Business-Oriented Data Modelling Masterclass –
Balancing Engagement, Agility, and Complexity

Presented for ING Bank by
Adept Events and Clariteq Systems Consulting

Alec Sharp

Consultant
Clariteq Systems Consulting Ltd.
West Vancouver, BC, Canada
asharp@clariteq.com
www.clariteq.com

ING BANK   ///AdeptEvents   CLARITEQ
                  www.adeptevents.nl   www.clariteq.com

1

Instructor / course developer background…

Alec Sharp, Clariteq Systems Consulting – *asharp@clariteq.com*

- 40+ years experience as an independent consultant:
  - Business Process Change – discover, model, analyse, and design/redesign processes
  - Application Requirements Specification
  - Data Modelling and Management   My roots!
    +
  - Facilitation & Organisational Change
  - Project Recovery
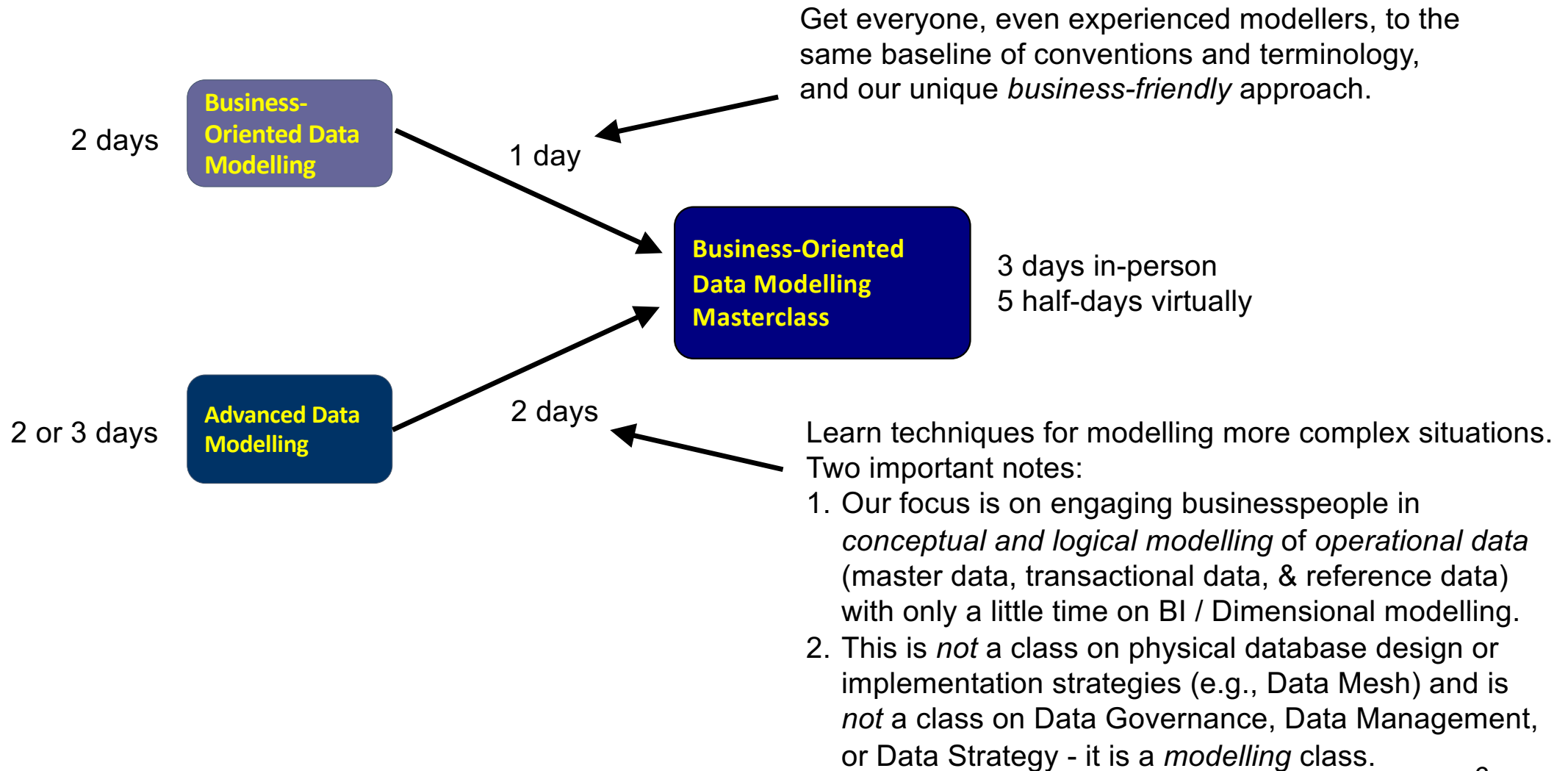
- Consulting, teaching, speaking globally (pre-pandemic)

- Awarded DAMA's global Professional Achievement Award for contributions to "human-friendly" data modelling

- Author of "Workflow Modeling"
  - best-selling book on process modelling & improvement
  - second edition – a complete re-write

Process — Business Process Modelling

Application — Use Case Modelling / Service Specification

Data — Concept Modelling

Check out the nice reviews on Amazon - http://amzn.to/dHun1o

WORKFLOW MODELING

2
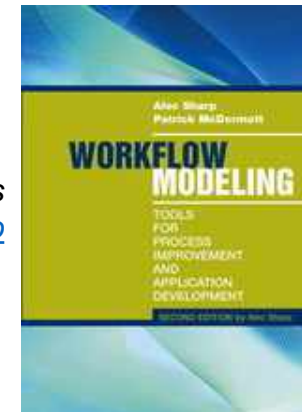
Background for this course

Business-Oriented Data Modelling

2 days

1 day

Get everyone, even experienced modellers, to the same baseline of conventions and terminology, and our unique *business-friendly* approach.

Business-Oriented Data Modelling Masterclass

3 days in-person
5 half-days virtually

Advanced Data Modelling

2 or 3 days

2 days

Learn techniques for modelling more complex situations. Two important notes:
1. Our focus is on engaging businesspeople in *conceptual and logical modelling* of *operational data* (master data, transactional data, & reference data) with only a little time on BI / Dimensional modelling.
2. This is *not* a class on physical database design or implementation strategies (e.g., Data Mesh) and is *not* a class on Data Governance, Data Management, or Data Strategy - it is a *modelling* class.

3

Overview and logistics

Introduction / Level-set

1. Essentials of concept modelling <mark>(longer)</mark> (& relation to other BA techniques)
2. Adding rigor, structure, & detail <mark>(shorter)</mark> (Conceptual to Logical)

Advanced Topics

1. Interesting structures
2. Modelling time & history
3. Rules on relationships and associations
4. Presentation techniques for data modellers
5. Relating Dimensional and Entity-Relationship models

You:
- Name (how should I address you?)
- Role / job title, organisation, and location
- Brief description of your work
- Is there a topic you are especially interested in?
- *Please try to keep your introduction to one minute or less*

Case study – Concept Model, Services, Use Cases, Business Processes

Client –

- Regulatory agency ensuring the safe design, installation, and use of technical equipment

- Natural gas systems, electrical systems, boilers and pressure vessels, elevating devices, & many more

Goal –

- Shift from an inspection-based model (~800 inspectors!) to client-managed safety programs

- Clients will apply for a *Client Safety Management Program Authorisation (CSMP Authorisation)*
  - must show effective processes and accurate record-keeping

- Clients will pay a fee for managing *their own safety programs!* Still beneficial!

5

Case study – Concept Model, Services, Use Cases

- Business Development chooses Pilot Program –
  boilers and pressure vessels in Oil & Gas fields

- Current systems won't support CSMP, time-consuming and expensive to change them –
  IT and Finance suggest 18 – 24 months of work

- BD is unimpressed by IT and Finance objections ("You're being mindlessly obstructionist!")
  and proposes work-around procedure. *Guess which tool they intend to use?*

- I'm hired to identify end-to-end implications –
  "Design a process and determine IT requirements that will allow this procedure to work."

- *Concept Modelling was a critical tool in understanding the underlying policies,
  and developing the process & requirements*

6

Always start with terminology (the "things")

From one-on-one interviews with 8-10 key stakeholders we gathered ~200 terms related to CSMP (Client Safety Management Program) – "anything that went by a name." Here are 24 that met the criteria to be a "thing"– the candidate *Entities.*

Device | Client | Unit | Location | Company | Site

Applicant | Pressure Vessel | Operator | Owner | Boiler | Licensee

Slug | Operation | Verification | Customer | Plant | Inspection

Pig | Facility | Permission | Authorisation | License | Confirmation

Identify synonyms and select one term.
How do these relate to one another?
What do you need to know about each?

7

Review of a Miro example – Terminology Analysis

Terminology analysis (continued):

Let's arrange these terms into columns of synonyms. It's always a surprise for the business to see how many terms are used to describe the same fundamental thing!
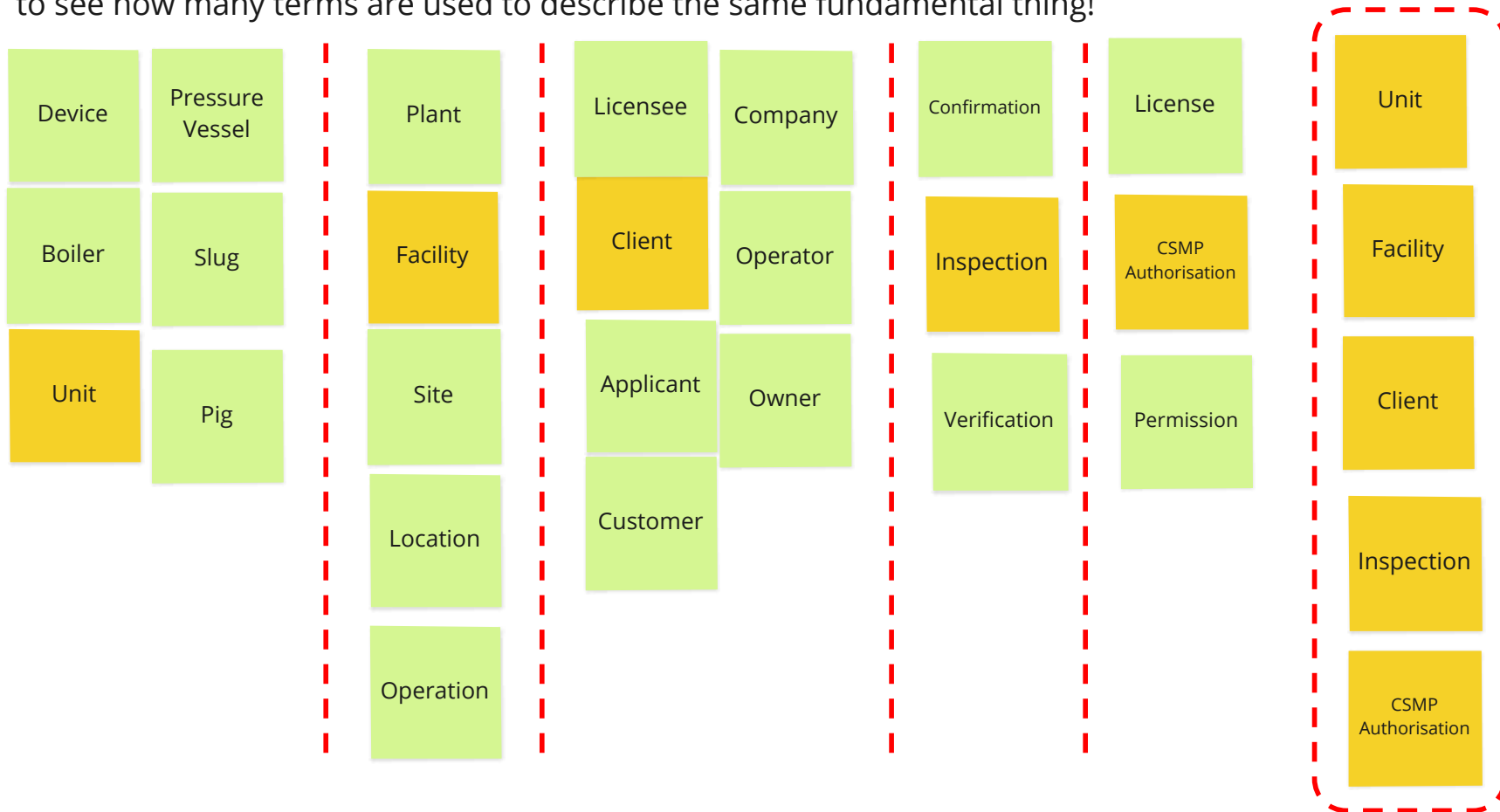
Concept Model Version 1; not perfect, but a good start

1. We arranged the entities / business objects by dependency
2. Then we drew relationship lines
3. Then we added a relationship name in each direction
4. Only then did we state (in words) the cardinality (1:1, 1:M, M:M)
   and then update the diagram with hash marks (†) and crowsfeet (⅄)
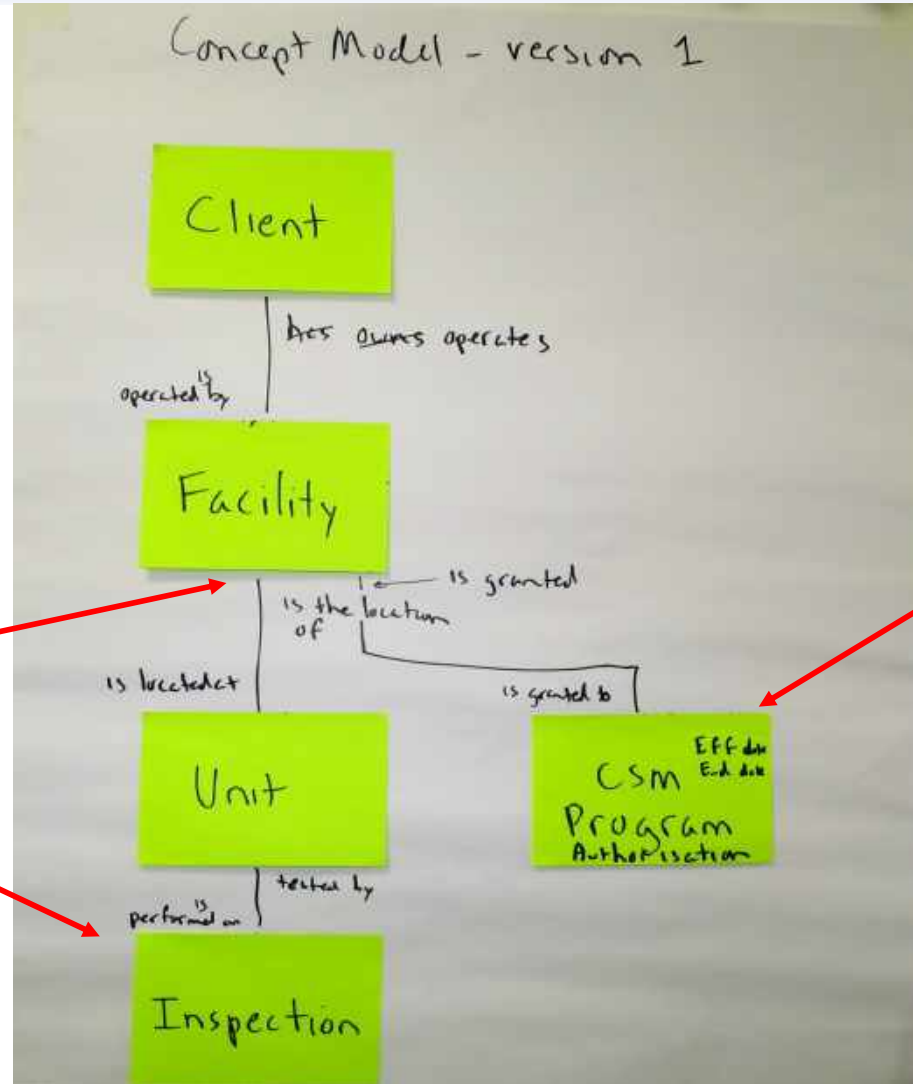
Definition -

A CSMP Authorisation is a permission (or license) to operate a self-managed safety program (a Client Safety Management Program) at a specific Facility, for a specified time period, usually 1, 2, or 5 years.

The CSMP Authorisation is "all or nothing" - it covers ALL the Units at a Facility.

