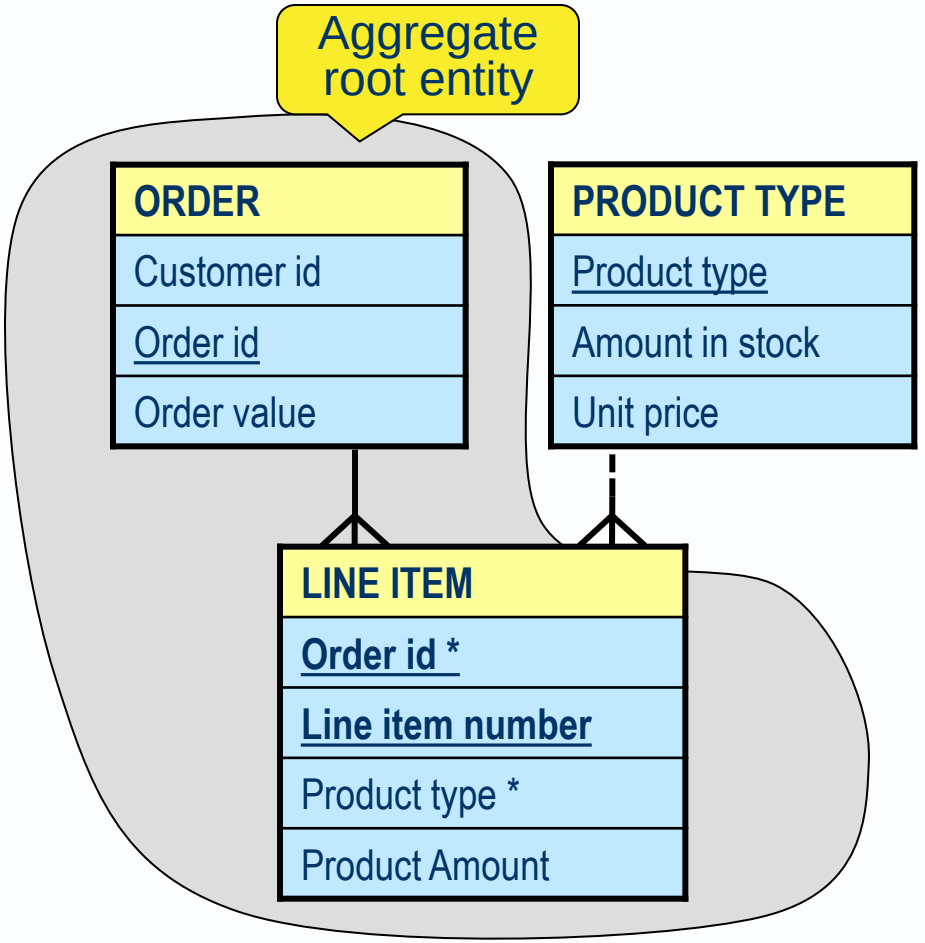
Aggregate entities

- Requirements
- Analysis of system input/output
- Data types
- Modelling system state and behaviour
- **Aggregate entities**
- Analysing association cardinality
- Analysing class hierarchies

Copyright Avancier Ltd 2015

Aggregate entities

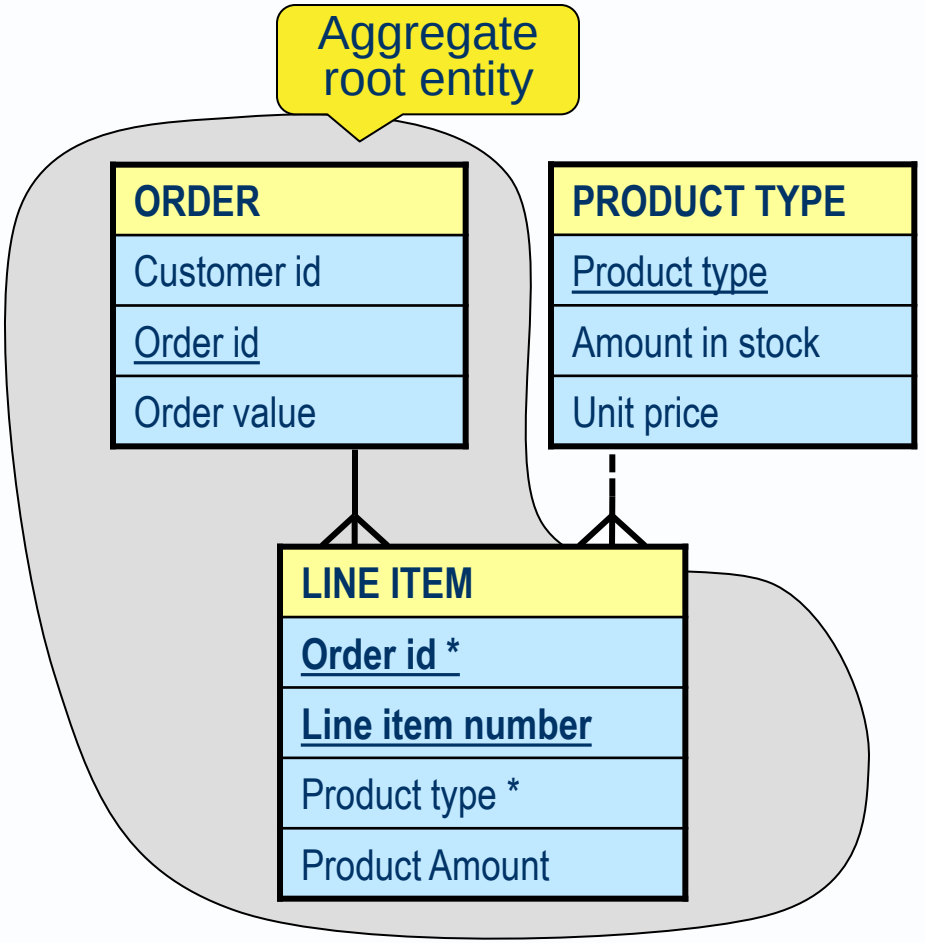
- The domain expert's choice of primary key gives you an indication as to how the business sees the entities.
- Hierarchical or composite identifiers suggest there is an “aggregate entity”, since the detail entity cannot live outside the life span of its parent.
- E.g. An Order with its Line Items may be considered as an aggregate entity if Line Items have a hierarchical primary key.



Copyright Avancier Ltd 2015

What does aggregation mean?

- Some say that aggregation implies "physical containment"
- This is a naive way of thinking what aggregation means here
- It rather more about encapsulation of one entity life history inside another.

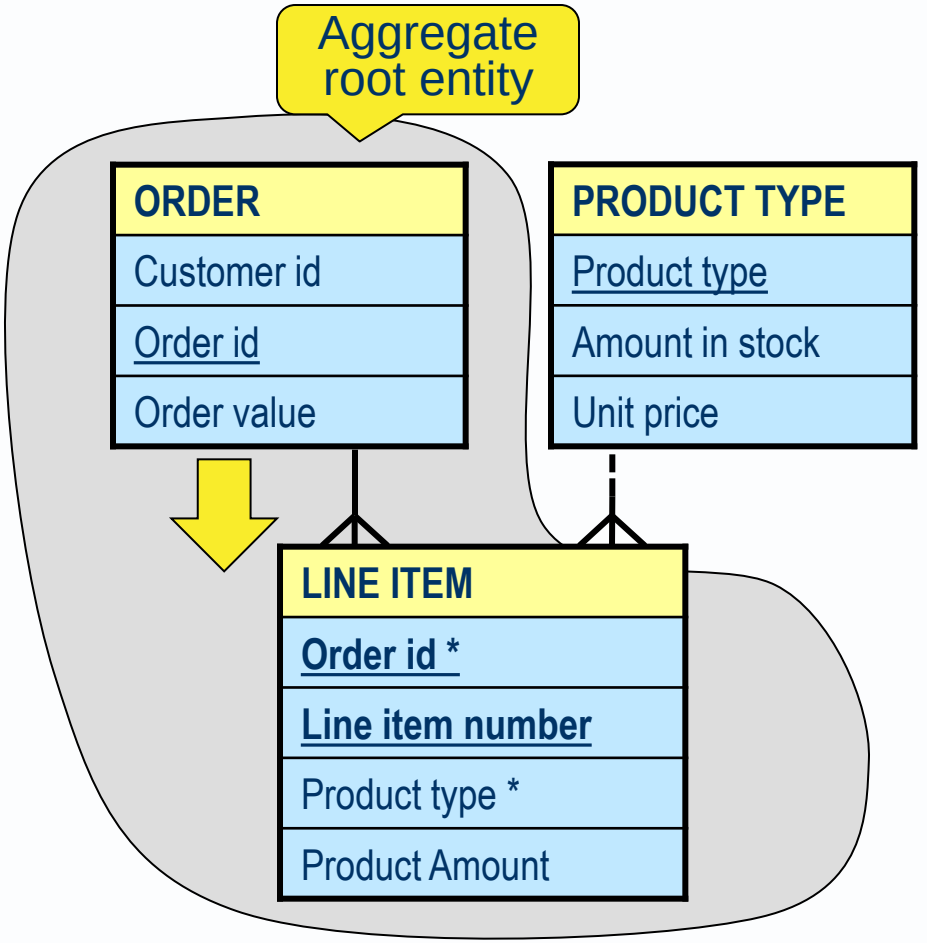


Copyright Avancier Ltd 2015

How an aggregate is regarded in an OO Domain Model

- The root is a single ENTITY contained in the AGGREGATE.
- The root is the only member of the AGGREGATE that outside objects are allowed to hold references to, although objects within the boundary may hold references to each other.