9
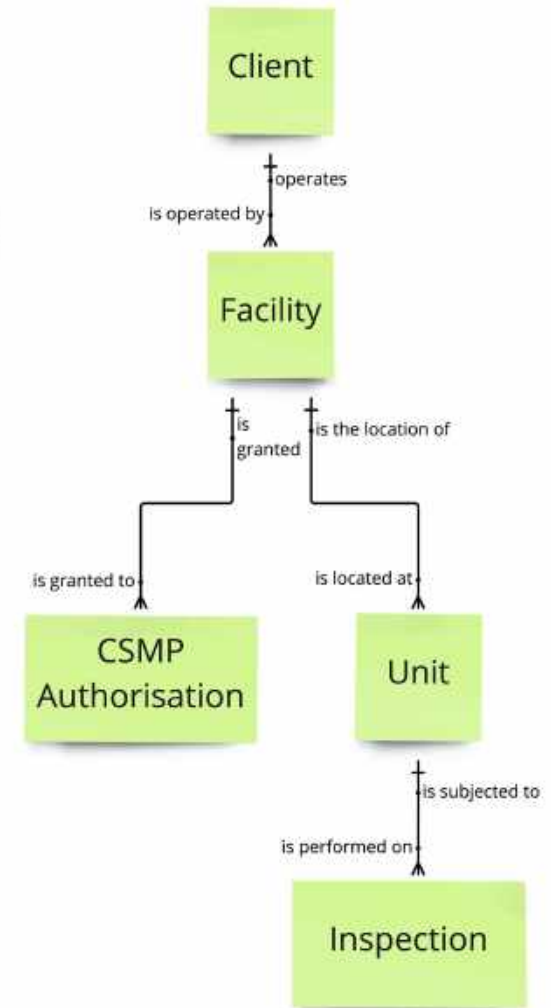
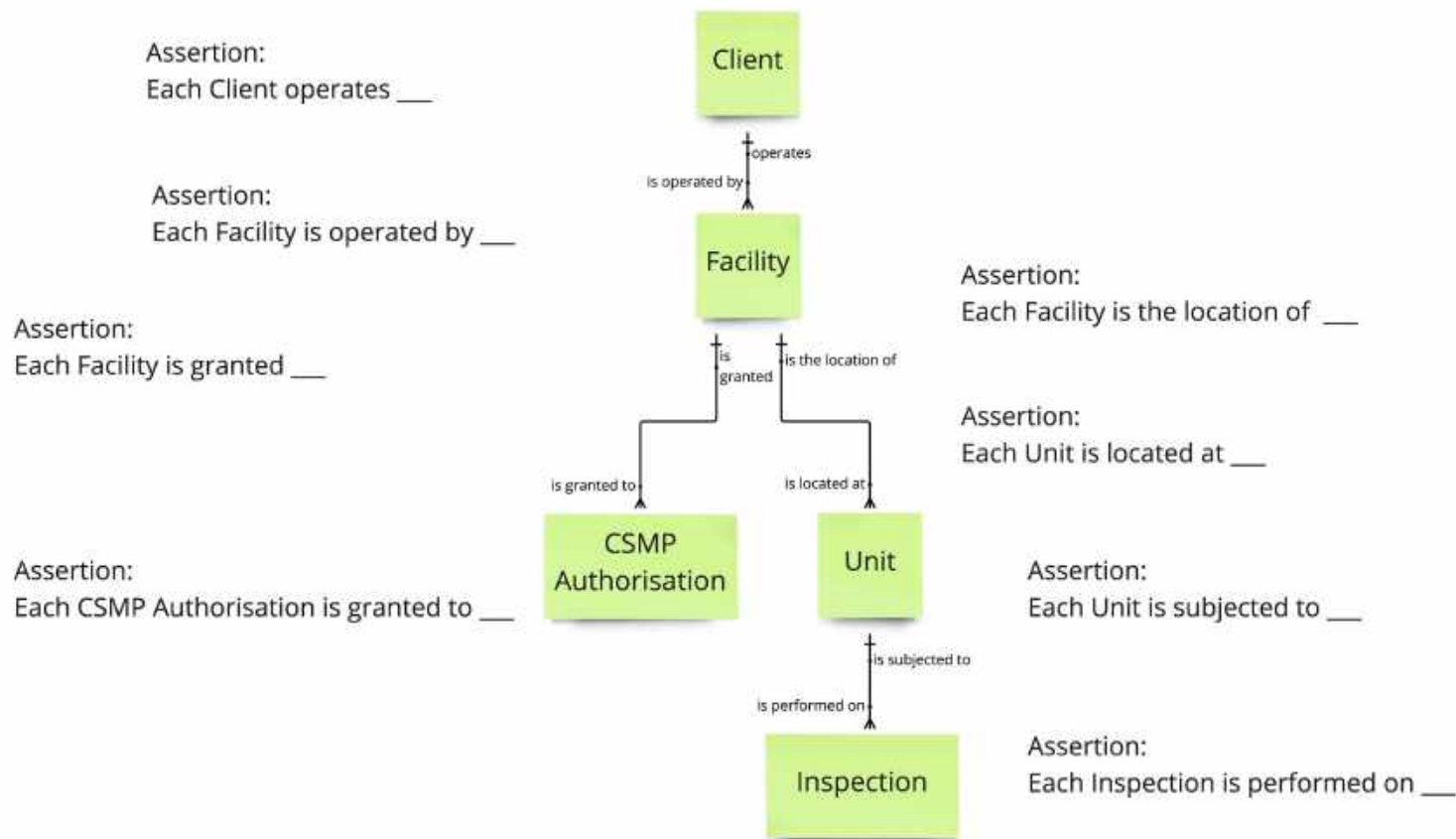# *Just boxes and lines, but raises important questions*

Are Units permanently part of one Facility?

What do we Inspect?

What do we issue the Authorisation to?

Concept Model Version 1; state Assertions and challenge them

Now, state the relationships **emphatically** as Assertions. **Each** Client operates **one or more** Facilities! Then, **challenge** them!
Again, don't worry yet about **optionality** – whether the relationship **must be** or **may be** be present.
We only care now about the **maximum** – each ObjectA is related to a **maximum** of **one** or **one or more (or many)** ObjectB.

Assertion:
Each Client operates ___

Assertion:
Each Facility is operated by ___

Assertion:
Each Facility is granted ___

Assertion:
Each CSMP Authorisation is granted to ___

Client

operates
is operated by

Facility

is
granted

is the location of

is granted to

is located at

CSMP
Authorisation

Unit

is subjected to

is performed on

Inspection

Assertion:
Each Facility is the location of ___

Assertion:
Each Unit is located at ___

Assertion:
Each Unit is subjected to ___

Assertion:
Each Inspection is performed on ___

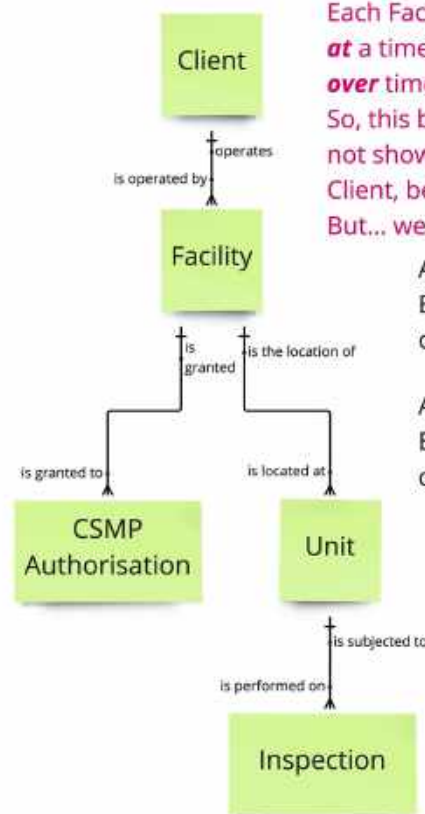# *Concept Model Version 1; revised Assertions from challenges*

Now, state the relationships **emphatically** as Assertions. **Each** Client operates **one or more** Facilities! Then, **challenge** them!
Again, don't worry yet about **optionality** – whether the relationship **must be** or **may be** be present.
We only care now about the **maximum** – each ObjectA is related to a **maximum** of **one** or **one or more (or many)** ObjectB.

Assertion:
Each Client operates
one or more Facilities

Assertion:
Each Facility is operated by
one Client

Assertion:
Each Facility is granted
one or more CSMP Authorisations

One CSMP Authorisation **at** a time,
but one or more **over** time

Assertion:
Each CSMP Authorisation is granted to
one Facility

Client

operates
is operated by

Facility

is
granted
is the location of

is granted to
is located at

**CSMP
Authorisation**   **Unit**

is subjected to

is performed on

Inspection

Each Facility is operated by one or more Clients
at a time (Joint Ventures) and
over time (changes in Ownership or Lease.)
So, this becomes a M:M relationship, and we should
not show a Facility as being dependent on a single
Client, because a Facility is an independent thing.
But... we don't always get our way!

Assertion:
Each Facility is the location of
one or more Units

YES, but one or more Facilities **over** time, because
Units can move between Facilities. So, this
becomes a M:M relationship, and we cannot show
a Unit as being dependent on a single Facility,
because a Unit is an independent thing

Assertion:
Each Unit is located at
one Facility

Assertion:
Each Unit is subjected to
one or more Inspections

Assertion:
Each Inspection is performed on
one Unit

BODM-MC:
Business-
Oriented
Data Modelling
Masterclass

Concept Model Version 2; revised from challenging Assertions

Now we will re-draw the initial Concept Model based on changes that came from challenging the Assertions in Ver. 1.
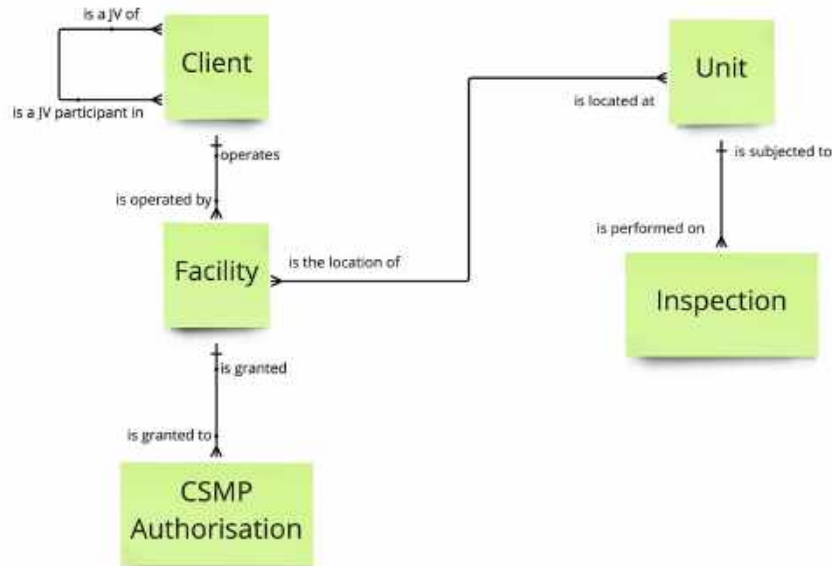
Note:

You don't always get what you *want* or what you think is the *right* thing in Concept Modelling. In this case the client (the Regulator) said they always wanted a Facility to be operated by ONE AND ONLY ONE Client.
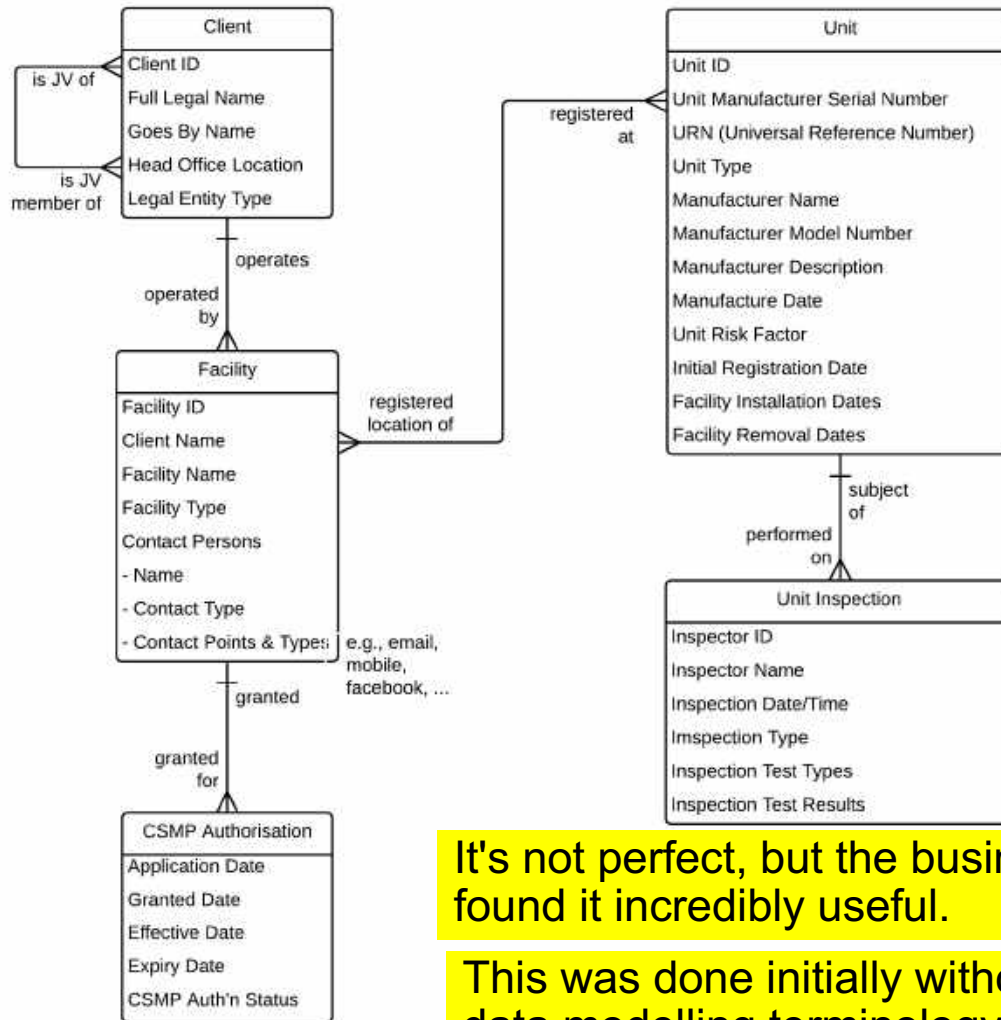
If a Facility was operated by multiple Clients, they would require the Clients to form a new Joint Venture Client. This was to ensure that if there were legal difficulties, there was only ONE Client to go after.

Or, as they put it, "one throat to choke."

Later in the project, they realised they needed a history of the Clients that had operated a Facility, so the Client-Facility relationship became Many-to-Many, and Facility was modelled (correctly) as an independent Entity, as shown here:

"What do you need to know about the things in the Concept Model?"

Client
- is JV of
- is JV member of
- Client ID
- Full Legal Name
- Goes By Name
- Head Office Location
- Legal Entity Type

operates
operated by

Facility
- Facility ID
- Client Name
- Facility Name
- Facility Type
- Contact Persons
  - Name
  - Contact Type
  - Contact Points & Types
  - e.g., email, mobile, facebook, ...

registered location of

granted
granted for

Unit
- Unit ID
- Unit Manufacturer Serial Number
- URN (Universal Reference Number)
- Unit Type
- Manufacturer Name
- Manufacturer Model Number
- Manufacturer Description
- Manufacture Date
- Unit Risk Factor
- Initial Registration Date
- Facility Installation Dates
- Facility Removal Dates

registered at

subject of
performed on

Unit Inspection
- Inspector ID
- Inspector Name
- Inspection Date/Time
- Imspection Type
- Inspection Test Types
- Inspection Test Results

CSMP Authorisation
- Application Date
- Granted Date
- Effective Date
- Expiry Date
- CSMP Auth'n Status

Sketching this out was *fast, and* raised many questions that had not occurred to the client…

- Is there one CSMP per Client, per Facility, or some other basis?

- Do Units frequently relocate, or even turn up at another Client?

- What is inspected – the Facility or the Unit?

- Does the CSMP cover all or some Units at a Facility?

- …and MANY more…

It's not perfect, but the businesspeople found it incredibly useful.

This was done initially without any data modelling terminology or symbols!

Model took ~90 minutes

14

Summary – what an analyst can do with a Concept Model

First, clarify language. (A platform)

Second, establish policies and rules.

And then, identify events or services, e.g.,
A **Unit** is…
- Registered   (requiring the service "Register Unit")
- Loaded   (requiring the service "Load Unit")
- Idled   (requiring the service "Idle Unit")
- Reactivated   (requiring…)
- Repaired
- Inspected
- Relocated
- Retired
- …

We did the same for Client, Facility, CSM Program, …

These are the essential capabilities

Something I always do when evaluating/selecting COTS S/W

Identify Services (Events) then Use Cases / User Stories

Finally, we'll identify the Services (verb - noun pairs) we need, and the Use Cases / User Stories by which the Services will be accessed

What events happen to a Unit - what are the needed services? (Verb - Noun)
- ...
- ...
- ...
-

Client
CSR
via Portal
via S-MAN
Decommission
~~Install~~
Register
~~Load~~
Reactivate
~~Operate~~
Load
Unit (a Business Object / Entity)
Idle
Repair
Service
Inspect
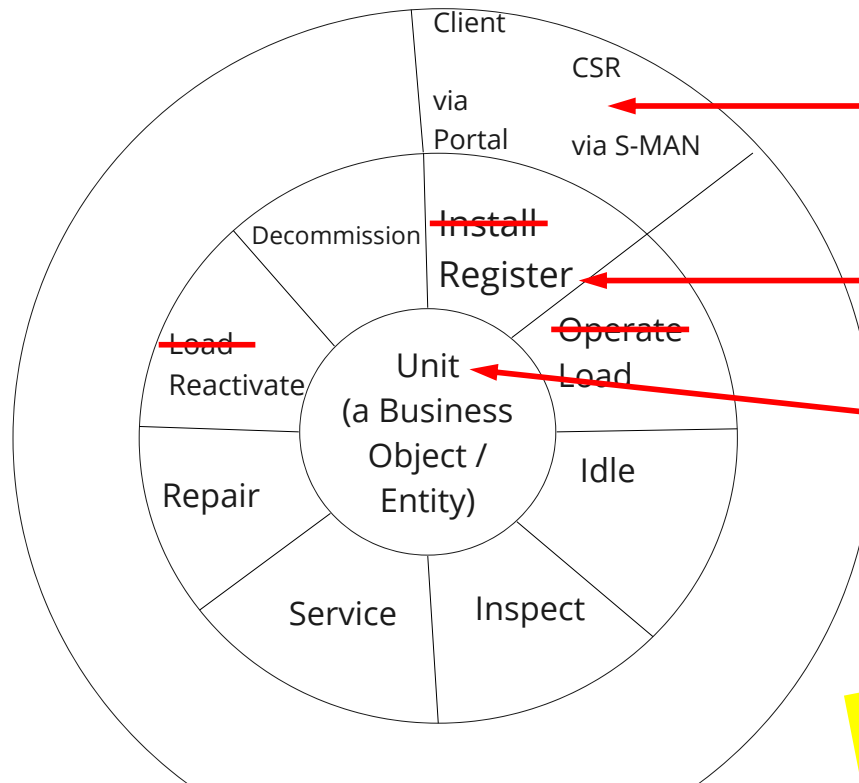
Who needs access to each Service, and How?