Eric Evans 2003



Copyright Avancier Ltd 2015

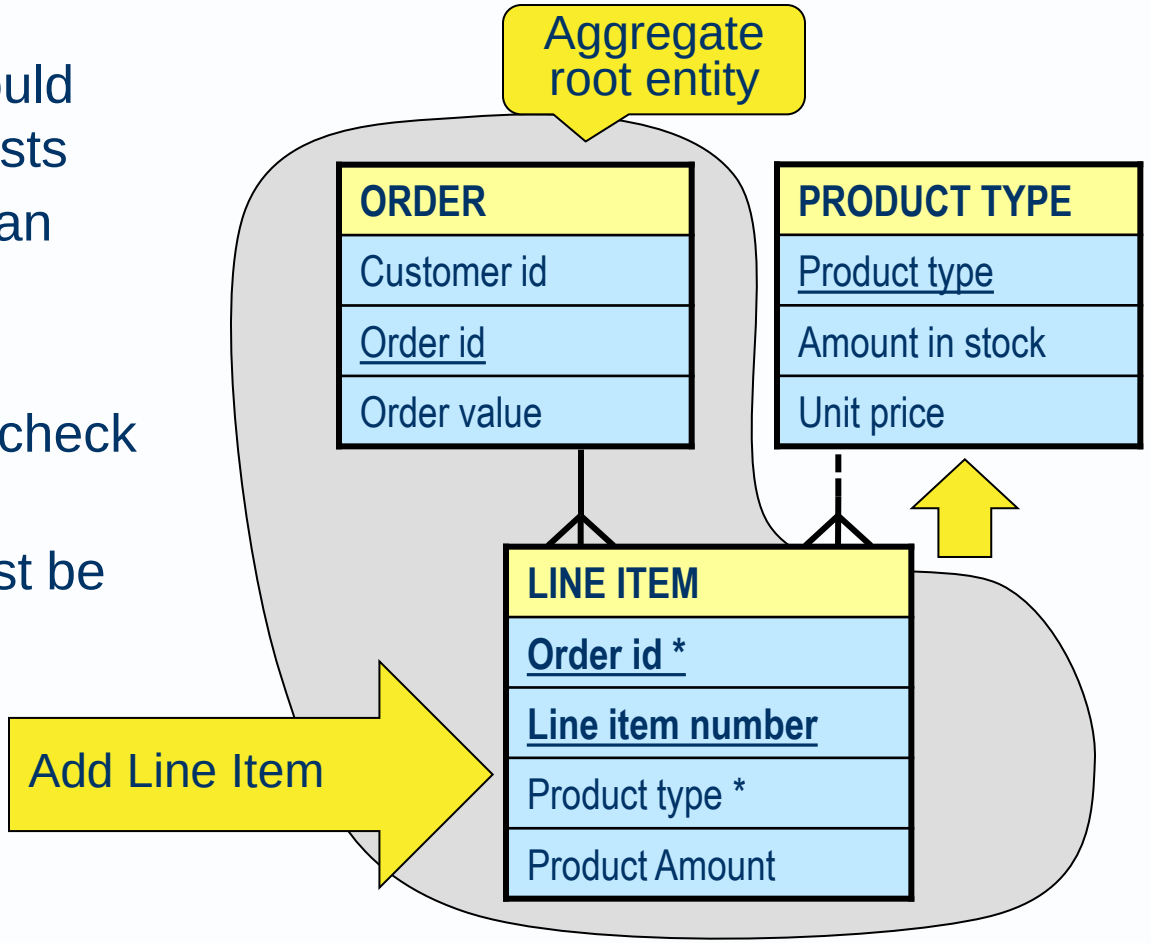
But what about referential integrity on birth events?

The birth event of Line Item should check that the Product Type exists

So, the transaction has to read an object outside the aggregate

Should we ask the database to check that rule automatically?

What if the Amount in stock must be checked also?



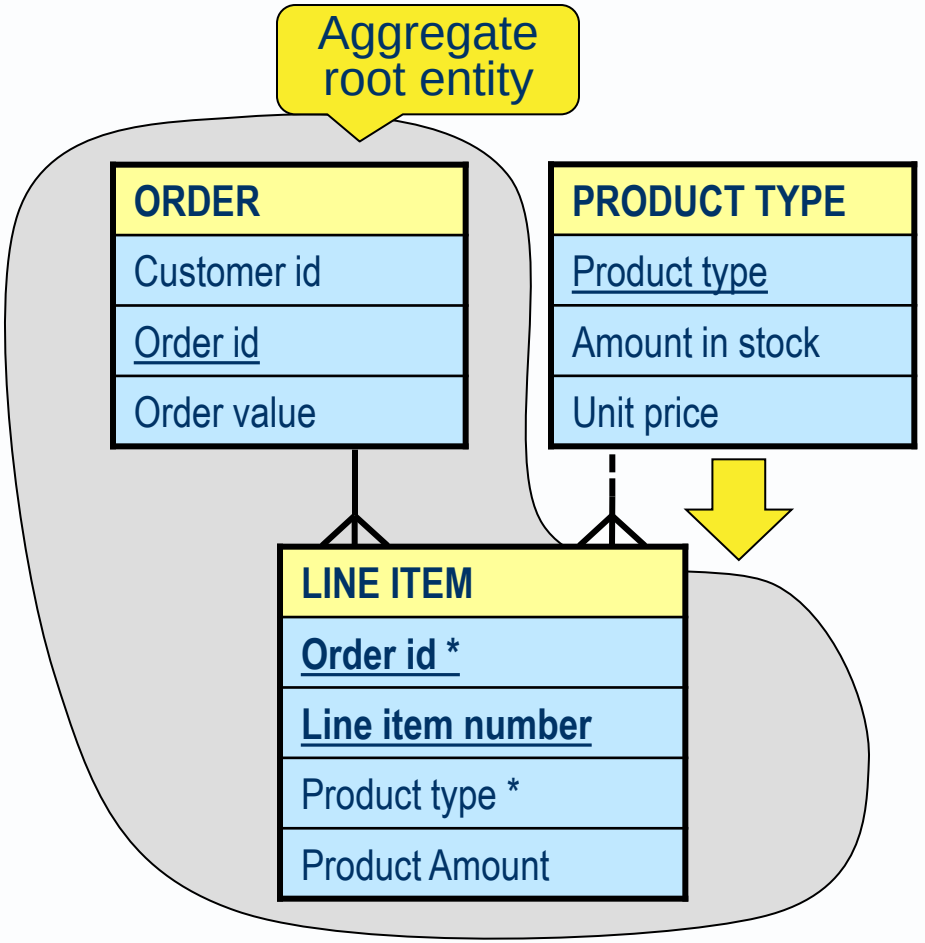
Copyright Avancier Ltd 2015

And cascading events?

Some events cascade from a master to its detail entities

And some of these may affect a detail entity in an aggregate owned by a different root entity

Suppose the product manager needs to withdraw a Product with safety issues, and so cancel all undelivered Line Items that request it.



Copyright Avancier Ltd 2015

Analysing association cardinality

- Requirements
- Analysis of system input/output
- Data types
- Modelling system state and behaviour
- Aggregate entities
- **Analysing association cardinality**
- Analysing class hierarchies

Copyright Avancier Ltd 2015

Association cardinality and optionality

- Two concepts
  - Optionality – the presence or absence of an association to another entity
  - Cardinality - the number of entity instances at the other end of an association

- *Cardinality* refers to instances of entity types. E.g.
  - One Product has **zero, one or more** Line Items.
  - One Order has **one or more** Line Items

- *Optionality* defines whether entities depend on each other's existence
- An Order cannot exist without Line Items, but a Product can.
- The association from Product to Line Item is *optional*

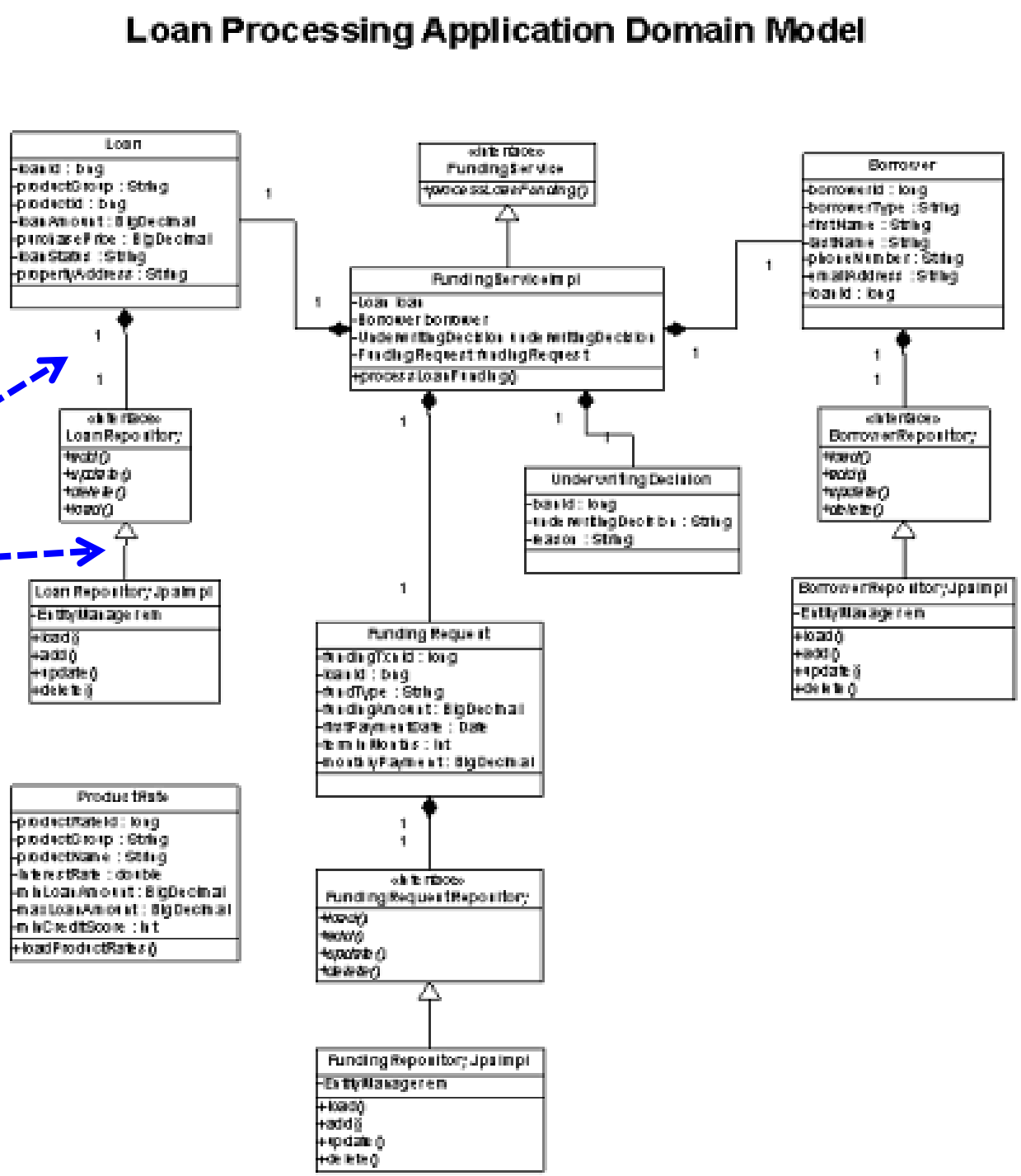
- The *optionality* of an association can be revealed by its *cardinality*

Copyright Avancier Ltd 2015

1 Borrower has 1 Loan?

- Class hierarchies appear in models of temporary and artificial things
- OO domain models often feature
- 1-1 associations
- subtypes

- This can obscure the cardinality of associations between instances of types that persist in the business domain