Use Case

Use Case or User Story - add Who and How

Service Specification (Events)

Service (or Event) - add a Verb to the Noun

Concept Model

Entity or simply a "thing" - a core Noun

A Concept Model is a great starting point for discovering your Services and Use Cases (User Stories)

Supports *Service-Oriented Business Analysis*

Note – "User Story" and "Use Case" are not so different

Different format and detail, but the same basic concept.
Initially, at the Scope level, they're much the same:

User Story (who – what – *why*):
  "As a Client, I need the ability to Register Unit(s,)
  so I can maintain compliance with my CSMP Authorisation"

Use Case: (who – what – *how*):
  "Client Register Unit via Portal"

When we add detail at the Concept level, they become identical:
• User Story / Use Case abstract
• Main success sequence – dialogue in "when-then" format
• Alternate sequences – variations, exceptions, errors

Develop high-level use cases and services

Service: *Register Unit*

- Check for presence of properly formatted UR Number

- Determine if Unit UR Number is previously known

- If known, has it (a) moved (b) changed ownership (c) …?

Use Case: *CSR Registers Unit via S-MAN*

- CSR will select "spreadsheet" of all Units covered by CSMP app

- S-MAN will highlight all that can proceed immediately

- For each category of Units requiring intervention…

Note:
> Services and Use Cases at the "upper conceptual" level to provide vendor with key elements of requirements and avoid the usual bulleted list requirements document.

Discussion – one Business Service, one or more Use Cases

One Service

Multiple Use Cases

Who	What (the Service – verb + noun)	How
Client	Register Unit	via Portal
Customer Service Rep (CSR)	Register Unit	via S-MAN (the ERP)
Client	Register Unit	via Mobile App
???	Register Unit	???
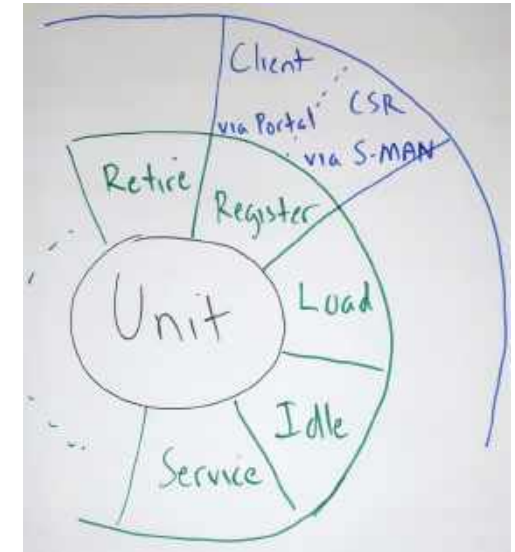
What is the value of documenting the Service only *once?*
("One Service available through multiple channels.")

- re-use of the asset, and therefore higher consistency
- better chance of getting it right – higher value from less effort
- if it's implemented as a single service, easier maintenance – it's in ONE place.

Why would we make a *single* Service available via *multiple* Use Cases?

- different actors need different "navigation and hand-holding,"
  e.g., casual vs. expert users
- different technology platforms have different capabilities,
  e.g., mobile phone vs. touch-screen kiosk

19

Clarify scope of the new process and identify participants

Trigger:
Client submits request to enter into a CSMP

Client Result:
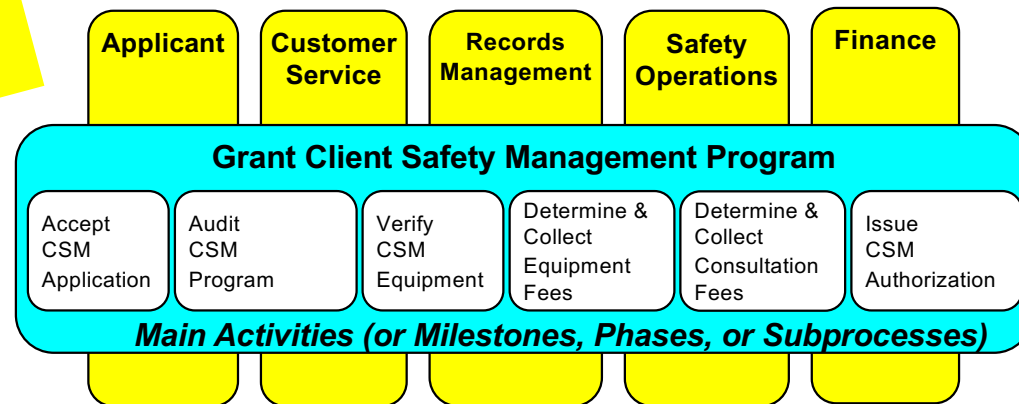*Approval granted* for a self-managed safety program.

Grant Client Safety Management Program

| Accept CSM Application | Audit CSM Program | Verify CSM Equipment | Determine & Collect Equipment Fees | Determine & Collect Consultation Fees | Issue CSM Authorisation |

Main Activities (or Milestones, Phases, or Subprocesses)

Agency Result:
Revenue collected. New participant in CSMP; confirmation that regulations are satisfied

Cases:
- New
- Grandfathered
- Ownership Change

Process Scope Model – pure "what"…

We also did an Augmented Scope Model – an additional level of detail for Activities

| Applicant | Customer Service | Records Management | Safety Operations | Finance |

Grant Client Safety Management Program

| Accept CSM Application | Audit CSM Program | Verify CSM Equipment | Determine & Collect Equipment Fees | Determine & Collect Consultation Fees | Issue CSM Authorization |

Main Activities (or Milestones, Phases, or Subprocesses)

Process Summary Chart – simplified "what," plus "who"

20

The initial, business-friendly workflow model

Process: Grant CSM Program Authorization, Case: "grandfathered" safety program (page 1 only)
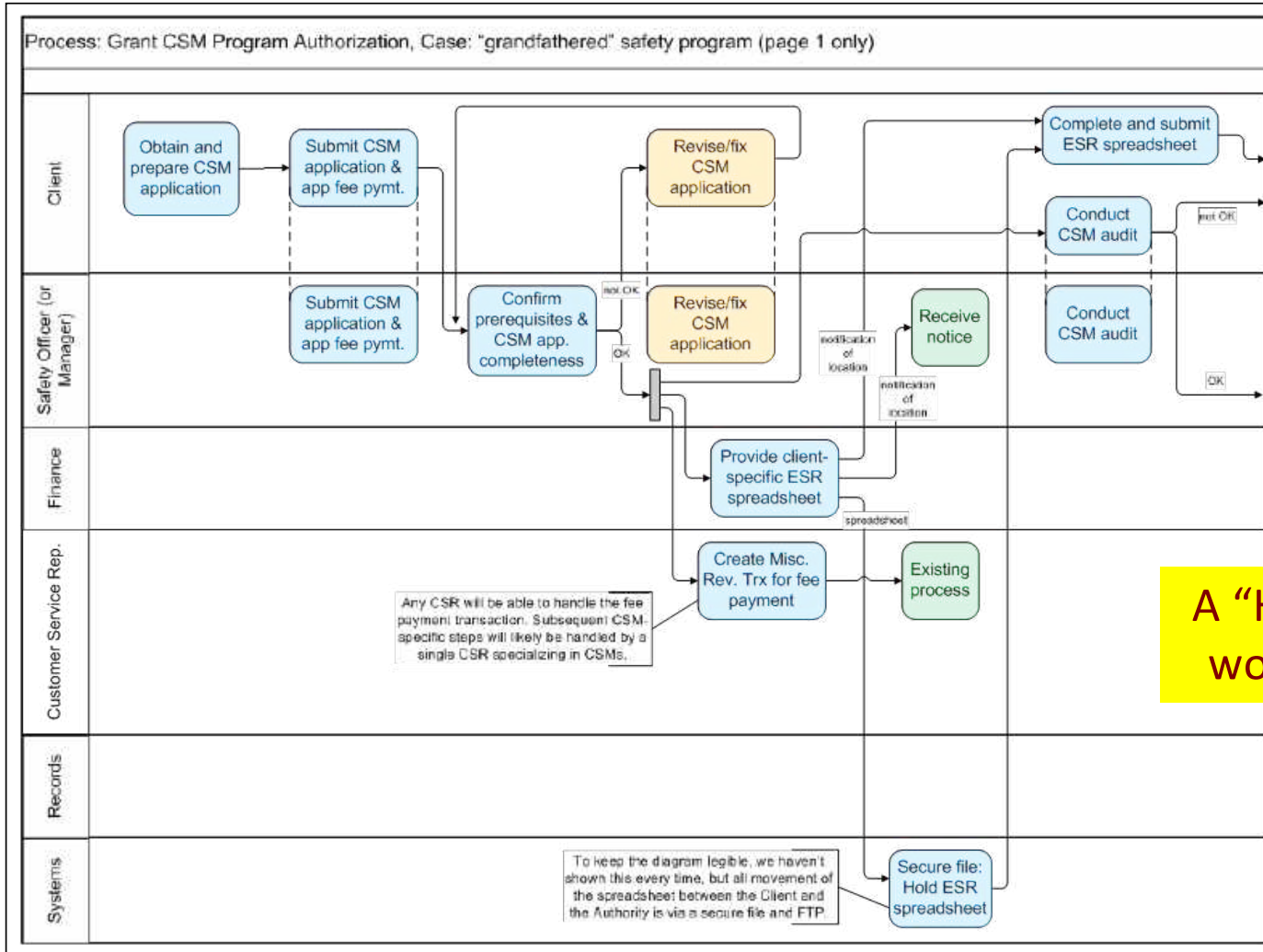
A "Handoff Level" workflow model

Then detail showing where use cases & services fit

Process: Grant CSM Program Authorization, Case: "grandfathered" safety program (2nd to last page only)

3) ...by interacting with

1) This Actor (or Role)...

2) ...completes this Activity...
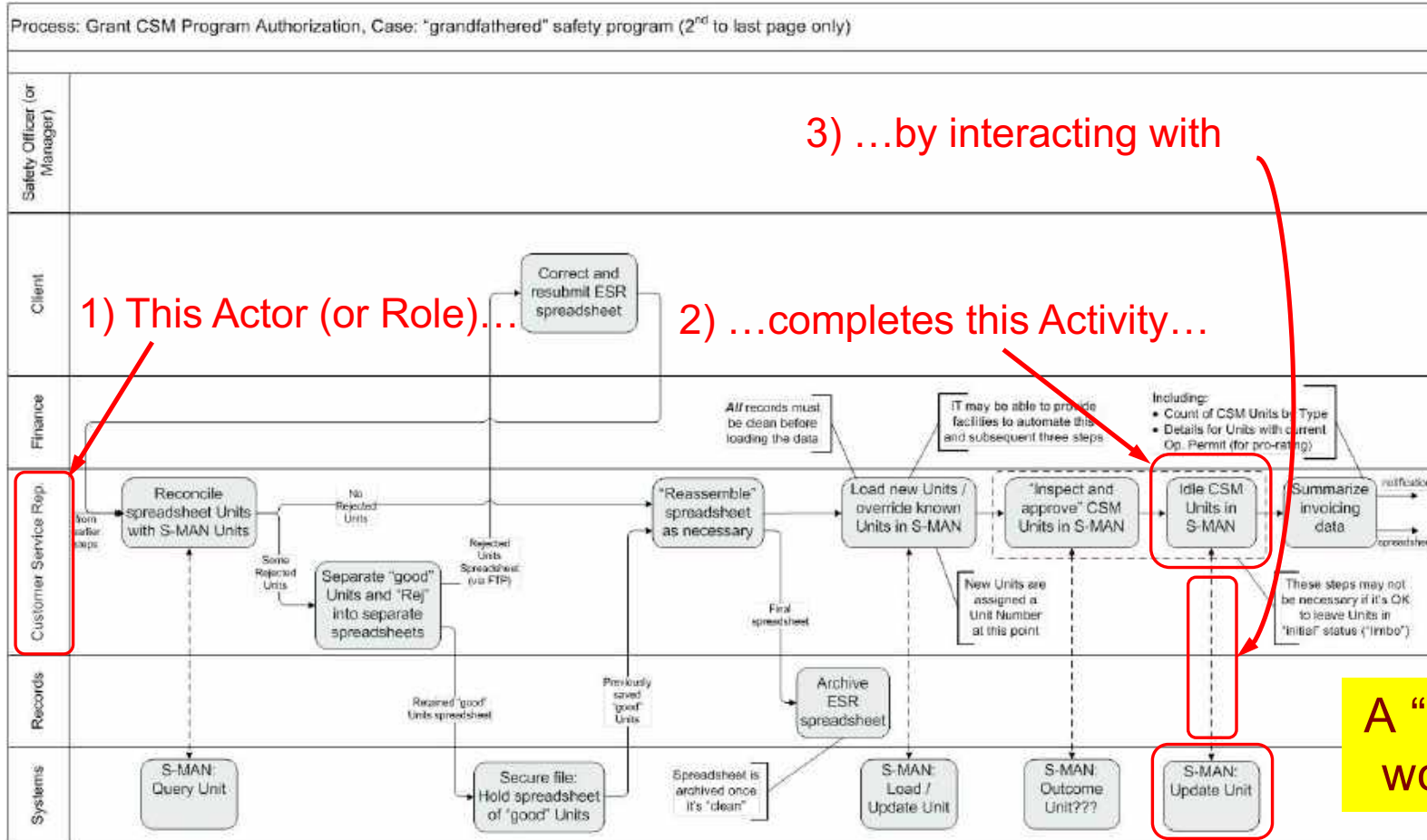
A "Service Level" workflow model

4) ... this Service offered by a System
(which collectively is a Use Case)

22
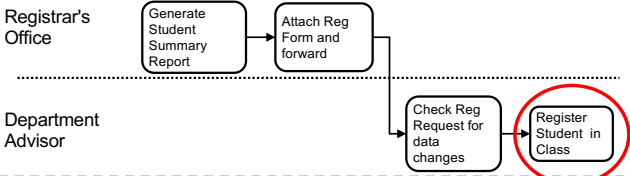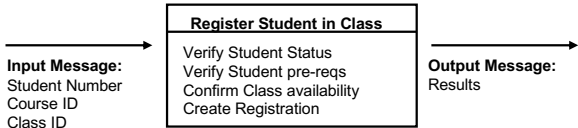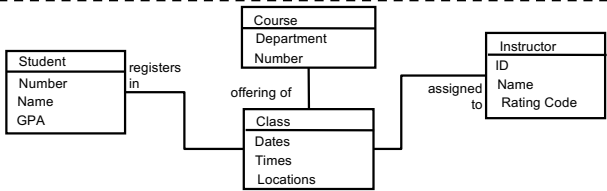
Mission accomplished! Conclusions:

- "Plan A" rejected – agreement that Unit data *must* get into S-MAN
- "Plan B" (change the app) looks good, but the vendor estimates are *HIGH*
- "Plan B Minus" (existing functionality plus CSR work) is *worth the cost*
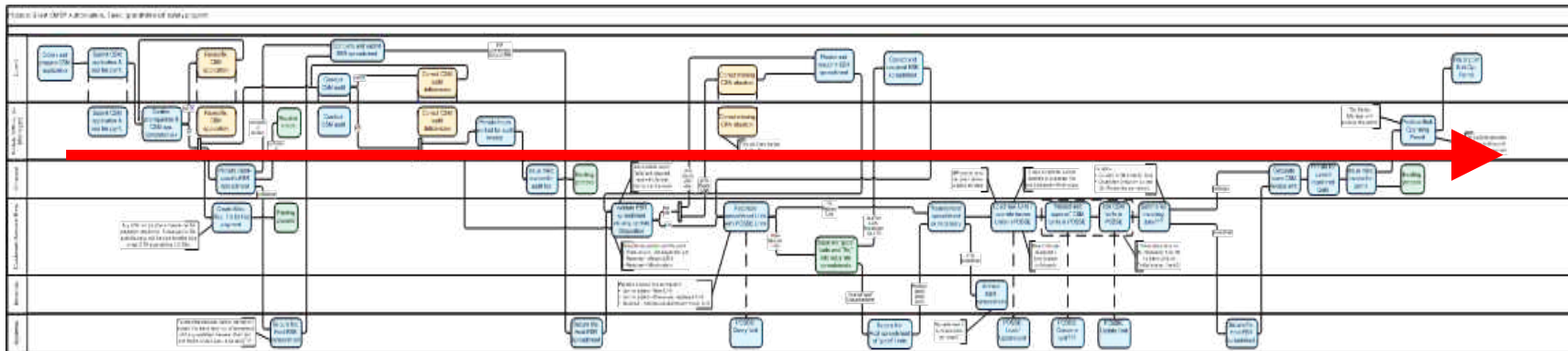
1. If requirements, issues, assumptions, etc. are in lists, people will argue endlessly; if they are in an *integrated* and *understandable* set of models, it's much harder to dismiss the reality of the situation

2. Process Models, Use Cases, Service Specs, & *Concept Models*: **essential!**

23

Our framework for Business Analysis

Framework Layer	**Technique sample**	**What it covers**

Goals

Business Objectives

The university is initiating the "Strategic Enrollment" program to raise Student graduation rates in part by ensuring Classes are available for Student registration when needed.