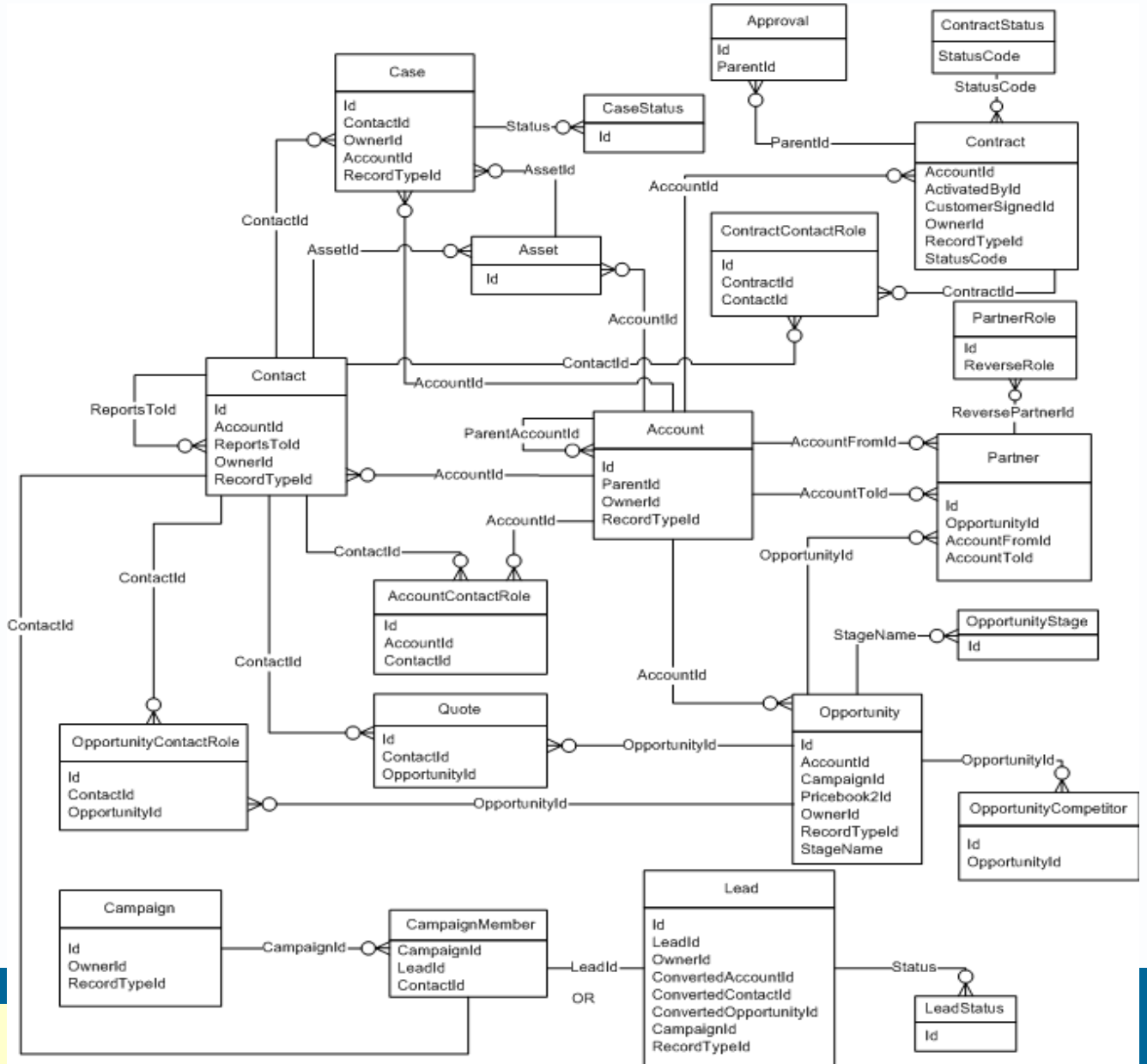


Who do business entity models mostly feature 1 to N associations?

Avancier

- In models of persistent business entities, the passage of time tends to turn
  - Subtypes into roles
  - Aggregations into associations
  - 1-1 associations into 1-N
  - 1-N associations into N-to-N with link entities

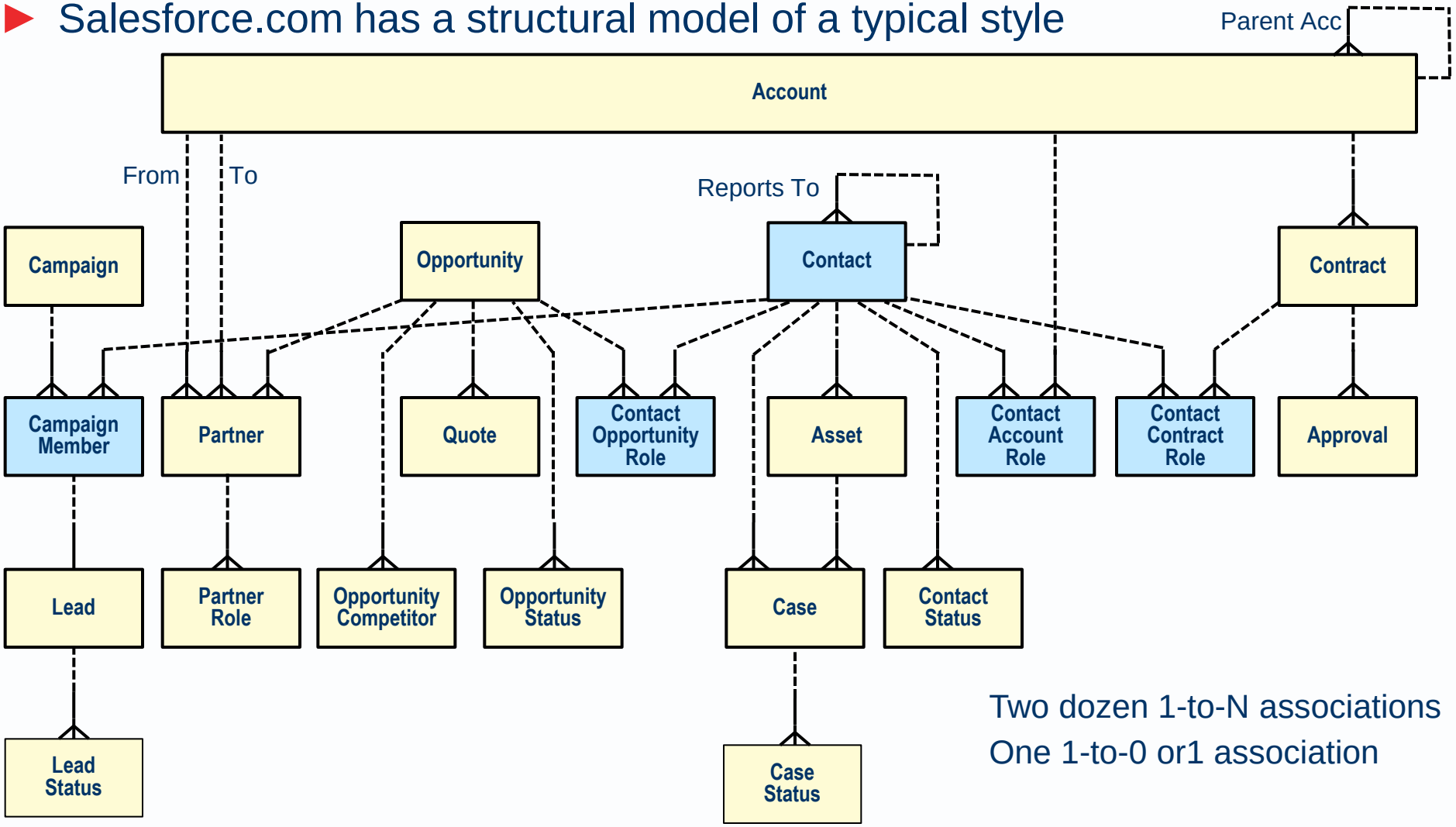
- See for example this Salesforce.com model



Drawing 1-N associations downwards clarifies the structure

Avancier

- Salesforce.com has a structural model of a typical style



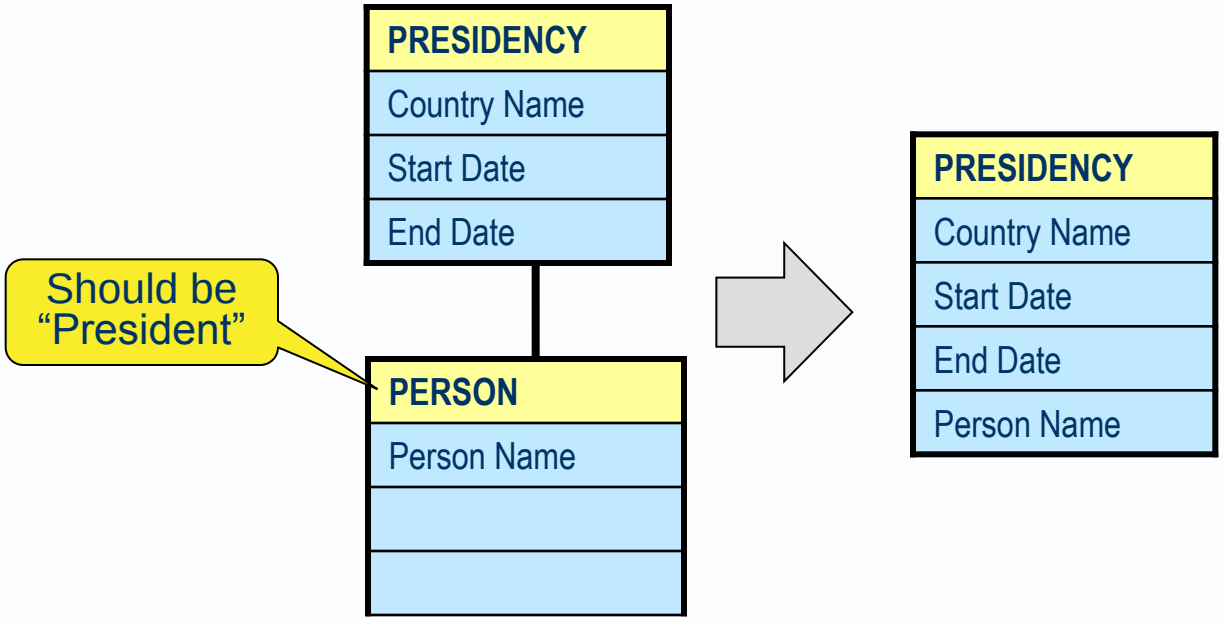
Two dozen 1-to-N associations

One 1-to-0 or1 association

Copyright Avancier Ltd 2015

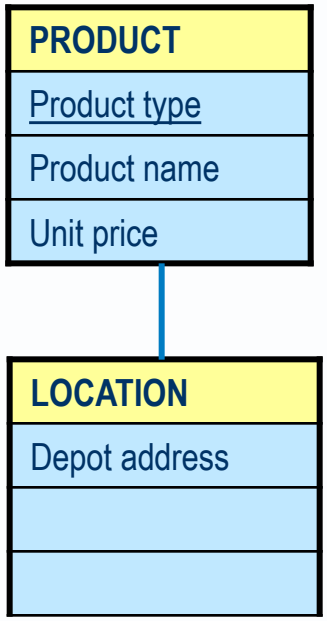
Beware 1-to-1 associations!

- If the association *truly* is 1-to-1, in the domain of interest
- then the entities are identified with each other and may be merged



Copyright Avancier Ltd 2015

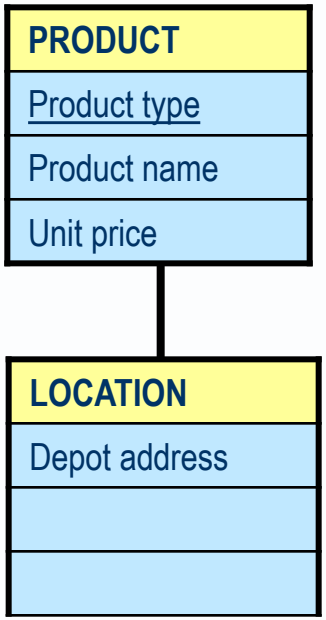
1 to 1 associations prompt rethinking of the structural model



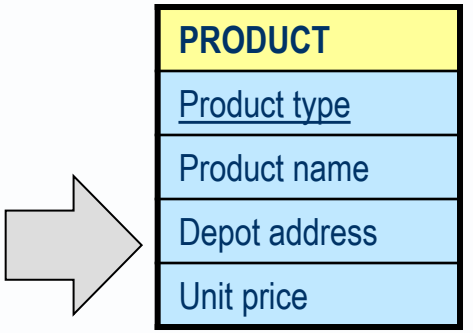
- A 1-to-1 association is so dubious that it prompts analysis and probably restructuring.
- Suppose a structural model relates two entity types thus: 1 Product to 1 Location.
- If the two entities really are inseparable, identified with each other, then the relationship between them is redundant.

Copyright Avancier Ltd 2015

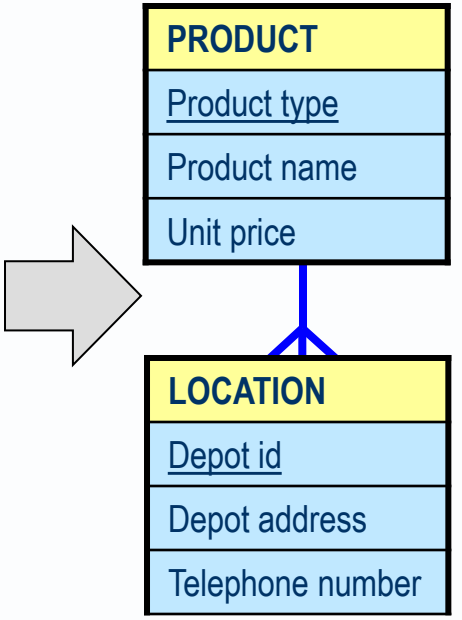
Either



- The obvious requirement is to know where a Product is stored, which indicates the Address is an attribute of Product.

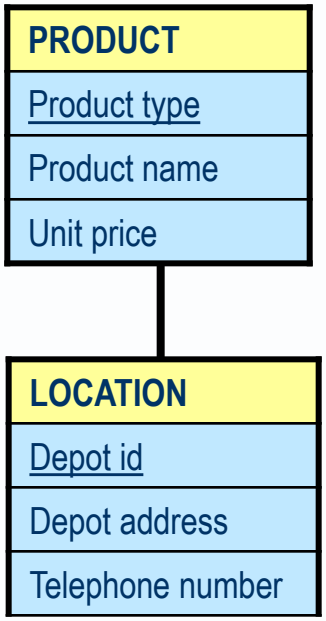


- Or else, there is 1 Product to N Locations.

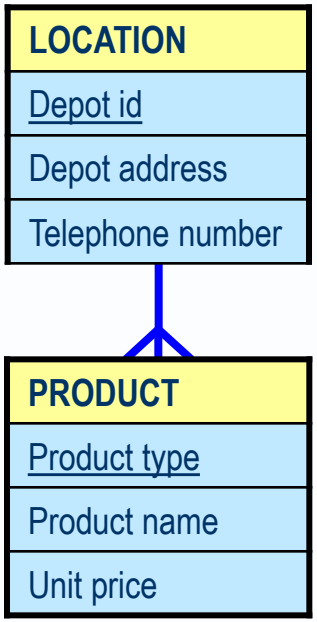


Copyright Avancier Ltd 2015

Or



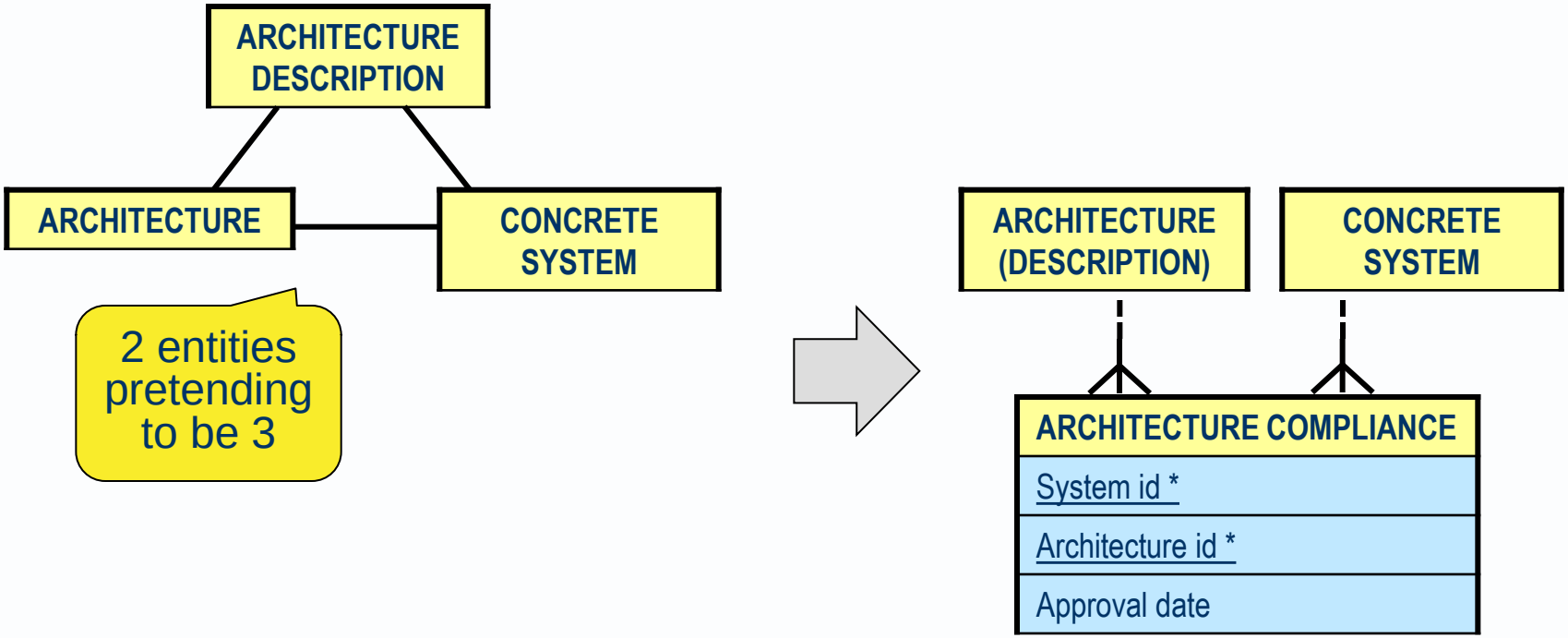
- If the requirement is to know what Product is found at a Location, then why constrain the Location to be associated with only one Product?
- The relation would be better structured as 1 Location to N Products.



Copyright Avancier Ltd 2015

An aside FYI

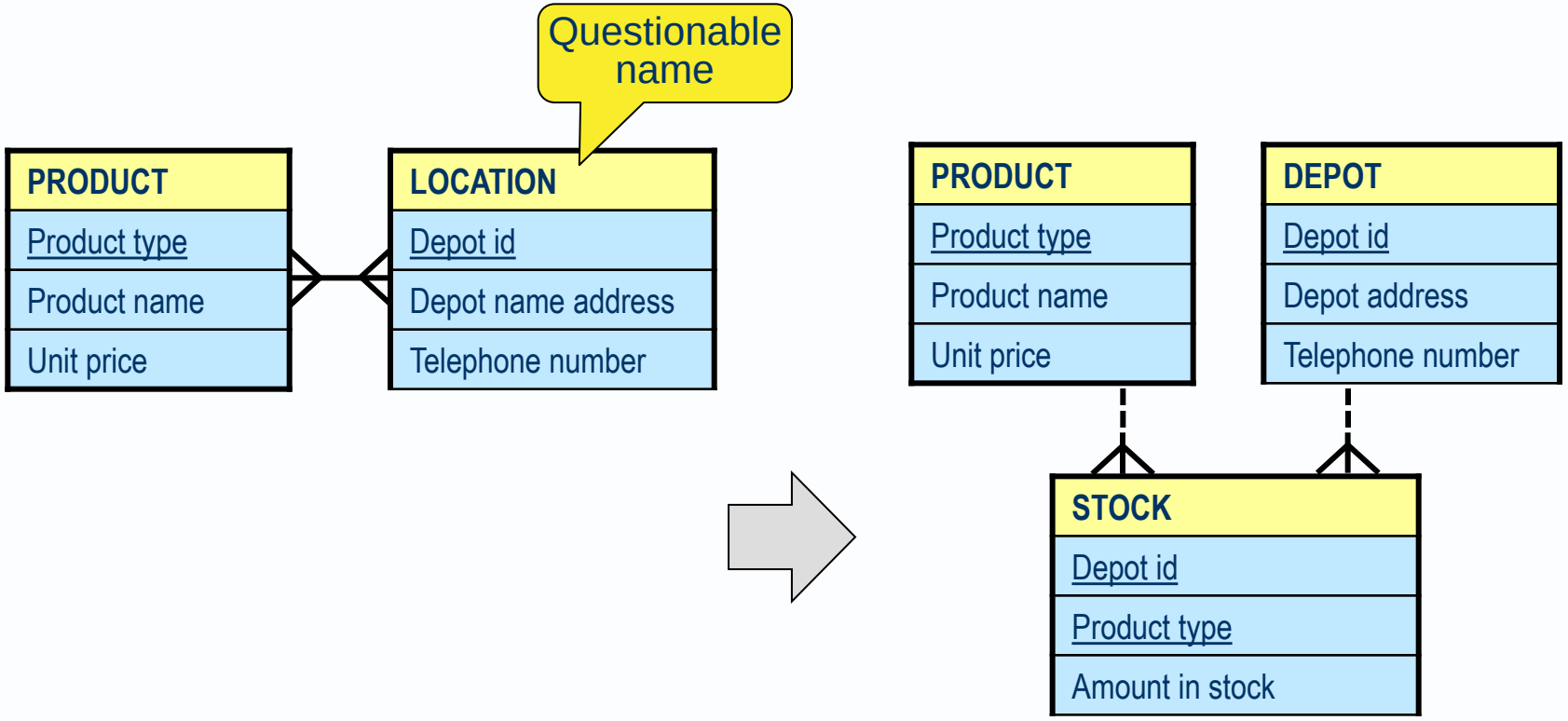
- An “architecture” is an abstract system description
- Being abstract, it can apply to more than one concrete system
- And one system can comply with several architecture descriptions



Copyright Avancier Ltd 2015

Beware N-to-N associations!