✓ *Project Charter* – documents the rationale, objectives, scope, and success measures for the project

This is not a sequence!

Process

Business Process

Registrar's Office — Generate Student Summary Report → Attach Reg Form and forward

Department Advisor — Check Reg Request for data changes → Register Student in Class

✓ *Process Model* - shows "what" in a Scope Model, then "who & how" in a Workflow Model – the steps done by the actors in the process

Business Process: gives great context for Business Analysis

Application

Presentation Services (user interface)

When advisor enters five characters of Last Name → Then System lists matching Students

When advisor selects list item → Then System displays expanded Student view with needed Classes

When advisor etc.

✓ *Use Case* – models how an actor interacts with a system to obtain (trigger) a service, typically to complete a step in a process

Use Cases and Services: where we capture Functional Requirements

Business Services (rules & logic)

Register Student in Class
Verify Student Status
Verify Student pre-reqs
Confirm Class availability
Create Registration

Input Message:
Student Number
Course ID
Class ID

Output Message:
Results

✓ *Service Specification* - describes a service – a package of rules and logic – that is triggered to complete or respond to a business event

---

Data

Data Mgmt. Services (databases)

Student
Number
Name
GPA
registers in

Course
Department
Number
offering of

Class
Dates
Times
Locations

Instructor
ID
Name
Rating Code
assigned to

✓ *Concept Model* - depicts the things and the facts about things the organisation needs to record; the things (the entities) are what processes and solutions act on.

Concept Model / Data Model: a great platform for Business Analysis
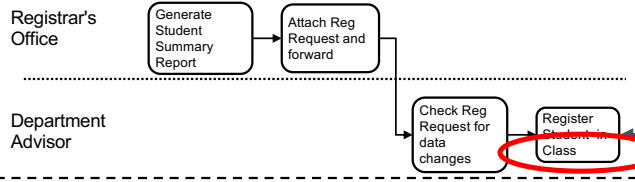
Key point! Everything relies on the Concept Model

Goals

Business Objectives

The university is initiating the "Strategic Enrollment" program to raise Student graduation rates in part by ensuring Classes are available for Student registration when needed.
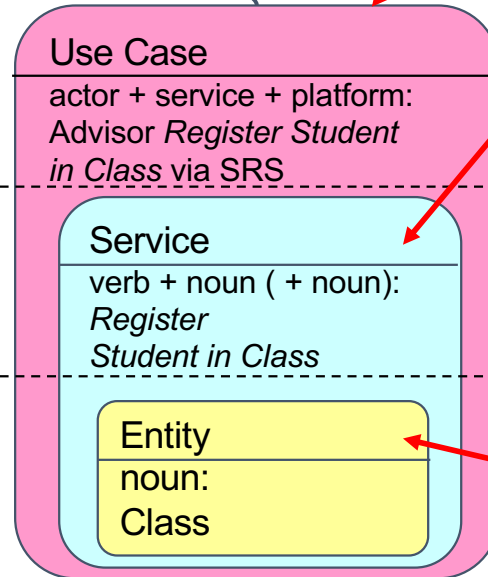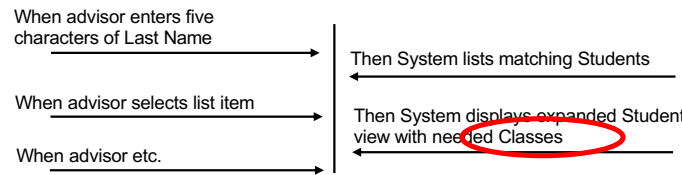
All use the language and constraints of the Concept Model (the "thing model") – the ultimate "what"
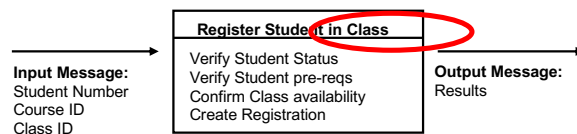
Process

Business Process

Registrar's Office

Generate Student Summary Report → Attach Reg Request and forward

Department Advisor

Check Reg Request for data changes → Register Student in Class

Use Cases/User Stories:
- Who (Actors) needs access to the Services, and how (Platform)?

Application

Presentation Services (user interface)

When advisor enters five characters of Last Name → Then System lists matching Students

When advisor selects list item → Then System displays expanded Student view with needed Classes

When advisor etc.

Business Services (rules & logic)

Register Student in Class
Verify Student Status
Verify Student pre-reqs
Confirm Class availability
Create Registration

Input Message: Student Number, Course ID, Class ID

Output Message: Results

Use Case

actor + service + platform:
Advisor *Register Student in Class* via SRS

Service

verb + noun (+ noun):
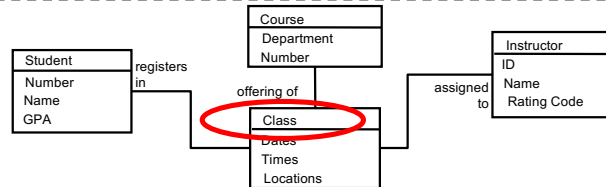*Register Student in Class*

Entity

noun:
Class

Verb-Noun pairs:
- The *Services* (event-handlers) that are at the heart of a *Service Oriented Architecture.*
- *Also* "building blocks" of Business Processes

Data

Data Mgmt. Services (databases)

Student: Number, Name, GPA — registers in — Course: Department Number — offering of — Class: Dates, Times, Locations — assigned to — Instructor: ID, Name, Rating Code
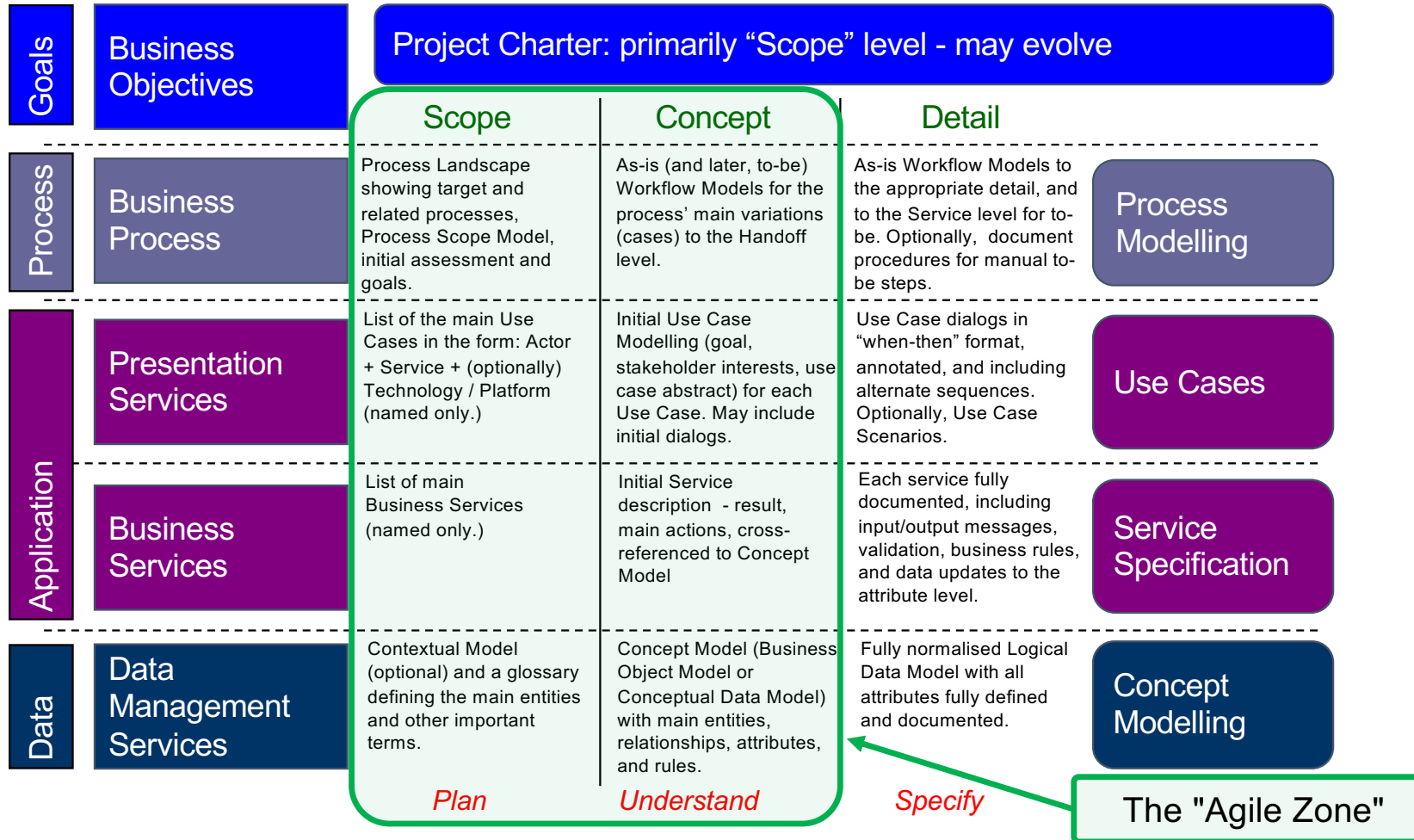
The core *Nouns or Things* in your enterprise. Also known as *Business Objects.*

Bonus – great starting point to discover your Events/Services and Use Cases/User Stories
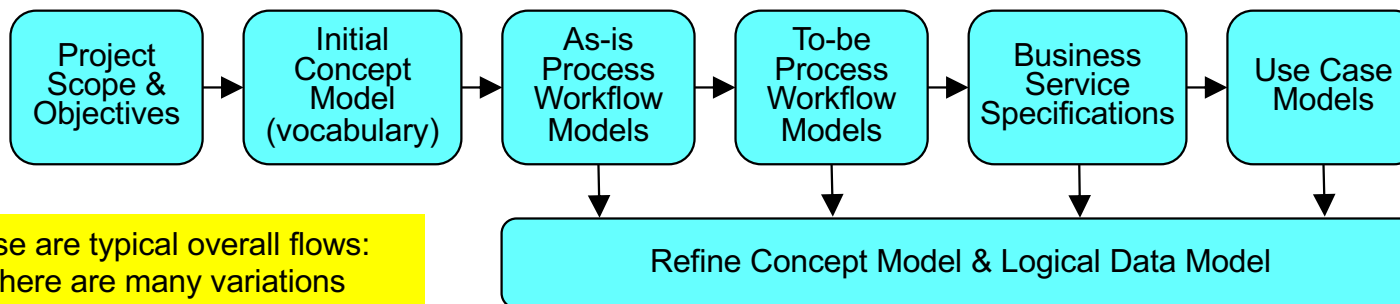
25

Progressive detail and Agile

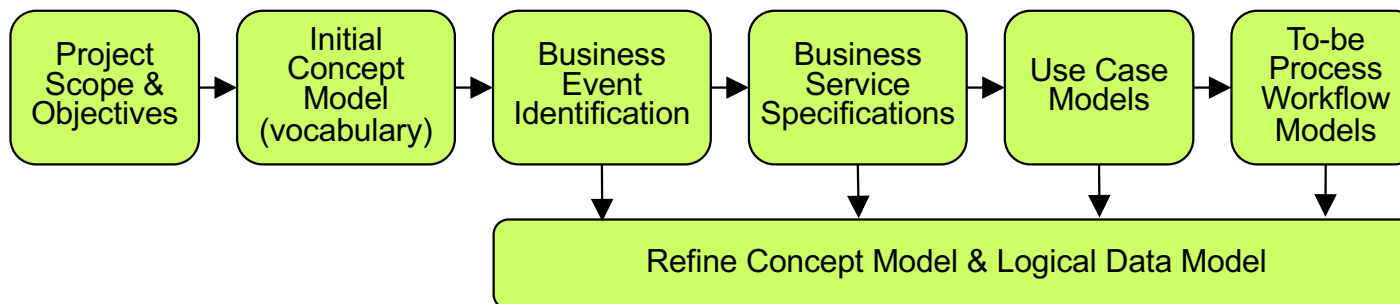## *Clariteq framework for analysis and architecture*

		Scope	Concept	Detail	
Goals	**Business Objectives**	Project Charter: primarily "Scope" level - may evolve			
Process	**Business Process**	Process Landscape showing target and related processes, Process Scope Model, initial assessment and goals.	As-is (and later, to-be) Workflow Models for the process' main variations (cases) to the Handoff level.	As-is Workflow Models to the appropriate detail, and to the Service level for to-be. Optionally, document procedures for manual to-be steps.	**Process Modelling**
Application	**Presentation Services**	List of the main Use Cases in the form: Actor + Service + (optionally) Technology / Platform (named only.)	Initial Use Case Modelling (goal, stakeholder interests, use case abstract) for each Use Case. May include initial dialogs.	Use Case dialogs in "when-then" format, annotated, and including alternate sequences. Optionally, Use Case Scenarios.	**Use Cases**
	Business Services	List of main Business Services (named only.)	Initial Service description - result, main actions, cross-referenced to Concept Model	Each service fully documented, including input/output messages, validation, business rules, and data updates to the attribute level.	**Service Specification**
Data	**Data Management Services**	Contextual Model (optional) and a glossary defining the main entities and other important terms.	Concept Model (Business Object Model or Conceptual Data Model) with main entities, relationships, attributes, and rules.	Fully normalised Logical Data Model with all attributes fully defined and documented.	**Concept Modelling**
		Plan	*Understand*	*Specify*	

The "Agile Zone"

26

Techniques and methodologies

- The same techniques are used in different sequences, with different emphasis, in different methodologies.
- There is no single fixed sequence through the techniques.

Larger project: process-oriented / "outside-in" –


```
Project        Initial        As-is          To-be
Scope &        Concept        Process        Process        Business       Use Case
Objectives  →  Model       →  Workflow    →  Workflow    →  Service     →  Models
               (vocabulary)   Models         Models         Specifications
```

These are typical overall flows:
- there are many variations
- there is always much iteration

Refine Concept Model & Logical Data Model

Smaller project: service or use case-oriented / "inside-out" –

```
Project        Initial        Business       Business                      To-be
Scope &        Concept        Event          Service        Use Case       Process
Objectives  →  Model       →  Identification →  Specifications →  Models  →  Workflow
               (vocabulary)                                                  Models
```

Refine Concept Model & Logical Data Model

Let's dive in to Concept Modelling!

Also known as…

- Conceptual Data Modelling

- Business Object Modelling

- Domain Modelling

- (rarely) Fact Modelling

- and probably other terms…

What actually is a Concept Model / Data Model?