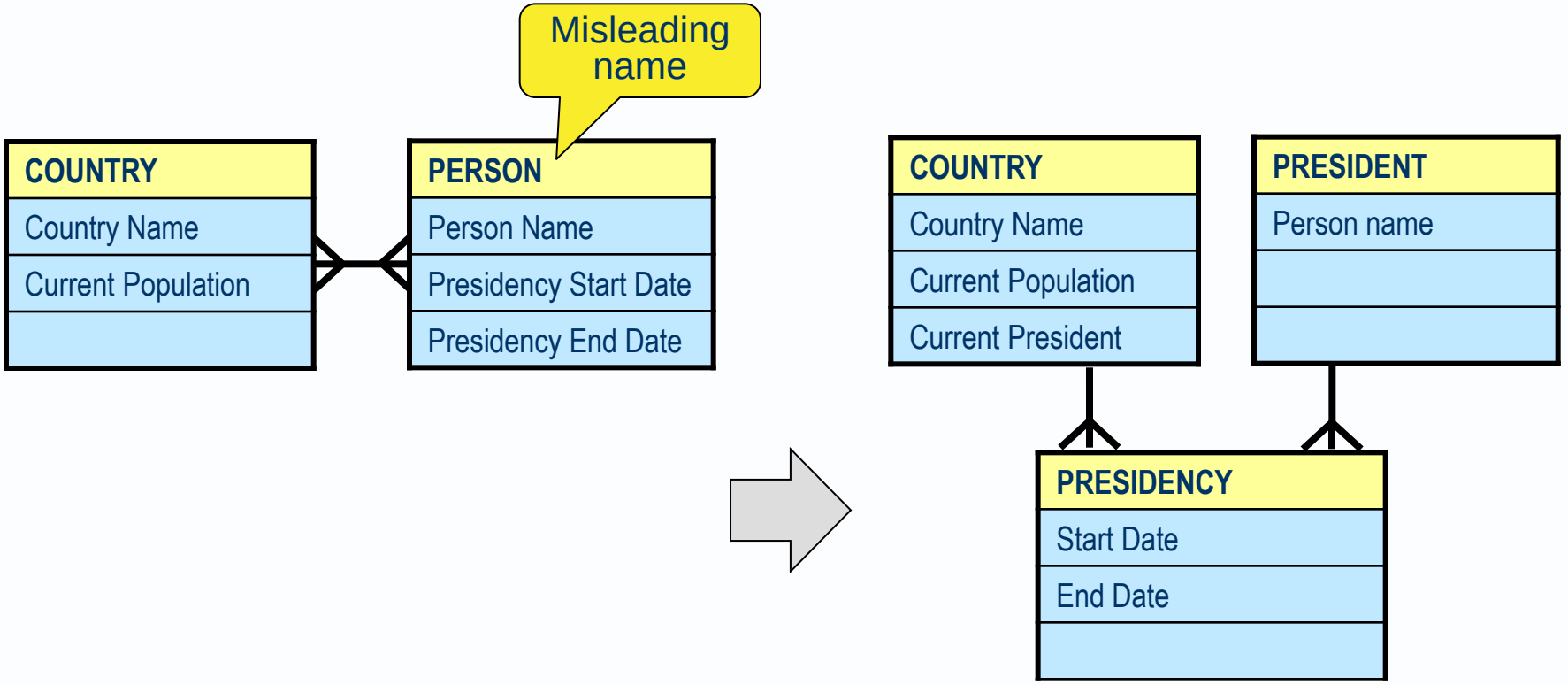
- Look for the business concept that resolves an N-to-N association
- What **event** or **thing** relates one of one entity type to one of the other?”
- The result is a model of 1-to-N associations, navigable in both directions



Copyright Avancier Ltd 2015

Beware N-to-N associations!

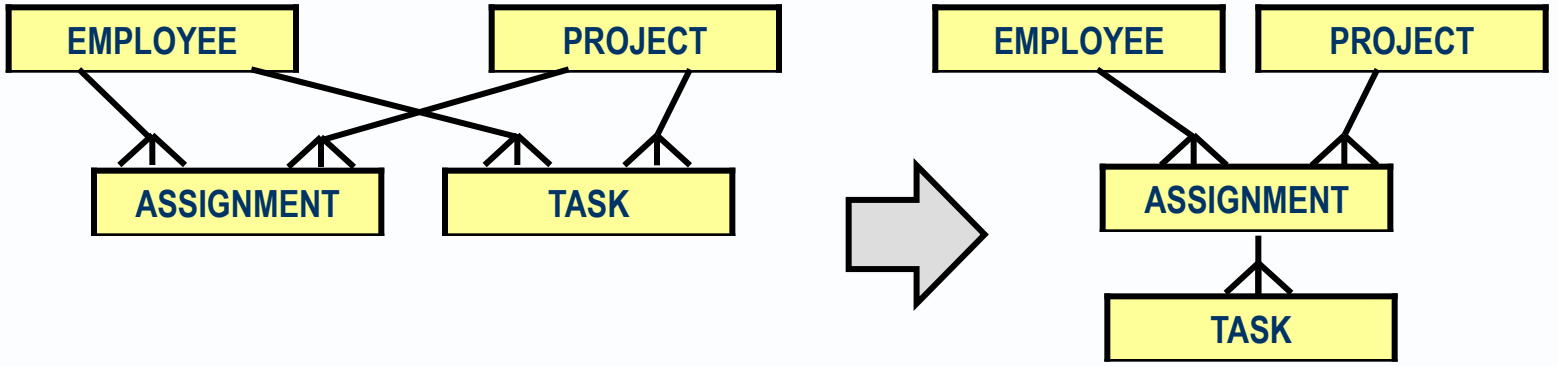
- Look for the business concept that resolves an N-to-N association
- What **event** or **thing** relates one of one entity type to one of the other?"
- The result is a model of 1-to-N associations, navigable in both directions



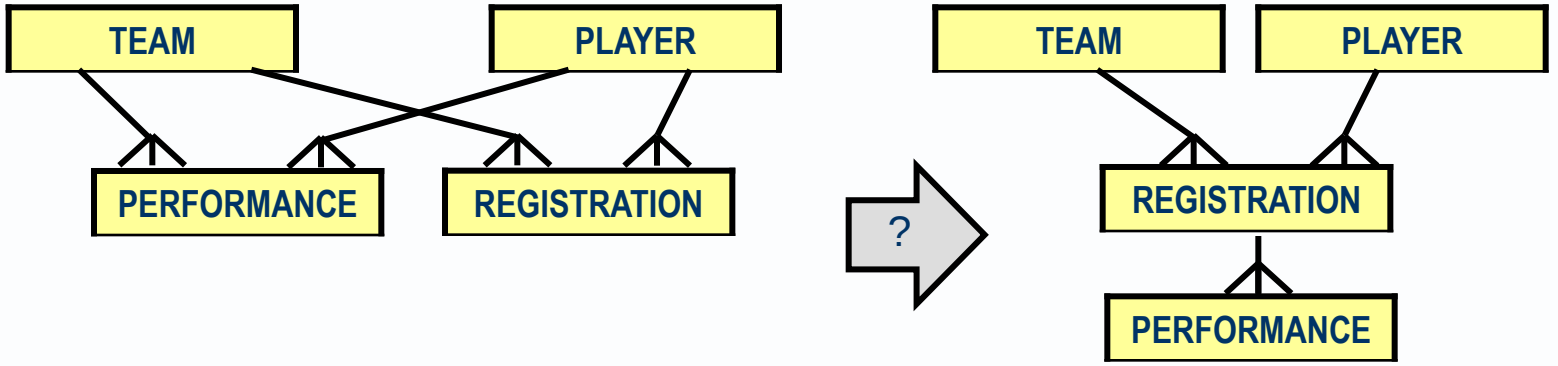
Copyright Avancier Ltd 2015

How logical analysis regards Double N-to-N associations

- Look for a 1-N constraint between the two link entities



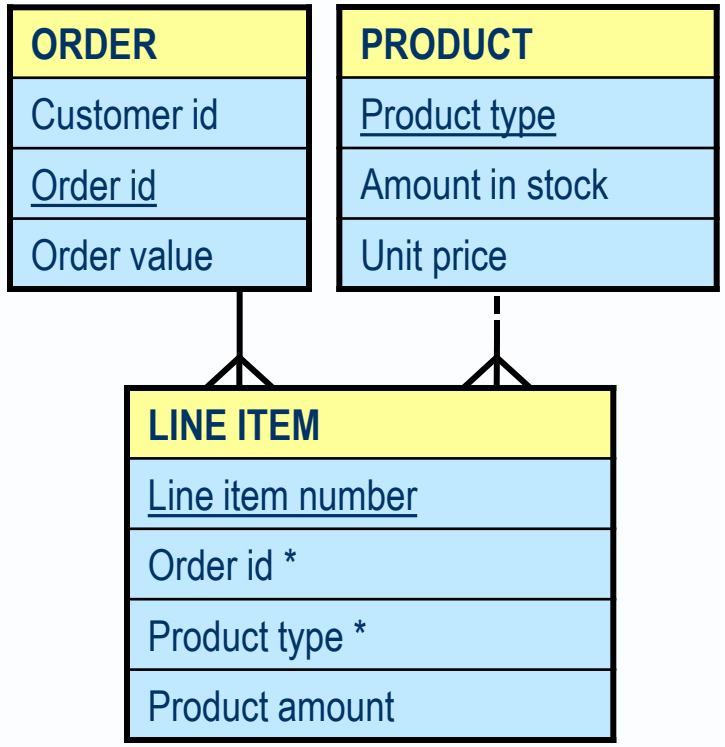
- Should this double V shape become this Y shape?