- A description of a business in terms of
  - **things** it needs to maintain records of – *Entities*
  - **facts about those things** – *Relationships & Attributes*
  - **policies & rules** *governing those things and facts*

- Models a view of the **real world**, not a technical design (therefore, stable and flexible)

- Can be comprehended by mere mortals (at least initially)

- Graham Witt – "A narrative supported by a graphic"

==**"Things" first, data later!**==

==Narrative component==

Student definition:
A Student is any person who has been admitted to the University, has accepted, and has enrolled in a course within a designated time. Faculty and staff members may also be Students
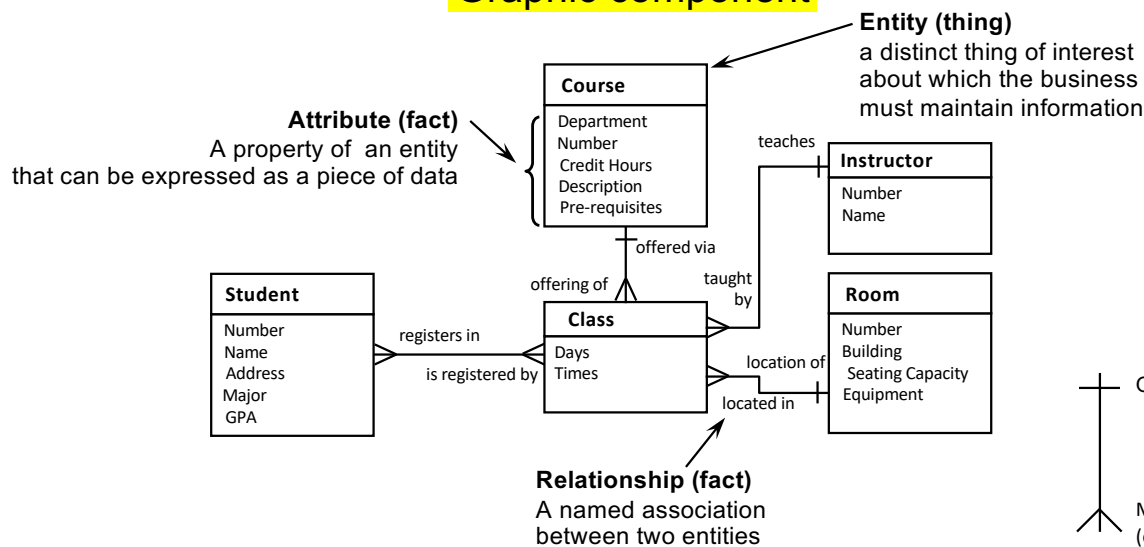
"Assertions" (policies & rules):
- Each Course can be offered through one or more Classes
  Each Class is an offering of a single, specific Course
- Each Instructor teaches one or more Classes
- Each Class is taught by one Instructor (which may or may not be true…)

Many rules can't be shown on the diagram:
- A Student can not register in two Classes of the same Course in the same Academic Term

==Graphic component==

Attribute (fact)
A property of an entity
that can be expressed as a piece of data

Entity (thing)
a distinct thing of interest about which the business must maintain information

Relationship (fact)
A named association between two entities

| — One |
| Many (or "Multiple" or "One or more") |

Course
Department
Number
Credit Hours
Description
Pre-requisites

Instructor
Number
Name

Room
Number
Building
Seating Capacity
Equipment

Student
Number
Name
Address
Major
GPA

Class
Days
Times

teaches
offered via
offering of
taught by
location of
located in
registers in
is registered by

29

A better looking version of the model on the previous slide

Independent Entities at the top
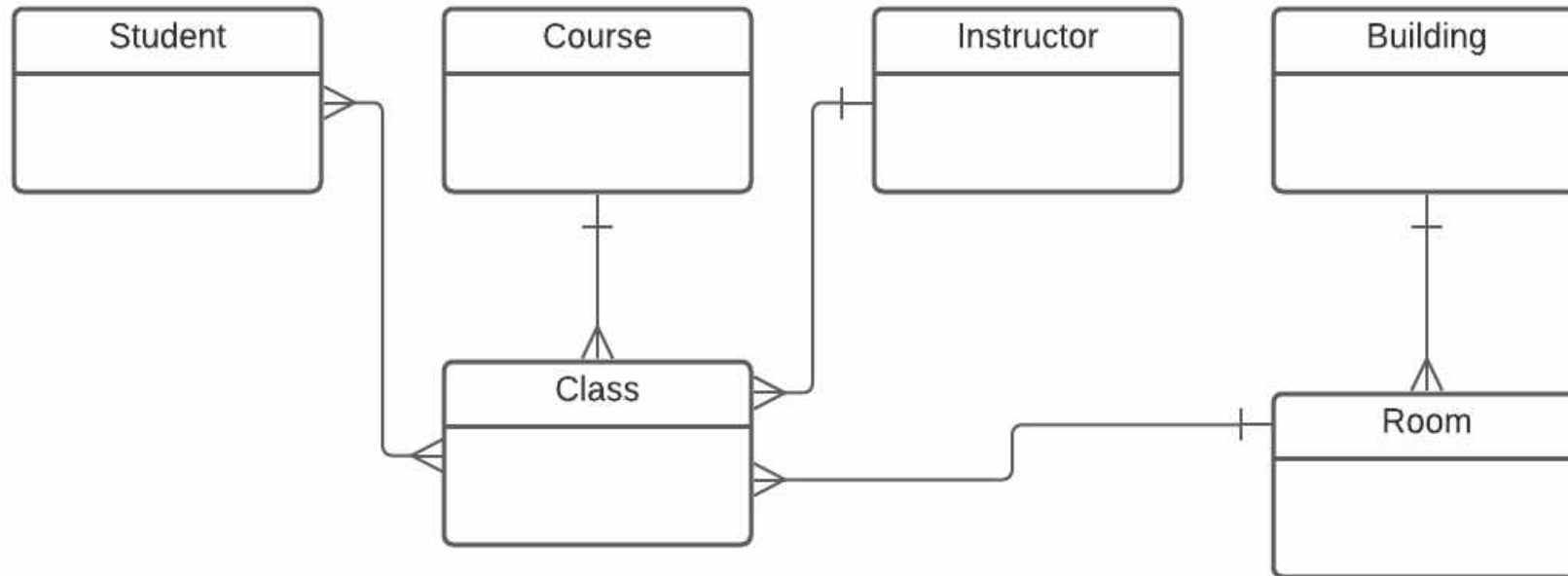
Drawn top-down by dependency

A few central ideas…

- Confusing *concept modelling / data modelling* with detailed *database design has* discouraged the use of data modelling – *but this is changing!*

- We don't call it "data modelling" because initially "data" is not the issue – we model:
  - the "things" / objects / concepts the business cares about:
    - terms and definitions – **language first!**
    - policies and rules
  - "things first, data later"

- A concept model provides a great platform for:
  - requirements discovery
  - package selection
  - business process change
  - business architecture, etc.

Concept Modelling principles
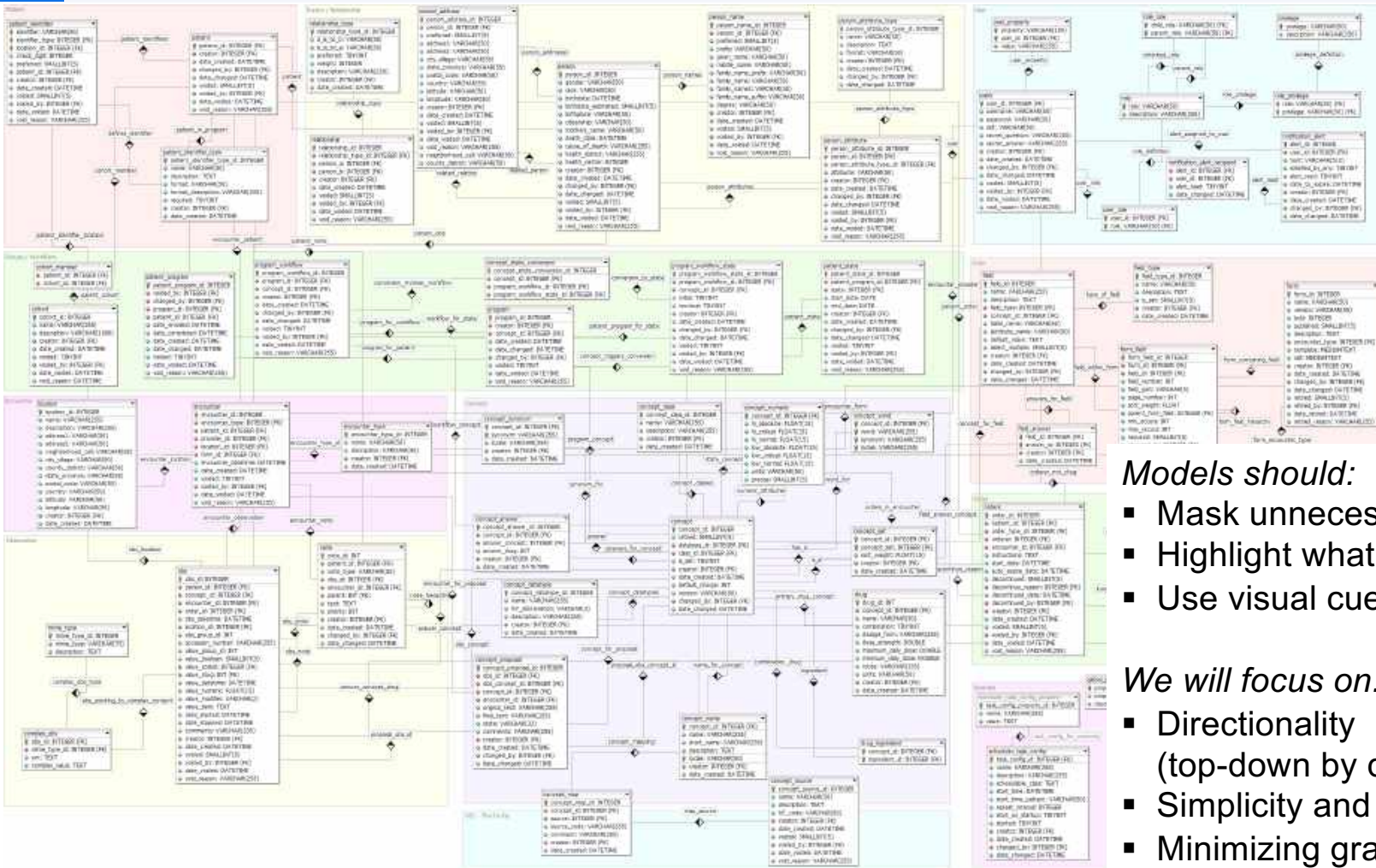
Models should:
- Mask unnecessary detail
- Highlight what matters
- Use visual cues consistently

We will focus on:
- Directionality
  (top-down by dependency)
- Simplicity and abstraction
- Minimizing graphic "widgets"

The basics: <u>E</u>RA – *Entities*

A distinct thing about which the enterprise must maintain facts in order to operate.

Criteria –

*   *singular noun* – we can talk about *one of them* ("Employee," not "Staff")
*   *multiple* instances
*   must *need to* and be *able to* keep track of *each* instance
*   has *facts* (attributes & relationships) that must be recorded
*   makes sense in a *"verb-noun"* pair
*   *NOT* an *artifact* like a spreadsheet or report

Fundamental to business analysis.
Entities are the things

*   processes act on
*   applications manipulate
*   databases record
*   BI & reporting tools provide info about

Two basic types:

*   independent – can stand alone
*   dependent – must have one or more parents

Course

Student

Building

Instructor

Academic Time Period

Class
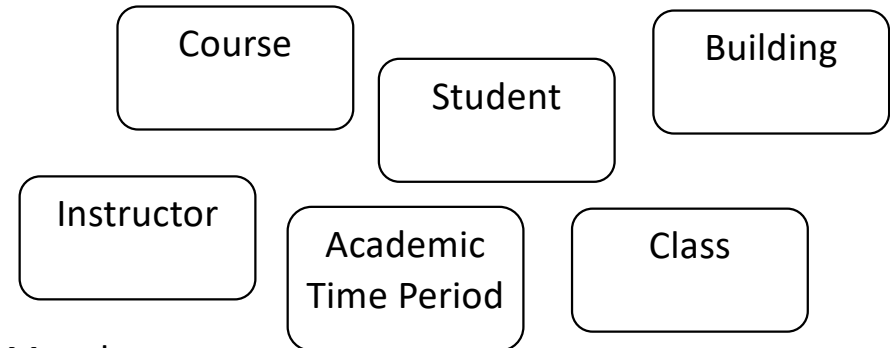
Must be:

- named: business-oriented noun / noun phrase

- defined: "What <u>is</u> one of these things?"
  or "What do you <u>mean</u> by _____?"

Independent ────────────► Instructor
- "strong"   Name
                            Telephone
- no relationships "on top" Office Location
  (no parents)

                            rated by
              rates
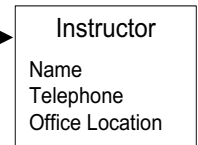
Dependent ────────────► Instructor
- "weak"   Evaluation
                          Date
- one or more relationships Performed By
  "on top" (to parent(s))   Evaluation Method
                            Overall Score

33
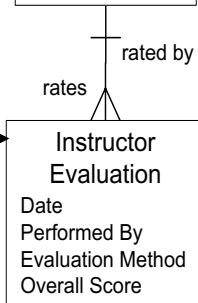
Naming and definition – the essence of Concept Modelling

Organisations need a *common language* more than ever…

- Data integration (data lake, data mesh, data fabric,
  data virtualisation, data warehouse,
  operational data store, …)

- Mergers/acquisitions/partnerships/…

- Business analysis – most requirements can't be stated
  without using a term from the Concept Model

- Performance measures, e.g., KPIs

Note – it often works best if you don't start by talking about
Concept Modelling or *Data Modelling…*
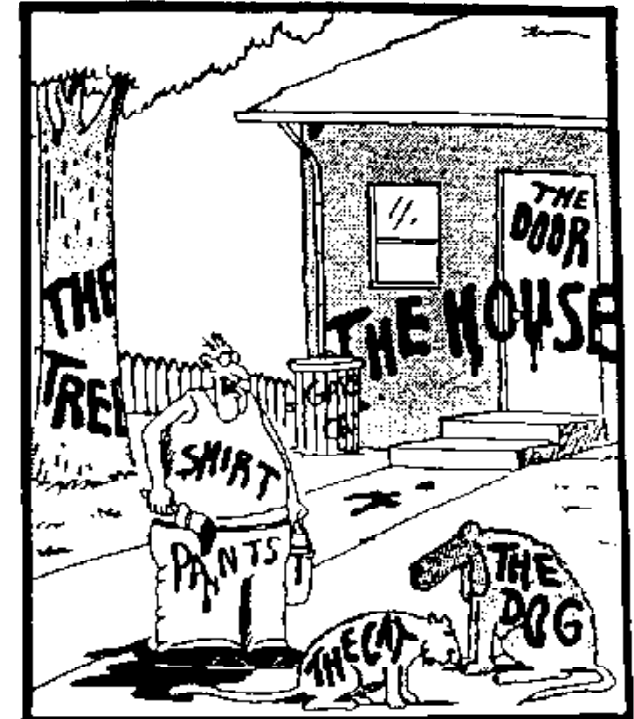
"Now! That should clear up
a few things around here!"

The basics – E<u>R</u>A – Relationships

An association between Entities that the business must keep track of

Named in both directions

- verb-based phrase

- the line tells us they *are* related,
  the name tells us *how*

Different types of relationships

1. parent-child or characterising – "bottom to top" relationship
   from an entity to a dependent entity (1:M)

2. associating – "side to side" relationship
   between entities that are not dependent on one another (usually M:M)

3. classifying – "side to side" relationship
   from reference data to the classified entity
   (seldom shown in the Concept Model)
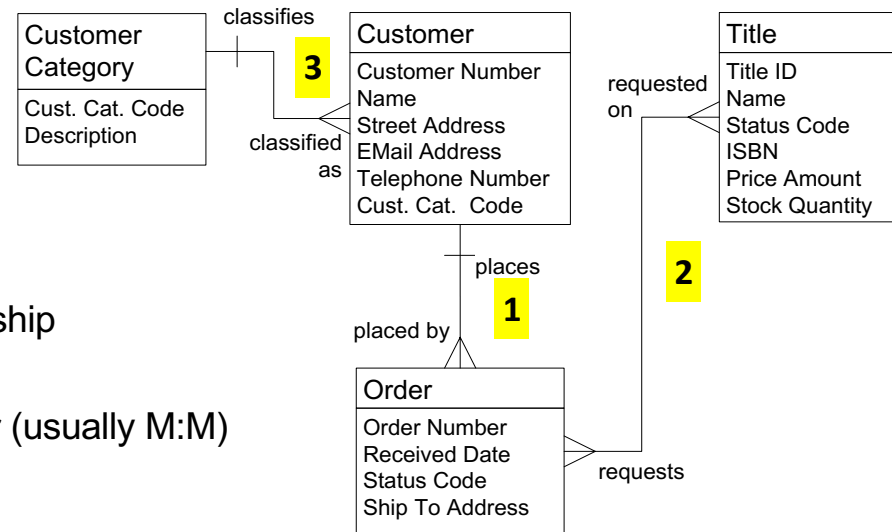
Dependency is shown top down – No Dead Crows

Relationships have rules

- cardinality – 1:1 (almost certainly wrong,) 1:M, M:M

- optionality – relationship *may be* present or *must be* present
  (not shown until later, in the logical model)

Customer Category	classifies **3**	Customer	requested on	Title
Cust. Cat. Code Description	classified as	Customer Number Name Street Address EMail Address Telephone Number Cust. Cat. Code		Title ID Name Status Code ISBN Price Amount Stock Quantity

places **1**

placed by

2

Order	requests
Order Number Received Date Status Code Ship To Address	

Order		Customer	
✗ Customer		✓ Order	

Relationship cardinality (maximum cardinality)

A kind of "business rule"
that applies to relationships

Each Building is the location of
a *maximum* of *Many* Positions
(or, ... *One or More* Positions)

Each Position is located at
a *maximum* of *One* Building
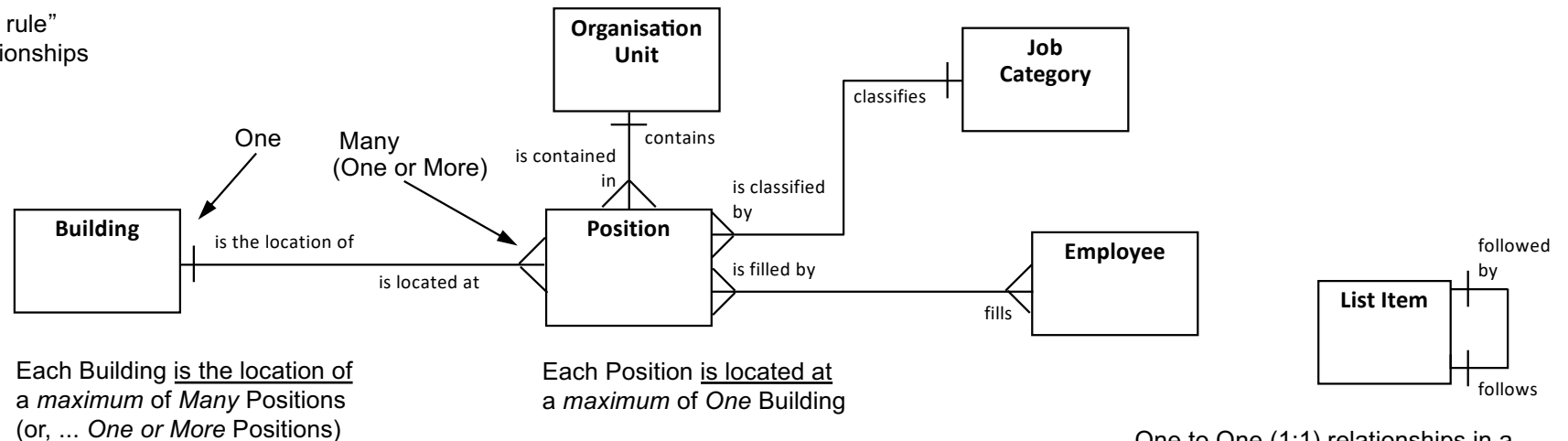
One to One (1:1) relationships in a
conceptual or logical model are
almost invariably an error except in
recursive relationships.

To determine cardinality, *first name the relationships properly, and only then:*

- for each entity, ask
  "Can one of these be related to a *maximum* of *One* of the other or a *maximum* of *Many* of the other?"

- record the answer (One or Many) at the "other" end;
  "One or More" works better for businesspersons than "Many"

- possibilities – 1:1 (error), 1:M (common), M:M (more work, eventually)

36

Relationships – state as assertions

1. You *must* state the relationship name as an assertion, in both directions (for clarity and confirmation)

2. Be clear on whether cardinality is "one" or "one or more" (don't worry about "may" and "must" at first)

3. *Emphatically* begin the assertion with the word "Each"

4. Try it on this model…

Each Instructor teaches one or more Classes (Sounds good…)

Each Class is taught by one Instructor…

Course

Department
Number
Credit Hours
Description
Pre-requisites

teaches

Instructor

Number
Name

offered via

offering of

taught by

Student

Number
Name
Address
Major
GPA

enrolls in

is enrolled by

Class

Days
Times

location of

located in

Room

Number
Building
Seating Capacity
Equipment
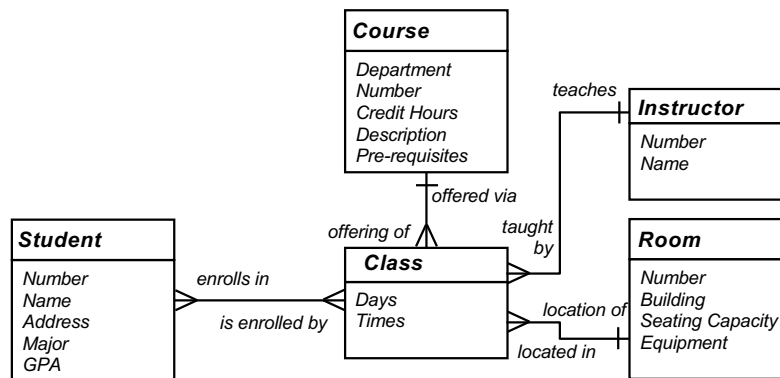
1. Student-Class

2. Course-Class

3. Instructor-Class

4. Room-Class

Which ones might be *incorrect?*

Discussion – state as assertions, identify incorrect ones

In some universities, Students in the same Class could be earning credit for *different* Courses – it could be a M:M relationship.

Math 100

Course

Department
Number
Credit Hours
Description
Pre-requisites

teaches

Instructor

Number
Name

is offered via

is an offering of

is taught by

Student

Number
Name
Address
Major
GPA

registers in

is registered by

Class

Days
Times

is the location of

Room

Number
Building
Seating Capacity
Equipment

is located in

Math 100, Class 3,
Spring Semester 2022
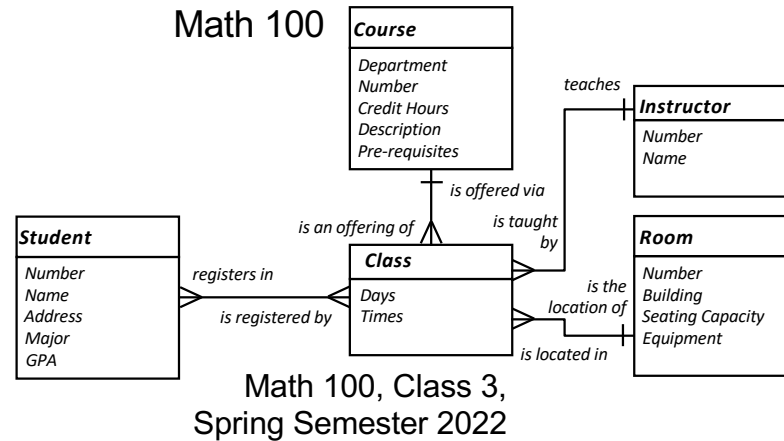
1. Student-Class
   Each Student *registers in* one or more Classes
   Each Class *is registered by* one or more Students ✓

2. Course-Class
   Each Course *is offered via* one or more Classes
   Each Class *is an offering of* one Course ❓ – depends on Policy

3. Instructor-Class
   Each Instructor *teaches* one or more Classes
   Each Class *is taught by* ✗one One or More Instructors

4. Room-Class
   Each Room *is the location of* one or more Classes
   Each Class *is located in* ✗one One or More Rooms

Each Class is taught by One or More Instructors. On what basis?
- team teaching
- backup
- replacement
- specialist
- guest lecturer
- lab assistant
- teaching assistant
- …

We are discovering reference data to describe an Instructor's Role.

All of this has an impact on the Business Process! It's easier to resolve these rules before working on the Process.

38

The basics: ERA – *Attributes*

A fact about an Entity recorded as a piece of data.
If facts are needed about a relationship,
we will later create an Entity that represents the
relationship and records its facts

Like Entities, attributes are named and defined

Not every possible fact – just the ones we need

Have properties

1. base or fundamental attribute

2. single-valued vs. multivalued –
   one attribute can have multiple values,
   at a time or *over* time

3. fundamental vs. redundant –
   the same value is recorded multiple times
   in different entities

4. "user-entered" vs. constrained –
   attribute can only come from a limited set,
   as in a drop-down list

Traditionally alphanumeric data; now includes richer types e.g.,
retinal scan image or voice audio clip

Eventually, an entity will contain only base /
fundamental / *essential* attributes:

- an *essential fact* about that thing (entity)
- *not* derived or calculated from other attributes;
  otherwise, clearly flagged "derived"
- *not* redundant
  (a redundant attribute is an attribute that is really an
  essential fact about a *different* entity, so its value is
  recorded multiple times, redundantly)

Customer
1 → Customer Number
1 → Name
Street Address
EMail Address
Telephone Number
2 → *Contact Points*

E.g., Mobile Number,
Office Telephone Number.
Facebook ID,
LinkedIn ID, …

places
placed by

Order
Order Number
3 → *Customer Name*
Received Date
Order Status Code ← 4
Ship To Address
2 → *Requested Titles*
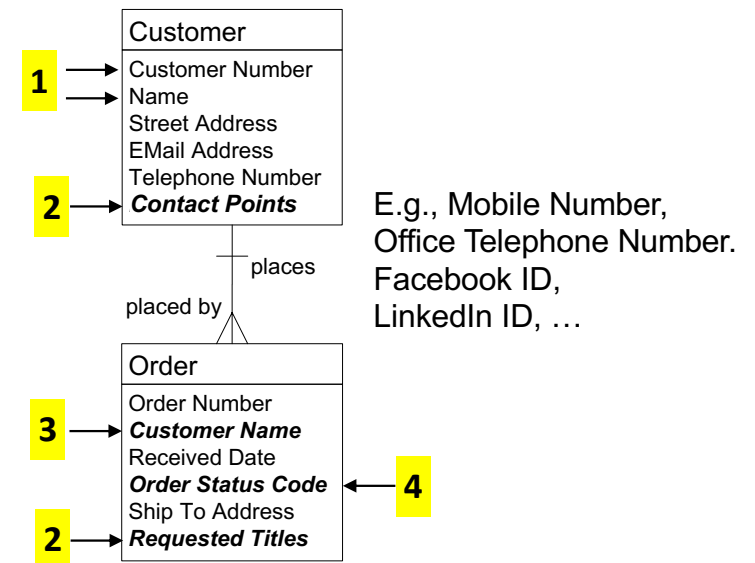
Summary – three types of data models

> **Different levels of detail support different perspectives**

1 Contextual (Scope)	**2** Conceptual (Overview)	**3** Logical (Detail)
✓ *Context model* ✓ Agreement on "big picture," context, and some vocabulary ✓ A block diagram of "subject areas," higher level than individual entities ✓ Shows the scope or "footprint" ✓ Optional – not useful on smaller projects	✓ *Concept Model* ✓ Agreements on basic concepts, vocabulary, and rules	✓ *Logical Data Model* ✓ Complete detail for physical design

Some important differences

✓ Main ("recognisable") entities only ✓ Main attributes only, many are non-atomic ✓ M:M relationships ✓ Doesn't show keys ✓ Not normalised ✓ A "one-pager"	✓ All granular entities ✓ All attributes included, all are atomic ✓ All M:M resolved ✓ Shows primary & foreign keys ✓ Fully normalised ✓ Five times as many entities

40

For reference – the Information Engineering symbol set

- This symbol set was refined and developed by Clive Finkelstein.
- Known in some tools as the "Martin IE" symbol set.
- Strengths are:
  - symbols are not "overloaded" – they explicitly convey only *one* idea.
  - can show as much or as little as needed in terms of rules.
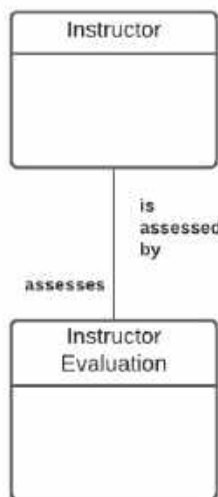
The two entities
are related -
that's all this shows
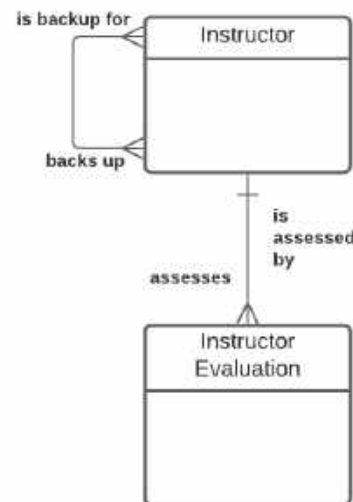
There is a 1:M relationship
from the parent entity
(business object) to the
child entity (business object.)
Optionality is not shown.
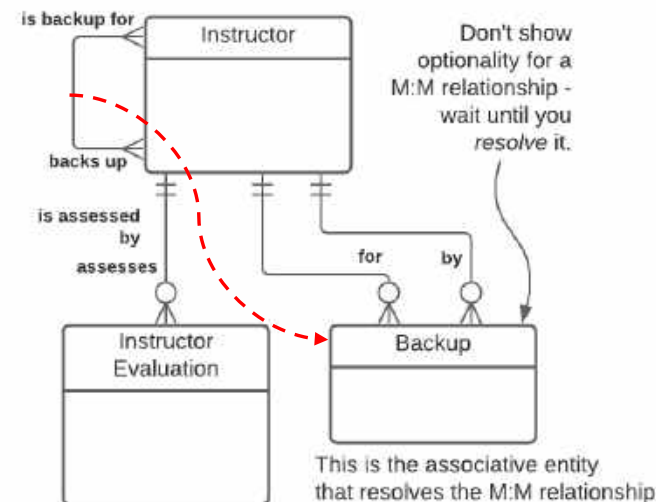
There is a 1:M relationship
from parent to child,
optional for the parent and
mandatory for the child.
(The parent *may* have a child,
the child *must* have a parent.)
This is by far the most common
relationship in a logical model.

Don't show
optionality for a
M:M relationship -
wait until you
resolve it.

This is the associative entity
that resolves the M:M relationship

A look ahead – from Concept Model to Logical Data Model

partial Concept Model

beginning the Logical Data Model

In this example we:
- resolve the M:M relationship between Instructor and Class
- move redundant Department attributes in Course up into a new Department entity
- create a reference entity to standardise the values of "Assignment Role"

Kernel

Characteristic

Kernel

Characteristic

Associative

Reference

Another example – Concept Model

- Shows main or core entities, relationships, and attributes

- Gets the *concept* across

- Great for communication, but not for database design

- Every Entity is a *singular noun* the business refers to daily

- Best done **before** any significant process modelling or application requirements (use cases and service specifications)
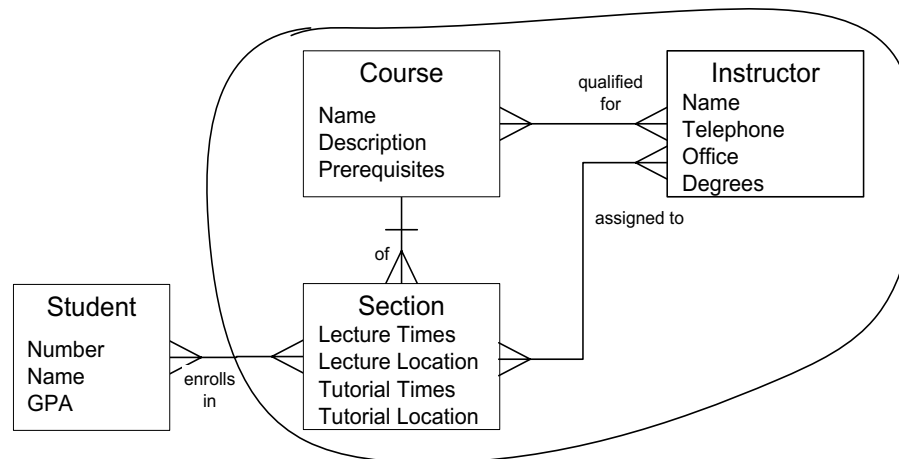
Let's see what happens when we take these three entities to the Logical level...

Another example – Logical Data Model

- All necessary detail –
  the data specifications

- Input to initial
  physical database design

- Completed *after* Use Cases
  and Service Specifications

Department

Department ID
Name
Abbreviated Name
Main Tel. Number
etc.

of   filled by

Dean

Department ID
Faculty Member ID
Effective Date
End Date

Faculty Member

Faculty Member ID
Name
Office Tel. Number
Building Code
Room Number
Rank Code
Rank Date
etc.

offered by

Course

Department ID
Course Number
Credit Hours
Method Code
Description Text
Outline Text
etc.

prerequisite
for

to teach   received by

Course Qualification

Department ID
Course Number
Faculty Member ID
Qualification SID
Rating Factor
Preference Code
Next Review Date
Qualification Eff. Date
Qualification End Date

attained by

Fac. Mem. Degree

Faculty Member ID
F. M. Degree SID
Level Code
Subject
Institution Name
Granted Date

for   for   offering of

Section Tutorial

Department ID
Course Number
Semester ID
Section Number
Tutorial Number
Day Code
Start Time
Duration
Building Code
Room Number

Section Lecture

Department ID
Course Number
Semester ID
Section Number
Lecture Number
Day Code
Start Time
Duration
Building Code
Room Number

Section

Department ID
Course Number
Semester ID
Section Number
Maximum Size
Status Code

to teach   assigned to

Instr. Assignment

Department ID
Course Number
Semester ID
Section Number
Faculty Member ID

44

A natural progression

Contextual

Focus – scope
context and boundaries,
glossary of main terms and definitions

Conceptual

Focus – overview / shared understanding
business perspective, all terms and definitions,
overall structure, major facts and rules

Logical

Focus – detail
all facts, detailed rules,
input to initial physical design

traditional modelling and development

reverse engineering

**Physical
DB
Design**

**The "Danger Zone" Analysts
should not worry about
physical design issues while
data modelling.**

!	*Get into the high "value-added" space*

- Contextual – helpful for large models
- Conceptual – a great way to add value
  - Improve communication among all players
  - Highlight disconnects – terms, rules, scope, …

45

Different ways to get started

Starting the model

Developing the model

Identify the main "topics" within scope, and what needs to be known about them.

Contextual

Top-down

Study reports, screens, forms, policies, laws, regulations, training material, industry standards, and other current artifacts or sources

"Sideways-in"

Conceptual

Top-down

Gather details, isolate things, put together basic model

Bottom-up

Logical

Some advice on starting the concept model

Don't begin with a lecture
on data modelling
(but I have a painful story
that had a happy ending)

**If you can, don't
even mention "data
modelling"**

_We use "terminology analysis" –
starting with the nouns –
at the outset of every project.
This was demonstrated earlier in the
Client Safety Management example._

47

Starting a data model bottom-up

1) Interview business representatives about their area:
   mandate and activities, goals and objectives, issues and opportunities,
   needs and wants, likes and dislikes, etc.…

   Nod sympathetically but ignore it all (almost!)

   Instead, capture "terms" – anything that goes by a name.