Copyright Avancier Ltd 2015

How cardinalities affect the naming of entity types

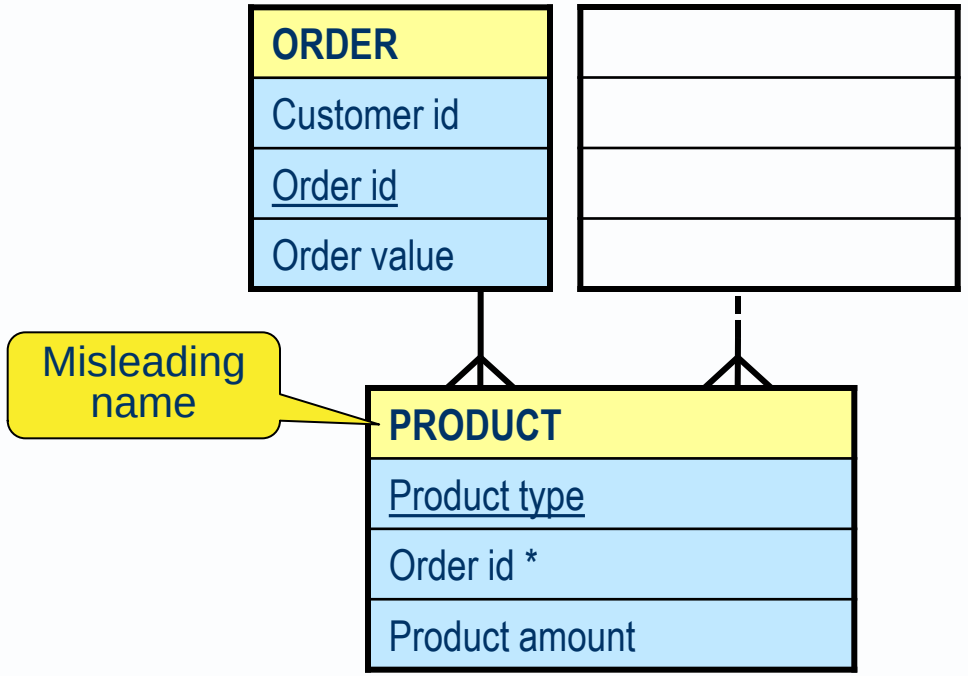
- Looking at a many-to-many association from one end only is fine
- But often leads to misnaming of entity types.



Copyright Avancier Ltd 2015

Narrow domain model with misleading entity name

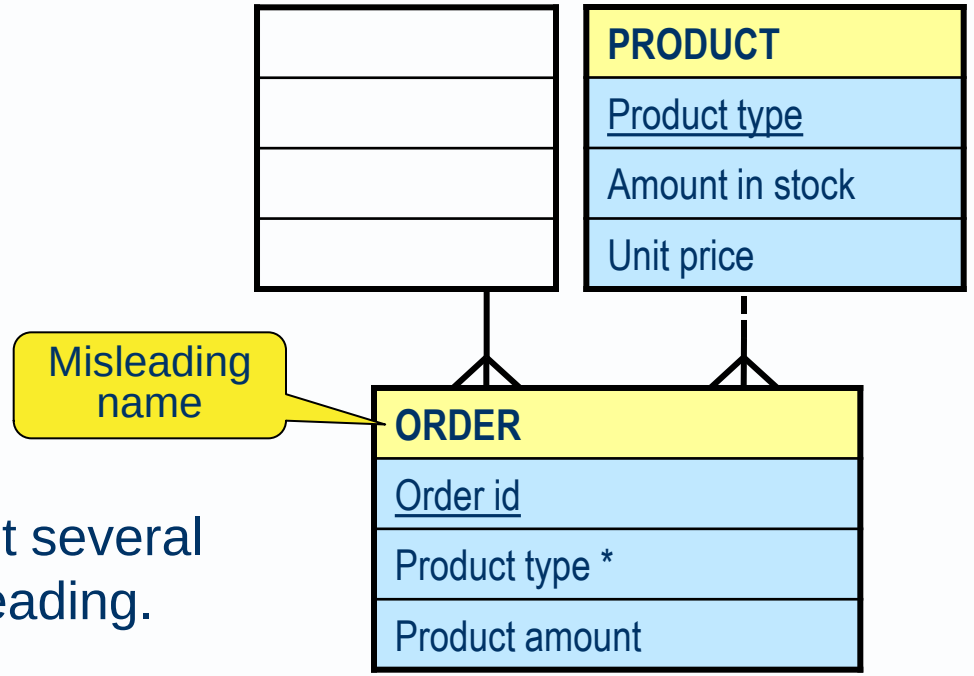
- For **Order management** you don't need to know if one Product appears in several Orders.
- So this structure may suffice
- But if the same Product may be requested on different Orders, the name Product is misleading.



Copyright Avancier Ltd 2015

Narrow domain model with misleading entity name

- As a **Product manager**, you don't need to know when one Order is for many Products.
- So this structure may suffice
- But if the same Order may request several Products, the name Order is misleading.



Copyright Avancier Ltd 2015

Analysing a class hierarchy

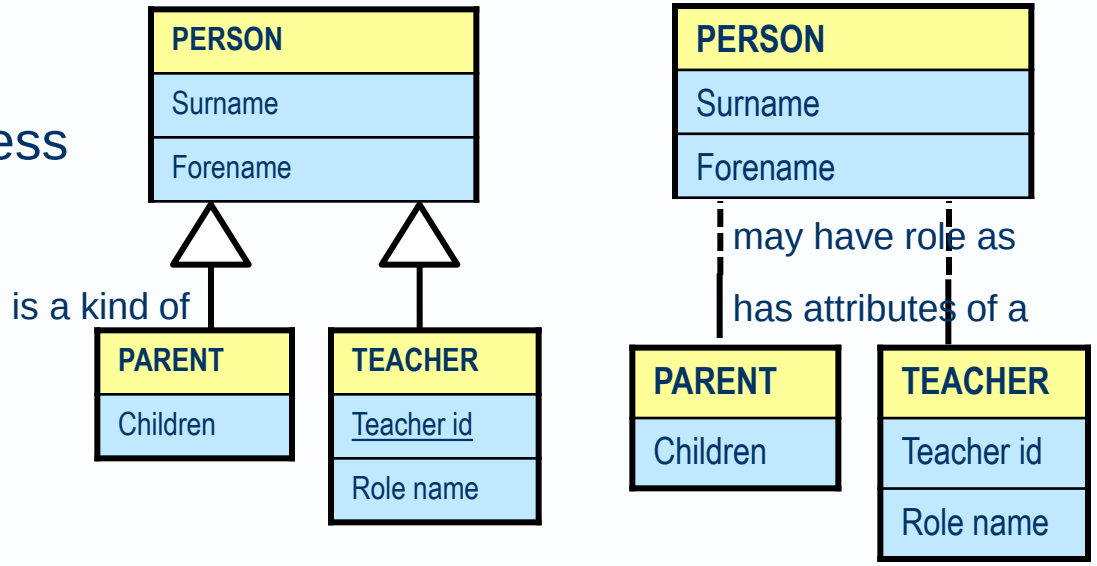
- Requirements
- Analysis of system input/output
- Data types
- Modelling system state and behaviour
- Aggregate entities
- Analysing association cardinality
- **Analysing class hierarchies**

- Class hierarchies appear in models of temporary and artificial things
- But in models of persistent business entities, the passage of time tends to turn
  - Subtypes into roles
  - Aggregations into associations
  - 1-1 associations into 1-N
  - 1-N associations into N-to-N with link entities

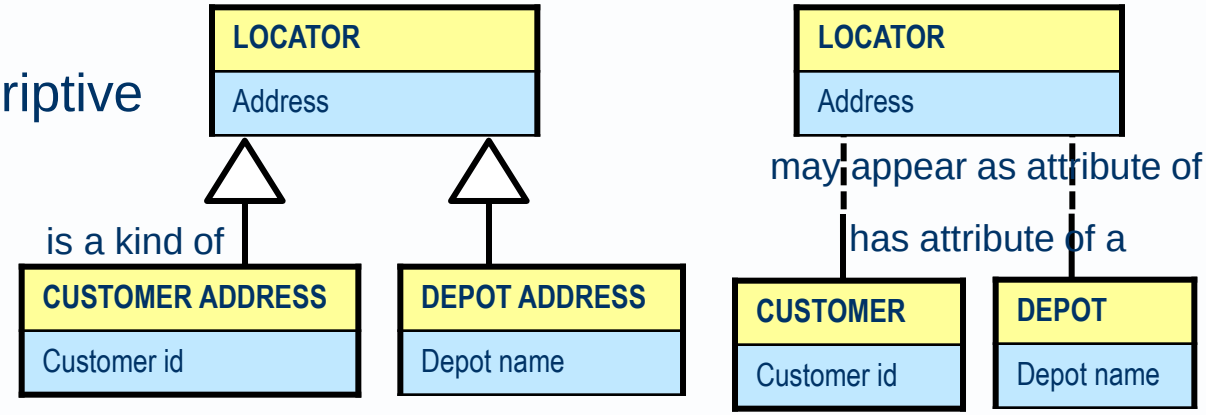
Copyright Avancier Ltd 2015

Beware “Person” and “Location”

- We rarely model human beings
- We model their *roles* in a business
  - Employee
  - Customer
  - Supplier …



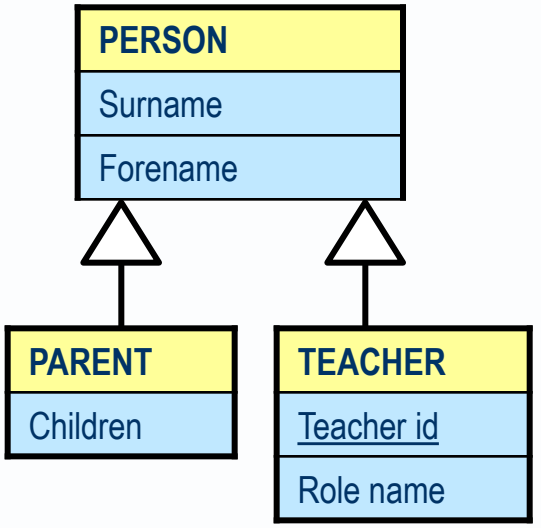
- We rarely maintain addresses as distinct entities
- We maintain them as descriptive attributes



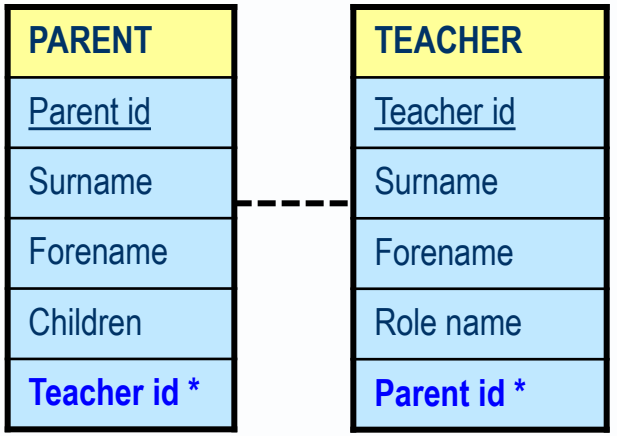
Copyright Avancier Ltd 2015

Analysis questions include

- 1) Do the subtypes have their own primary key ranges in the business?
- 2) Is it important in this system to know one Person is both Parent and Teacher?
- 3) If yes, is there a way to uniquely identify a Person who plays both roles?
- 4) Can we prevent one Person being recorded twice?