2) Later, write each term on a large Post-it

3) In a facilitated session, participants sort terms into categories:

- Things (entities, but don't use the term… yet)
- Facts about things (add new "thing" if it's not there already)
- "Other stuff"

As needed, introduce criteria to
be a "thing" (an entity)

"Other stuff" includes:
- Metrics
- Organisations, departments, jobs, roles, …
- Processes, functions, activities, tasks, …
- Systems, tools, equipment, mechanisms, …
- Reports, forms, screens, queries, …
- Other – too vague, only one instance,
  a "fact of life," not a thing we track, etc.

Exercise 1: Starting a conceptual data model

The assignment:
The following describes project tracking at Amalgamated Automaton. Read it over and be prepared to discuss the things about which the business needs to record information, and the important facts about them. The instructor will lead the development of an initial data model.

Amalgamated Automaton, Inc. has a growing Information Systems department. Until recent years, the department was concerned almost entirely with selecting, installing and maintaining purchased software packages. Recently, however, the focus has shifted towards the in-house development of application software.

One of the problems confronting the IS department is that they have no base of historical data to aid in trend analysis or estimating development effort, nor any effective means of charging back development costs. The proposed solution is to develop a simple Project Tracking System, which will work in conjunction with the existing Personnel and General Ledger Systems.

When a development project is initiated, a project name and a short description are recorded, among other things. Soon, before any further work is done on the project, a new account is created on the G/L System, identified by a G/L account number. Project costs will be charged to this account, and the project budget is recorded as the initial account balance in dollars.

Project planners break a project down into many tasks, perhaps hundreds. A typical project task might be "Test Order Entry Module". Some of the facts which are required about tasks include a brief task description, estimated work hours, and the scheduled start and finish dates.

Eventually, individual employees are assigned responsibility for the tasks. Some tasks will be the responsibility of many employees, and an employee might be assigned to many tasks. As each employee is assigned to a project task, their planned start and finish dates, their contribution to the task (not a "kind of work," but their specific duties on the task – e.g., "Develop test scripts"), and the estimated number of hours they are to spend on the task are recorded. Employee information such as the employee name and number are available from the existing Personnel System, although it will have to be modified to record the employee's hourly charge out rate.

When an IS employee begins work on a new task, their actual start date is recorded. A running total of the number of hours that they have worked on each started task is updated regularly. At the same time, the remaining balance in the project account is updated. When an employee completes a task assignment, the actual completion date is recorded.

49

Workshop example
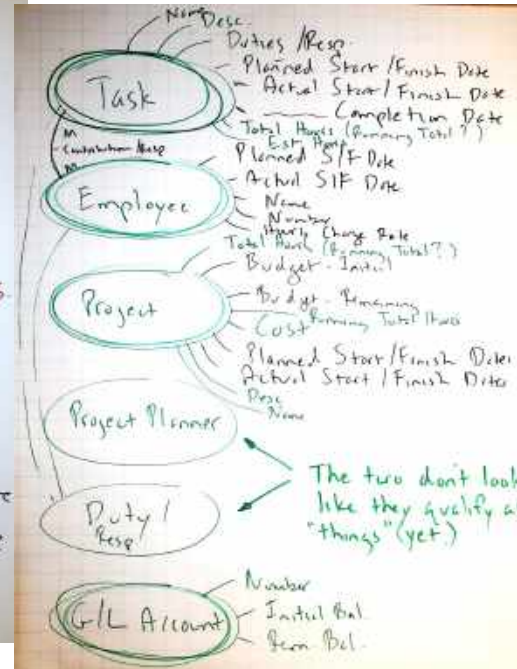
Introduce "thing criteria" as necessary:

- *singular noun* – can talk about *one of them* (*Worker* not *Staff*, *Item* not *Inventory*)
- *multiple* instances
- must *need to* and be *able to* track *each* instance (uniquely identify each)
- has *facts* that must be recorded
- makes sense in a *"verb-noun"* pair
- *NOT an artifact* like a spreadsheet or report (not a Call Log or Worker Directory or…)

Entities – more specific criteria

An *entity* is a distinct thing the business *needs* to know about,
often described as a *person, place, thing, event, concept,* or *organisation* and…

- is named with a *singular noun* that implies a single instance
  - not a plural or collective noun, list, set, collection, report, etc.
  - we can discuss "one of them"

- has *multiple occurrences* (or instances)
  - *need* to and *can* keep track of (differentiate) each occurrence

- has *facts* that must be recorded, e.g.
  - *Student* attributes: Number, Name, Birth Date, Major, GPA, …
  - *Student* relationships: "majors in" *Subject*, "enrolls in" *Section*

- is acted on by *processes,* so they make sense in a "verb-noun" pair

- refers to the *essence*, not the implementation ("What, not who or how") –
  *the most common error is to identify artifacts (forms, reports, spreadsheets, …)
  as entities!*
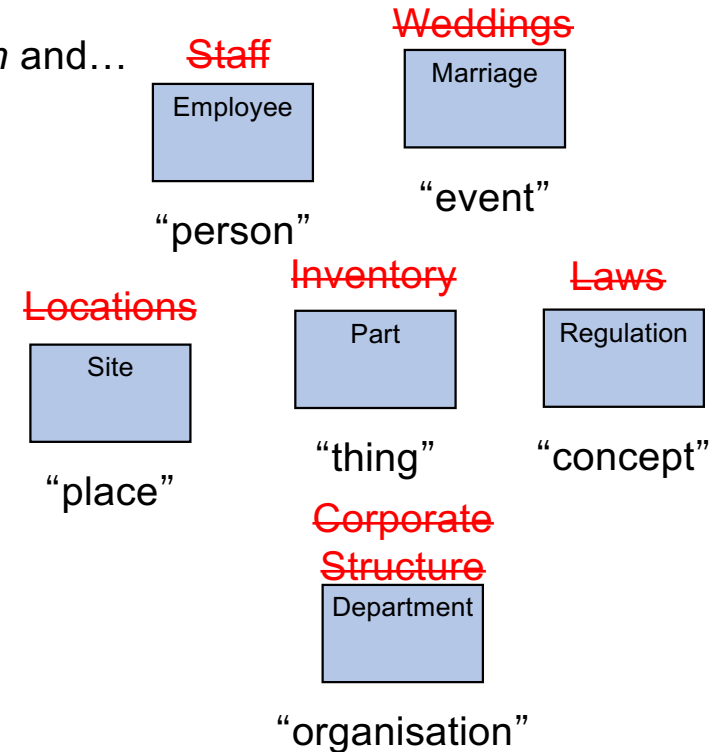
Let's look at some common errors…

~~Staff~~
Employee
"person"

~~Weddings~~
Marriage
"event"

~~Locations~~
Site
"place"

~~Inventory~~
Part
"thing"

~~Laws~~
Regulation
"concept"

~~Corporate Structure~~
Department
"organisation"

Identifying Entities – four common errors

1. Treating an "artifact" (a spreadsheet, report, web page, form, etc.) as an Entity – an Entity is a fundamental thing – *"what"* – with no reference to *"who or how."* Artifacts typically contain attributes from *multiple* Entities
e.g., *"Admission Request Form"* or *"Orders Summary Spreadsheet"*
or *"Daily Call Log"* or *"Class Roster"* or *"Materials List Fax"* or…

2. The "types vs. instances" problem – failing to clarify if the Entity deals with *types* of things (or *categories* or *kinds* or *classes* of things)
vs. specific *instances* of things
e.g., *"Vehicle"*
(An example of this is coming up.)

3. Identifying an Entity that exists in the real world,
but whose *instances* can't be uniquely identified
e.g., *"Transit System Passenger"*

4. Identifying Entities that are simply too vague, or are just a "fact of life;"
that is, the name doesn't imply a single *instance*
e.g., *"Weather"* or *"the Environment"* or *"the Economy"* or *"Society"*

Types vs. Instances – "What do you mean by a Bus?"

A category of Bus – a "meta-Type?"
(transit, articulated, intercity, minibus, …)
A Make and Model of Bus – a Type?
An individual Vehicle? – an Instance?

Model	Length	Width	Introduced
Xcelsior[18]	35 feet (11 m) 40 feet (12 m) 60 feet (18 m)	102 inches (2.6 m)	2008
MiDi	30 feet (9.1 m) 35 feet (11 m)	96 inches (2.4 m)	2013

"What do you mean by a Bus?"

254 British Properties

Inbound From Glenmore and Bonnymuir via Bonnymuir, Stevens, Taylor Way to Park Royal terminus (extends to Downtown Vancouver during Monday-Friday peak hours).

Outbound From Park Royal (from Downtown Vancouver during Monday-Friday peak hours) via Marine Drive, Park Royal South, Taylor Way, Southborough, Eyremount, Cross Creek, Chartwell, Crestwell, Eyremount, Fairmile, Southborough, King Georges Way, Robin Hood, Kenwood, St. Andrews, Bonnymuir to Glenmore terminus.

Park Royal to British Properties and return to Park Royal

MONDAY TO FRIDAY							
Connecting Buses Leave Downtown Vancouver	Leave Park Royal	Leave Eyremount at Highland	Leave Bonnymuir at Glenmore	Leave Eyremount at Highland	Leave Marine at 14th	Arrive Park Royal	Arrive Downtown Vancouver Connecting Buses
6.35	6.53R		7.03	7.15	7.31	7.34	7.54
6.45	7.23R		7.33	7.45	8.01	8.04	8.24
7.47	8.07R		8.17	8.28	8.44*	8.47	9.16
8.20	8.40	8.53	9.06		-	9.15P*	9.41
9.22	9.47P	10.00	10.13		-	10.22P*	10.43

254 British Properties

A Bus Route?

A Bus Route Scheduled Departure

An instance of a Bus Route Scheduled Departure?

Never be afraid to ask "What do you mean by…?"

Discussion – good Entity or not?

Which of the following might **not** be valid entities?
And if not, *why* not?

Transcript	Student	Building	Student Directory	Faculty Member	Instructor History
Department	Course	Organisation Chart	Prerequisite List	Payment	Student Body
Class Roster	Scholarship	Faculty	Assistant Dean	Admission Date	Phillips Building
Registration	Section	Course Catalogue	Physics	Class	Professor
		Admission Request Form			

Thinking space…

Discussion – good Entity or not?

Which of the following might **not** be valid entities?
And if not, *why* not?

✗ Transcript
a report

✓ Student

✓ Building

✗ Student Directory
a report

✓ Faculty Member

✗ Instructor History
a list, "history" is not singular

✓ Department

✓ Course

✗ Organisation Chart
a visual report

✗ Prerequisite List
a list

✓ Payment

✗ Student Body
not singular

✗ Class Roster
a report

✓ Scholarship

✓ Faculty

✗ Assistant Dean
a Job Title

✗ Admission Date
an attribute

✗ Phillips Building
an instance

✓ Registration

✓ Section

✗ Course Catalogue
a report

✗ Physics
an instance

✓ Class

✗ Professor
a Job Title

✗ Admission Request Form
a form (artifact)

58

Entity definition basics

Definitions must focus on what a single instance is:

- Not "how they're used" or "how they're created" or "why we care" or "how the process works" or "interesting problems and tidbits" etc.
- They simply answer the question "What is *one* of these things?"

The most useful questions:

"Can anyone think of examples that might surprise someone else – that is, anomalies or potential sources of confusion?" E.g., to define *Customer…*

- "In our area, other divisions are treated as customers"
- "We record recipients of charitable donations as customers."

"Could we list some examples?"

- Rita Smith, Acme Auto, Ministry of Finance, homeowners… (aha!)

"Does this deal with "kinds of things" or "specific things?"

- "kind" - Customer Category vs. "specific" – an individual Customer
- if it's a specific thing, still ask if there are recognised types (e.g., Personal, Corporate, Government; Lead, Prospect, Active)

!	Key Point
"What is one of these things?"	

59

Entity definition – bad example then a good format

Customer
We have a variety of Customers that operate in multiple geographies, and these must be tracked in order to consolidate purchasing statistics and enable our rating process to identify our best Customers.

Customer

1. A Customer is a person or organisation that is a past, present, or potential user of our products or services.

2. Current examples include Solectron (contract manufacturer,) Cisco Systems (OEM,) Arrow Electronics (distributor,) Best Buy (retailer,) M&P PCs (assembler,) and individual consumers.

3. Excludes the company itself when we use our own products or services but includes cases where the Customer doesn't have to pay (e.g., a charity.)

Entity definition format:

1. A description of which real-world things will be included in scope.
   This might be developed from a list of standard "thing types" – person, organisation, request, transfer, item, location, activity, etc.
   Be sure to identify any specific inclusions ("This includes…" or "This is…")

2. Illustrate with examples:
   - 5 – 10 sample instances
   - diagrams or scenarios
   - illustrations such as reports or forms

3. Interesting points – anomalies, synonyms, common points of confusion, etc.
   May include specific exclusions ("This excludes…" or "This is not…")

Discussion – starting an Entity definition

"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Employee" means?

-
-
-
-
-
-
- *...*

Employee

Project

Account

Task

Discussion – starting an Entity definition

*"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Employee" means?*

F/T vs. P/T?

Only IS Department?

Include management,
or only individual contributors?

Still in recruitment (an applicant)?

Onboarded? on probation? active? retirees?

Include contractors, student interns, vendor staff, etc.?

Volunteers?

A type of worker (DBA or tester) or a specific person?

A robotic, automated, or AI agent?

Employee

Project

Account

Task

62

Starting an Entity definition

"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Employee" means?

F/T vs. P/T?	*– Both*
Only IS Department?	*– No*
Include management, or only individual contributors?	*– Yes, everyone*
Still in recruitment (an applicant)?	*– No*
Onboarded? on probation? active? retirees?	*– Yes, all*
Include contractors, student interns, vendor staff, etc.?	*– Yes, all*
Volunteers?	*– Yes*
A type of worker (DBA or tester) or a specific person?	*– No, only a specific person*
A robotic, automated, or AI agent?	*– No, only a real person*

Employee

Project

Account

Task

Defining the Entity "~~Employee~~" – "Worker"

Definition format:

1. A description of which real-world things are within in scope, and any specific inclusions ("This *includes*…" or "This *is*...")

2. Illustrate with examples – 5 to 10 sample instances or types

3. Interesting points – anomalies, synonyms, common points of confusion, etc.
   May include specific exclusions
   ("This *excludes*…" or "This *is not*...")

Worker (renamed from Employee):

A *Worker* is a person, whether or not directly employed by *the company,* but with some sort of employment contract or arrangement, who has been or may be assigned to a Project.

Worker includes:
- Full or Part-time Employees who have been onboarded, including Probation, Active, Seconded, Suspended, Retired…
- Contractors
- Consultants
- Student Interns
- Vendor Staff Persons
- Company Owners and Managers

Key points:
- "Worker" was chosen as the entity name because it is more generalised than "Employee."
- A Worker may not necessarily be billable on a Project, e.g., a non-chargeable Subject Matter Expert or Volunteer
- Worker excludes:
  - Job Roles, e.g., DBA or Technical Writer
  - Robotic, Automated, or AI Agents (this might change)

Another example – starting an entity definition for Task

"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Task" means?

-
-
-
-
-

Worker

Project

Account

Task

Another example – starting an entity definition for Task

*"Can anyone think of examples that might surprise someone else –
that is, anomalies or potential sources of confusion."
E.g., how could we legitimately have different ideas what "Task" means?*

Key points that typically arise:

- A *type* of Task or a *specific* Task?
- Part of a *specific* Project or used across *multiple* Projects?
- Produces a specific *deliverable* or *state*?
- Time-bounded or ongoing?
- Performed by *one* Worker or *one or more* Workers?
- …

A ***Task*** is a specific, time-bounded, unit of work, within a single Project, intended to be performed by one or more Workers, that produces an intended deliverable or achieves a specific state.