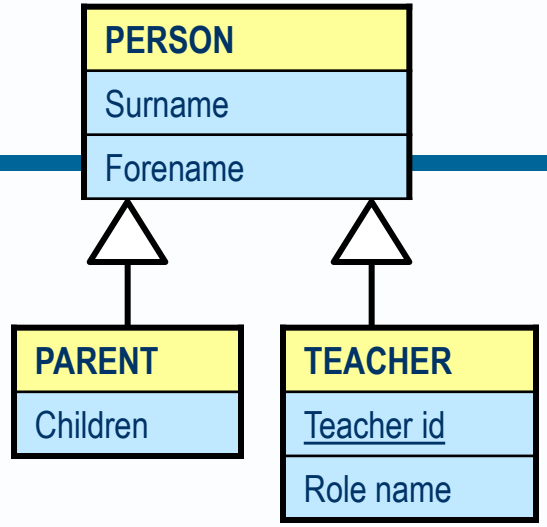


- Even if no, yes, yes and yes
- You could still cross-refer from one role to the other using an optional foreign key



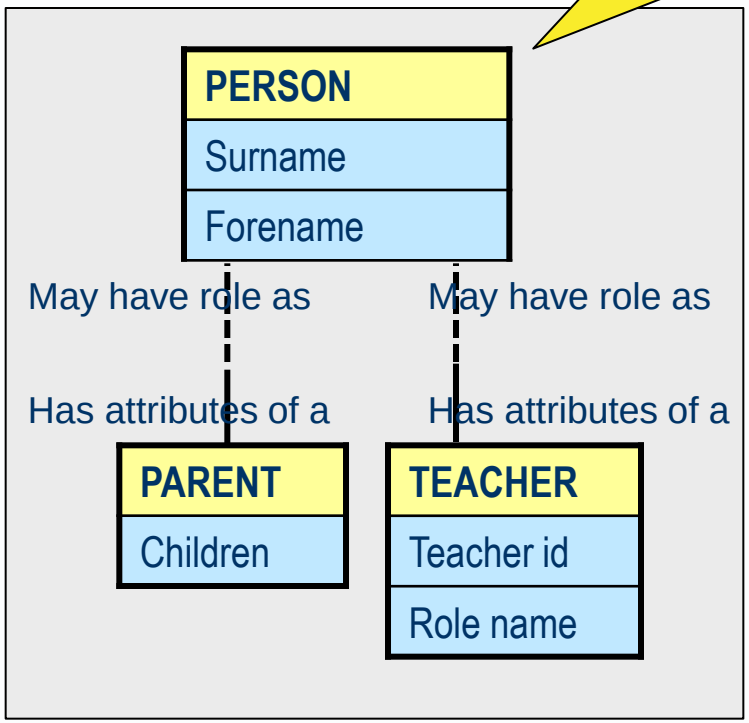
Copyright Avancier Ltd 2015

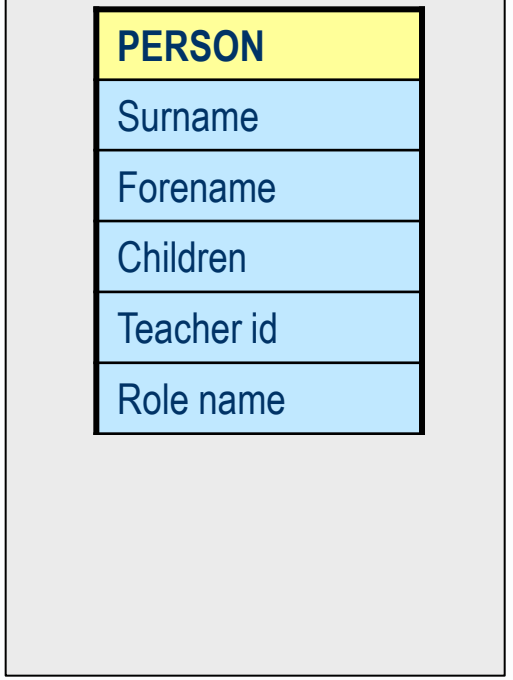
Viewing subtypes into roles

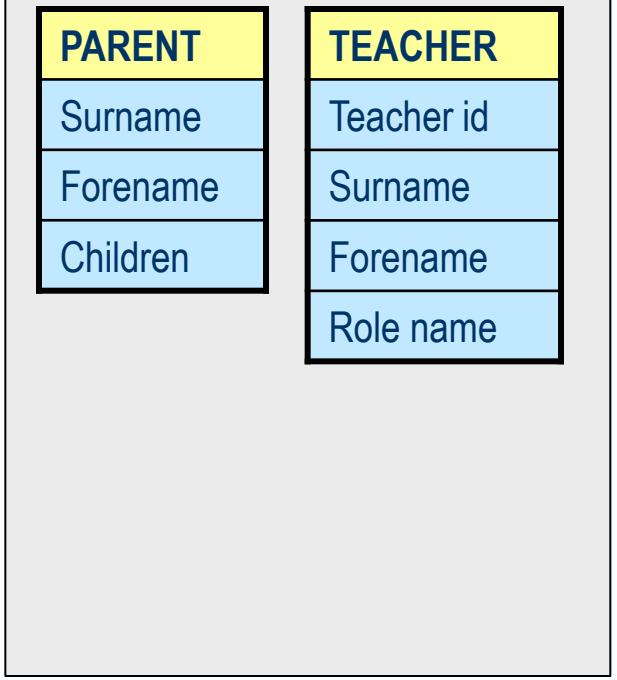


- Three possible mappings below

A better domain model?





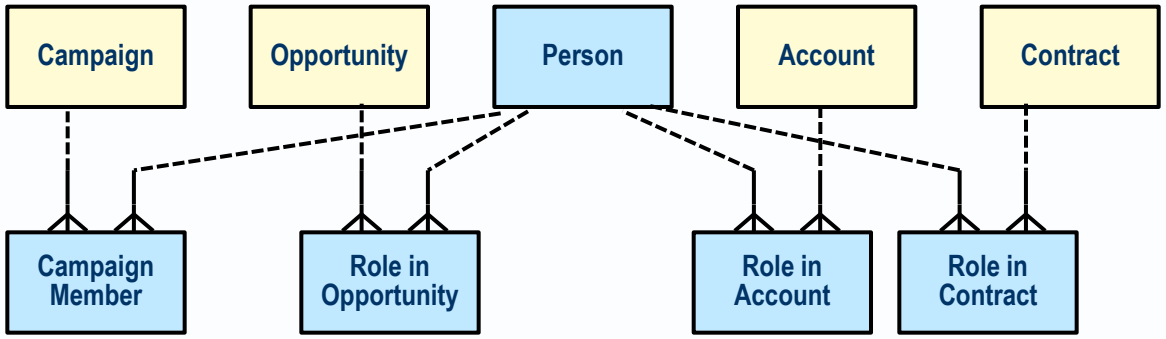


Copyright Avancier Ltd 2015

Other analysis questions

- Are you interested in people before or after their roles in the business?
- Could a person play more than one role of each kind?

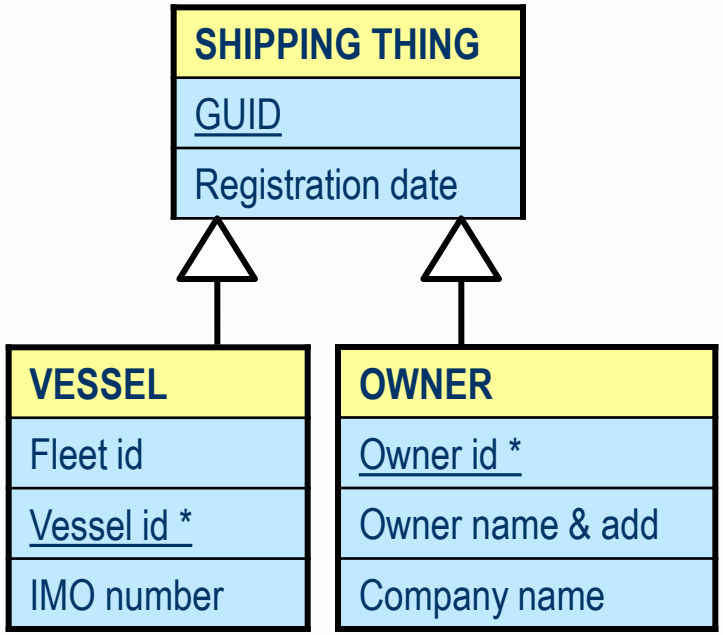
- Defining a Person as a super type of Roles is not a good practice
- A better and more flexible way to model multiple Roles of a Person was shown in the Salesforce.com model



Copyright Avancier Ltd 2015

Generalisation (forms a container of common properties)

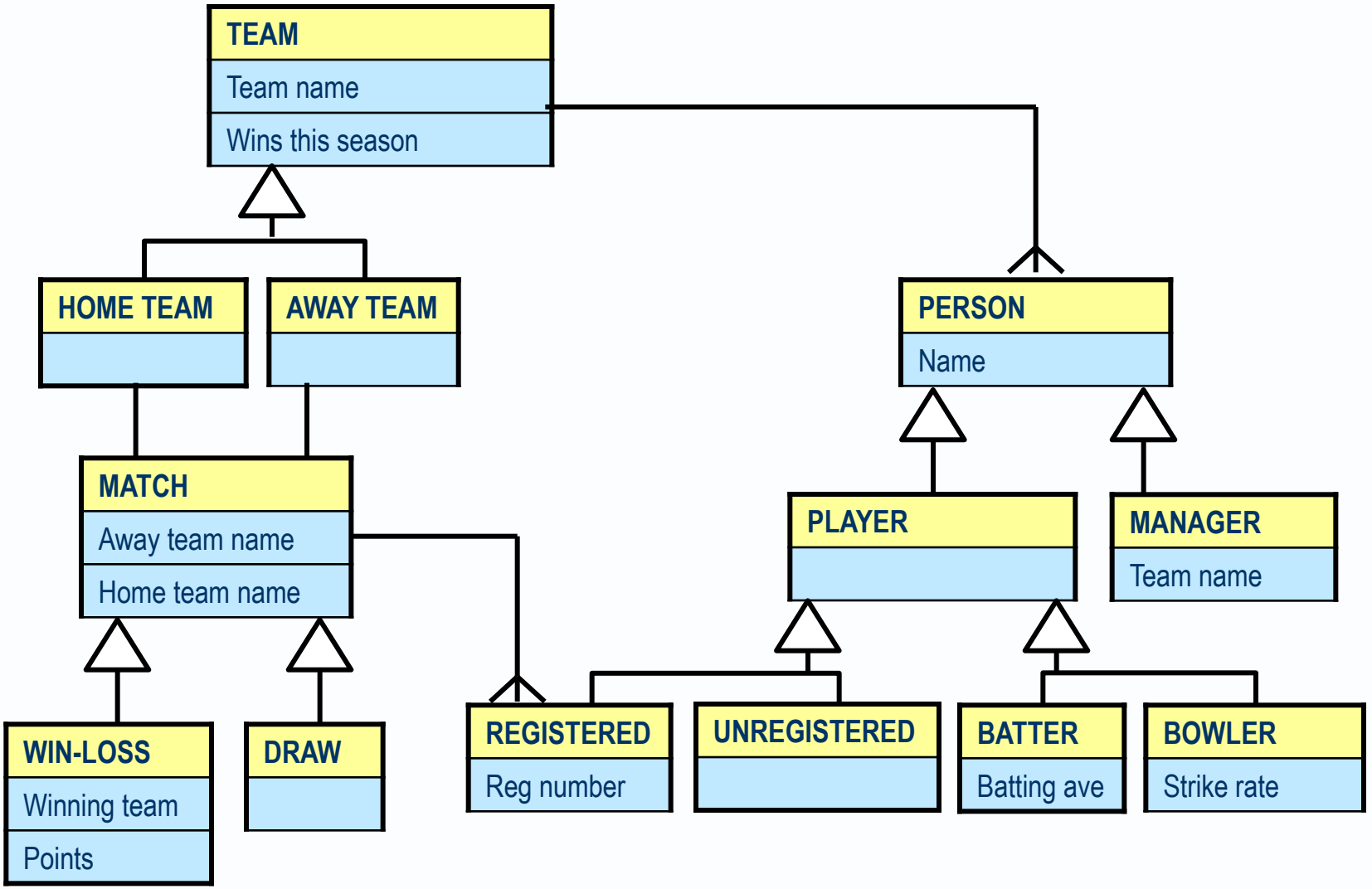
- Class hierarchies can get out of hand, become too complex to understand, as in the OMG repository design!
- More importantly, they are often misconceived
- OO evangelism can lead people to create vacuous generalisations, and pointless class hierarchies.



Copyright Avancier Ltd 2015

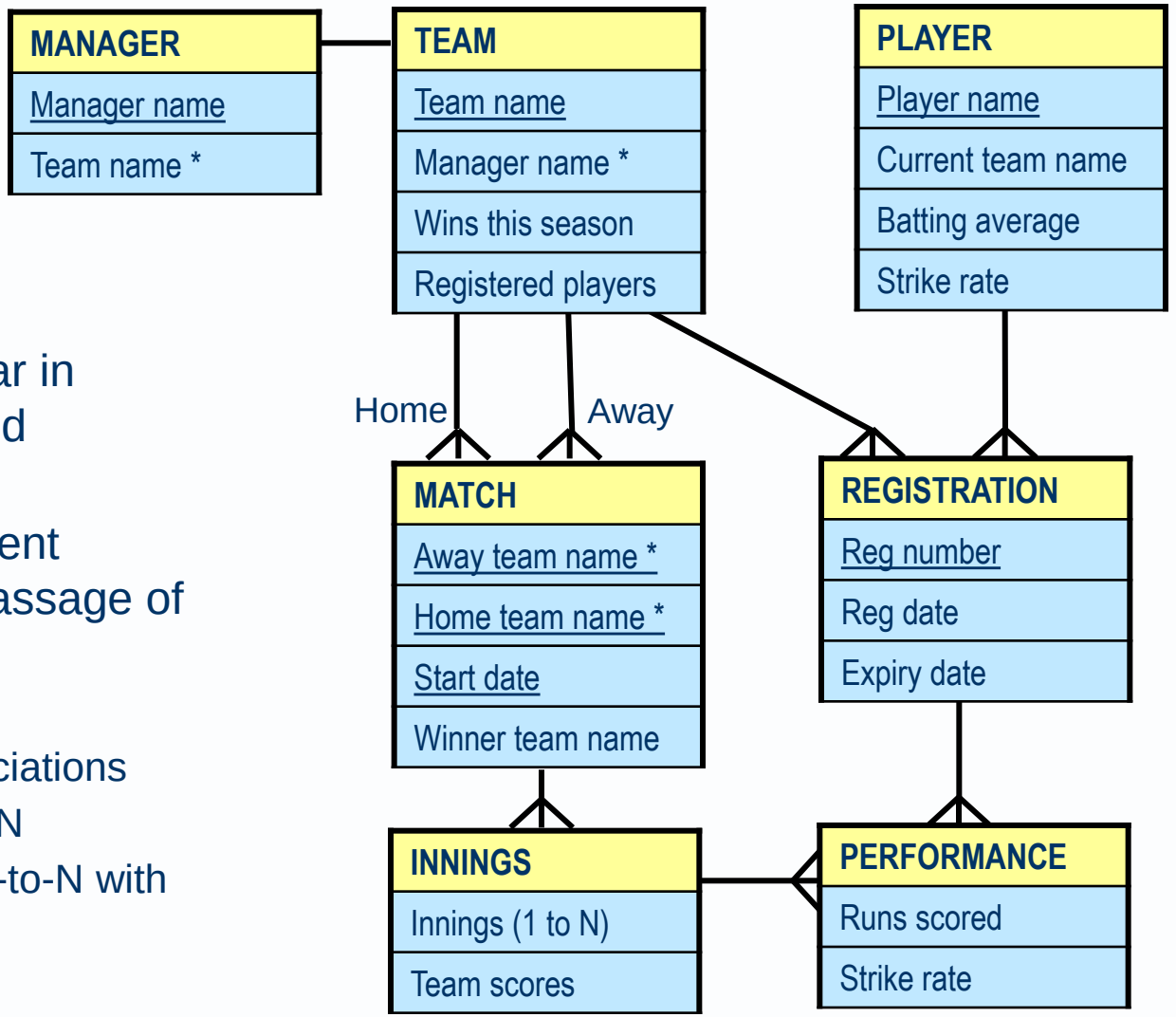
It is all to easy invent class hierarchies

- E.g.



Copyright Avancier Ltd 2015

A more useful domain model (sketch only)

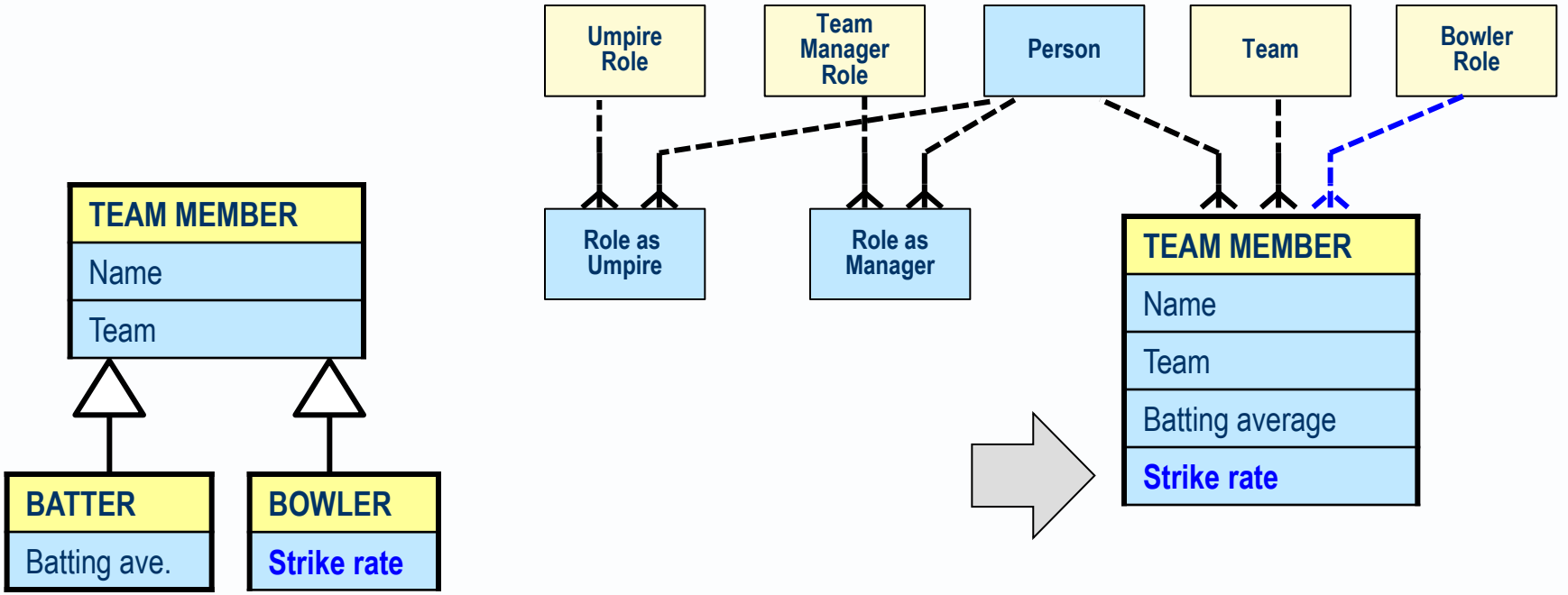


- Class hierarchies appear in models of temporary and artificial things
- But in models of persistent business entities, the passage of time tends to turn
  - Subtypes into roles
  - Aggregations into associations
  - 1-1 associations into 1-N
  - 1-N associations into N-to-N with link entities

Copyright Avancier Ltd 2015

Null attributes and optional associations

- Every player bats, but only some bowl, so the strike rate can be a null attribute, an optional association to the bowler role



Copyright Avancier Ltd 2015

The end

- Requirements
- Analysis of system input/output
- Data types
- Modelling system state and behaviour
- Aggregate entities
- Analysing association cardinality
- Analysing class hierarchies

Copyright Avancier Ltd 2015