Examples:

- Code *Place Order* service
- Test *Place Order* service

Excludes:

- types of Tasks
- ongoing (non time-bounded) activities such as management or administration
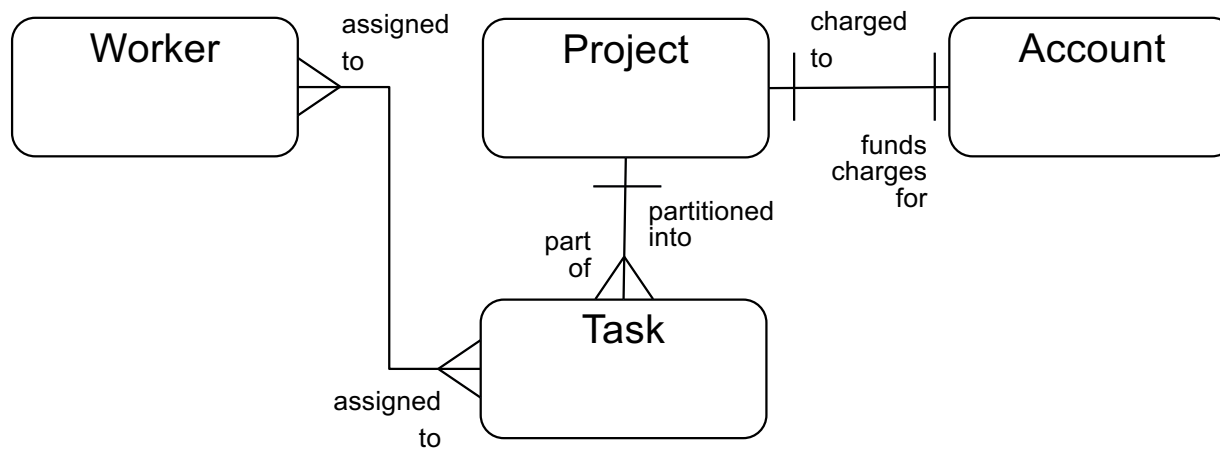
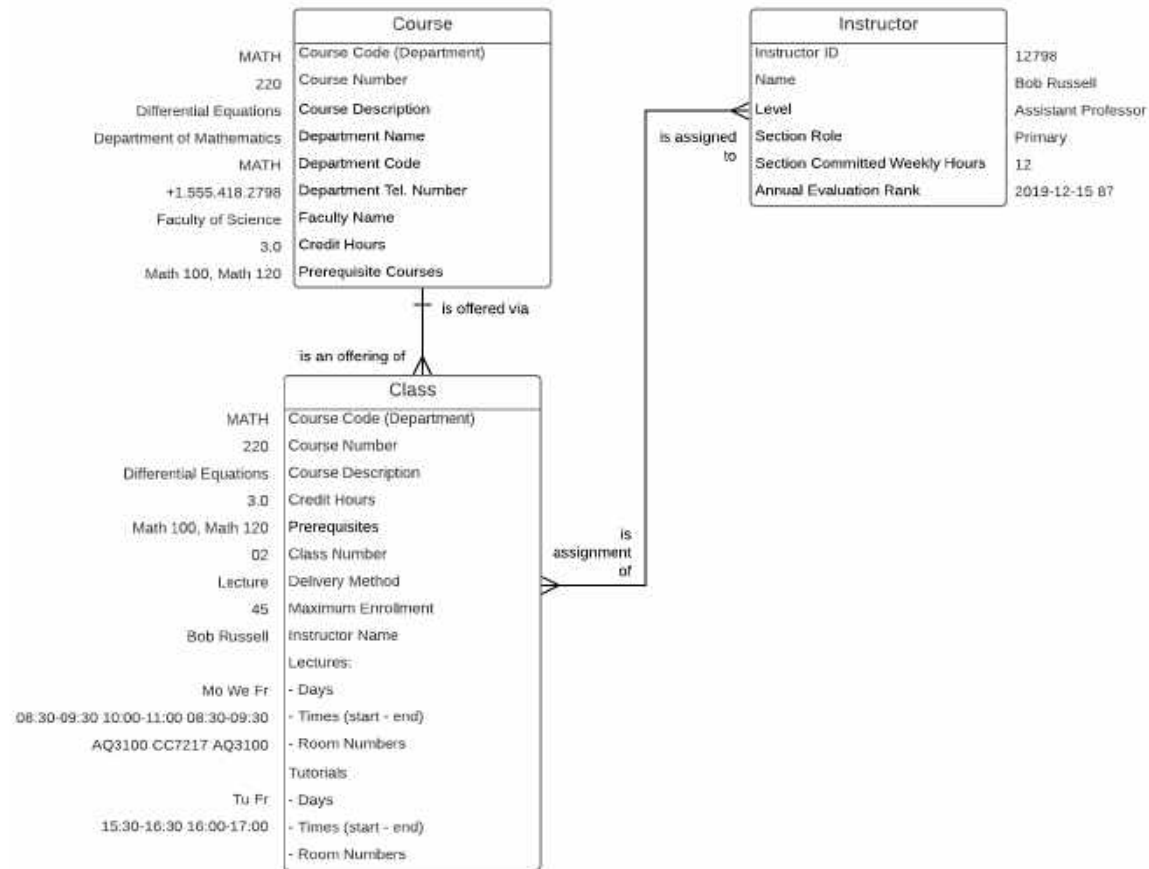Worker

Project

Account

Task

66

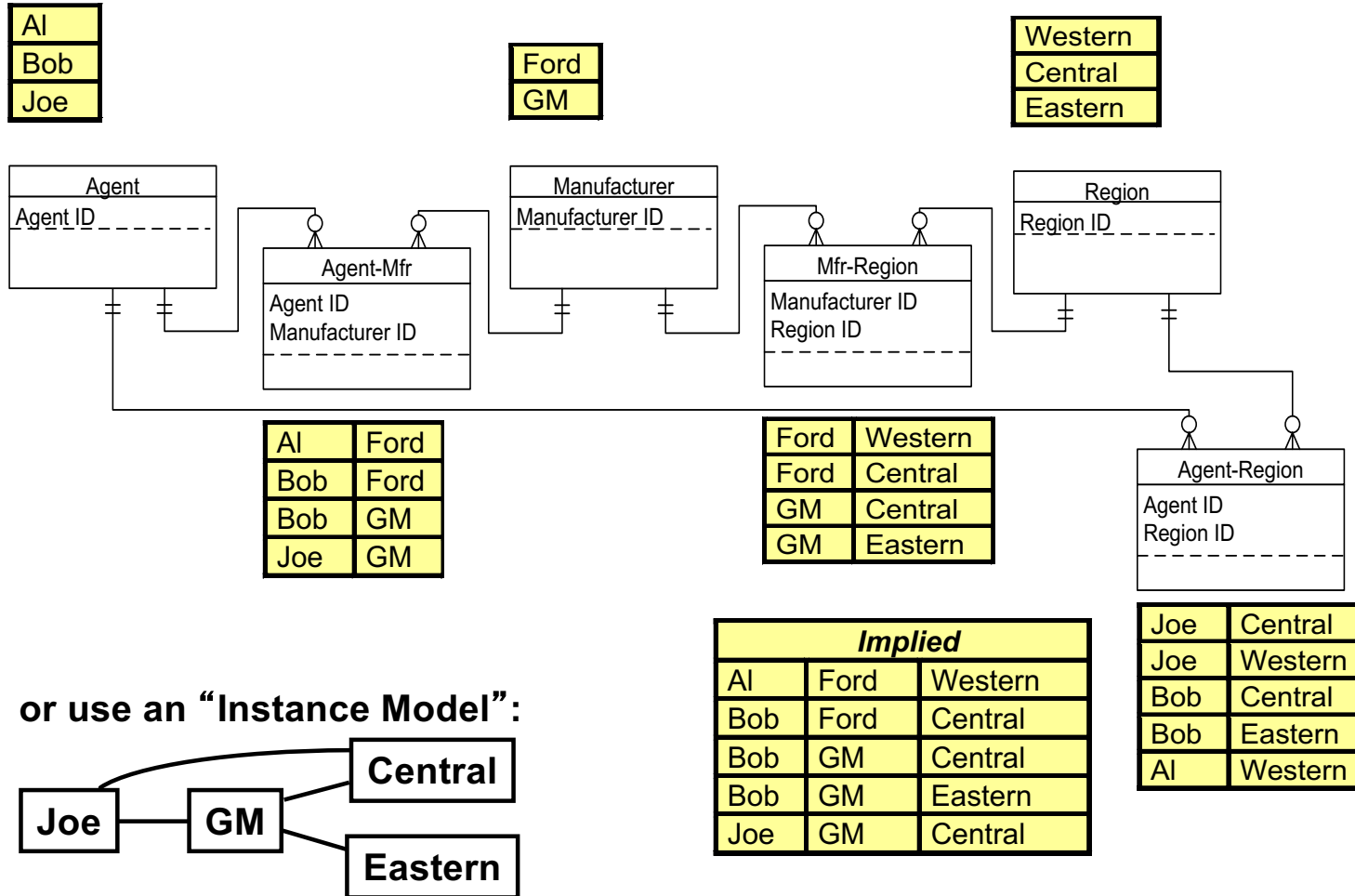# *Now we have definitions – it's "safe" to draw the ER model*

First arrange entities top-down by dependency.

Then add relationships with a verb-based phrase.

Then add cardinality (1:1, 1:M, M:M.)

"Demonstrate the Data"

*In addition to Entity definitions, it can be helpful to show
sample data values on an E-R Diagram.*

Sample instance tables (SITs) are also helpful

Al
Bob
Joe

Ford
GM

Western
Central
Eastern

Agent
Agent ID

Manufacturer
Manufacturer ID

Region
Region ID

Agent-Mfr
Agent ID
Manufacturer ID

Mfr-Region
Manufacturer ID
Region ID

Agent-Region
Agent ID
Region ID

Al	Ford
Bob	Ford
Bob	GM
Joe	GM

Ford	Western
Ford	Central
GM	Central
GM	Eastern

Joe	Central
Joe	Western
Bob	Central
Bob	Eastern
Al	Western

Implied		
Al	Ford	Western
Bob	Ford	Central
Bob	GM	Central
Bob	GM	Eastern
Joe	GM	Central

or use an "Instance Model":

Joe	—	GM	—	**Central**
				Eastern

Exercise 1 – initial E-R Diagram

Basic framework...

Worker

Project

charged to

G/L Account

Note:

We've named the entity "Worker" rather than "Employee" because that name is more general. (E.g., it includes contractors and interns.)

is part of

Task

assigned to
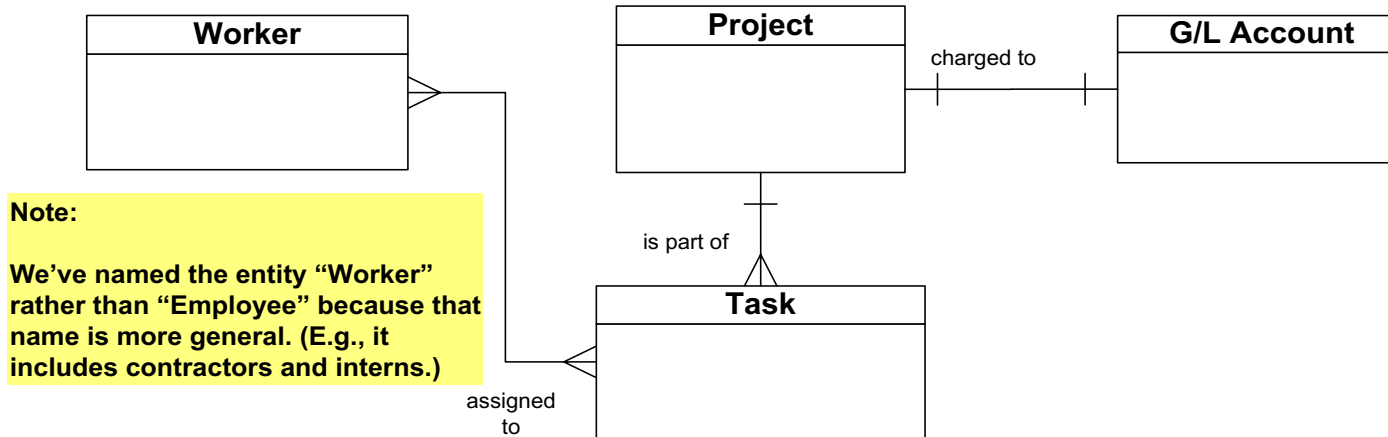
With simplified attributes...

Worker
Name
Number
Charge Rate
Assigned Tasks

Project
Name
Description

charged to

G/L Account
Number
Balance

is part of

Task
Description
Dates
Assigned Workers

assigned to
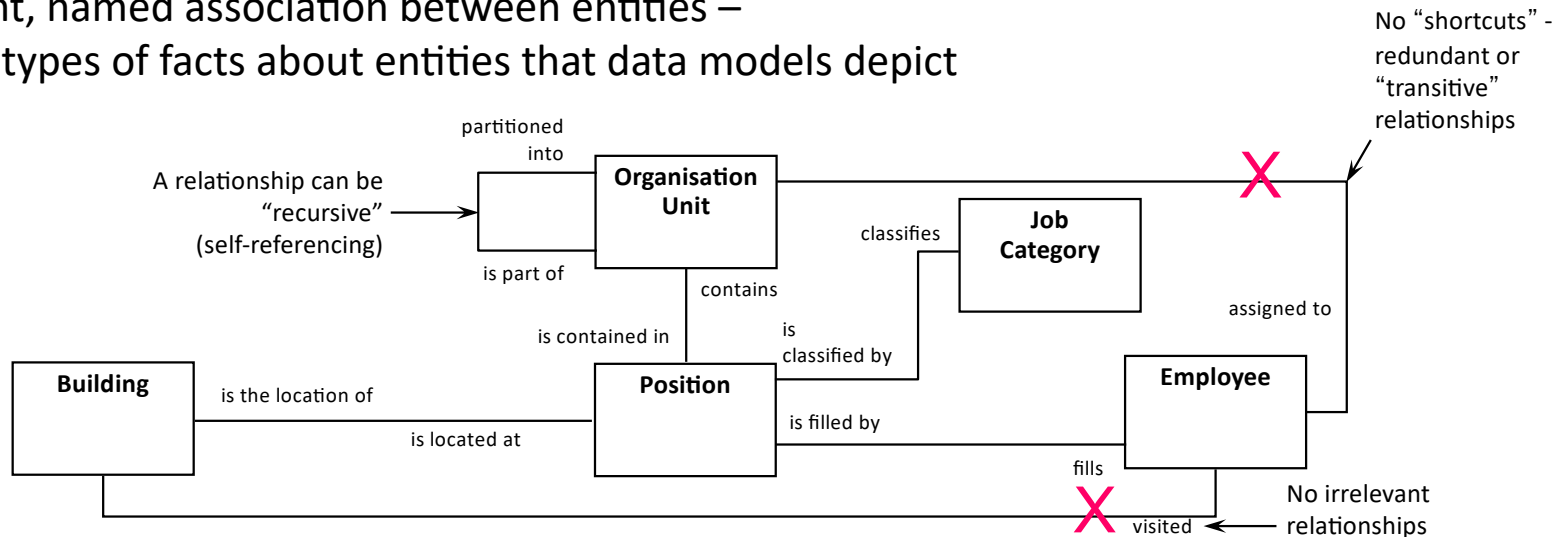
70
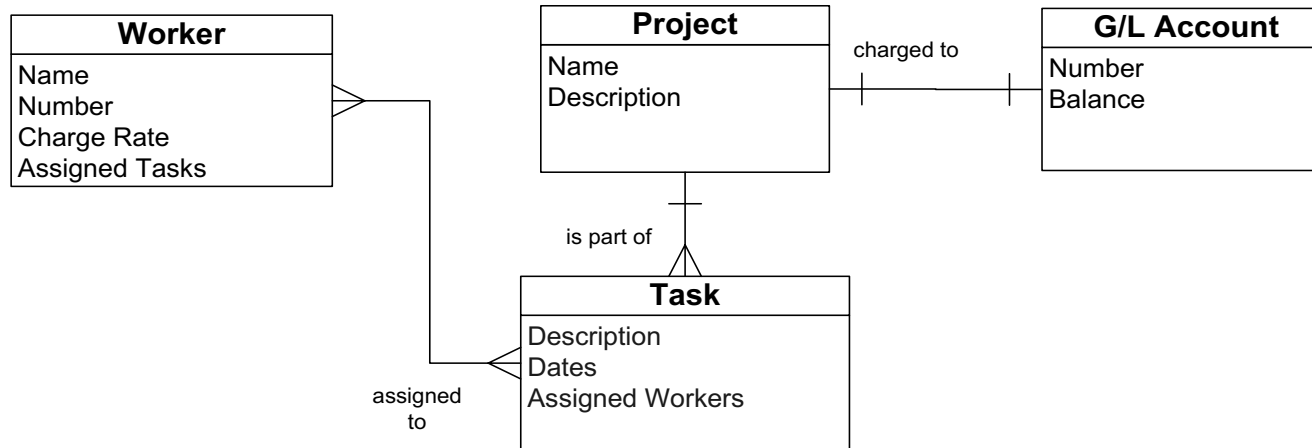
Relationships – a few more points

A significant, named association between entities –
one of the types of facts about entities that data models depict
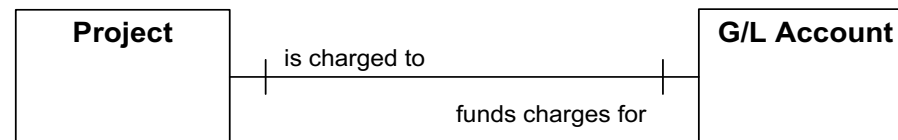

Guidelines

- named with a descriptive, verb-based phrase – not "has" or "is related to"
  (the line tells us they *are* related; the name tells us *how*)

- named in both directions – try to use the same root word at both ends
  (e.g., "classifies" and "is classified by")

- the complete name reads like a sentence (noun verb noun) –
  "Position is classified by Job Category"

1:1 relationships – almost always an error!

- ■ Note – a 1:1 relationship might be necessary in the Physical Database Design
  e.g., "Fixed Asset" records financial data about a "Network Component" but they are in
  two separate systems (the G/L System and the Configuration Management System)
  connected by a 1:1 relationship

Network Component	—— is recorded as —— records details of	Fixed Asset
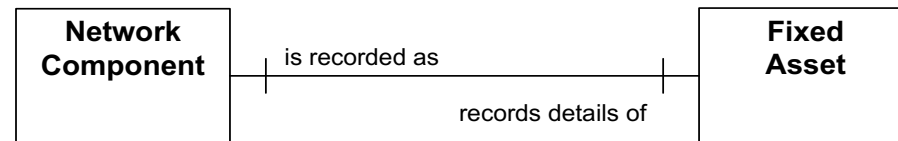
- ✖ Incorrect analysis
  e.g., Project costs are probably prorated across *many* Accounts

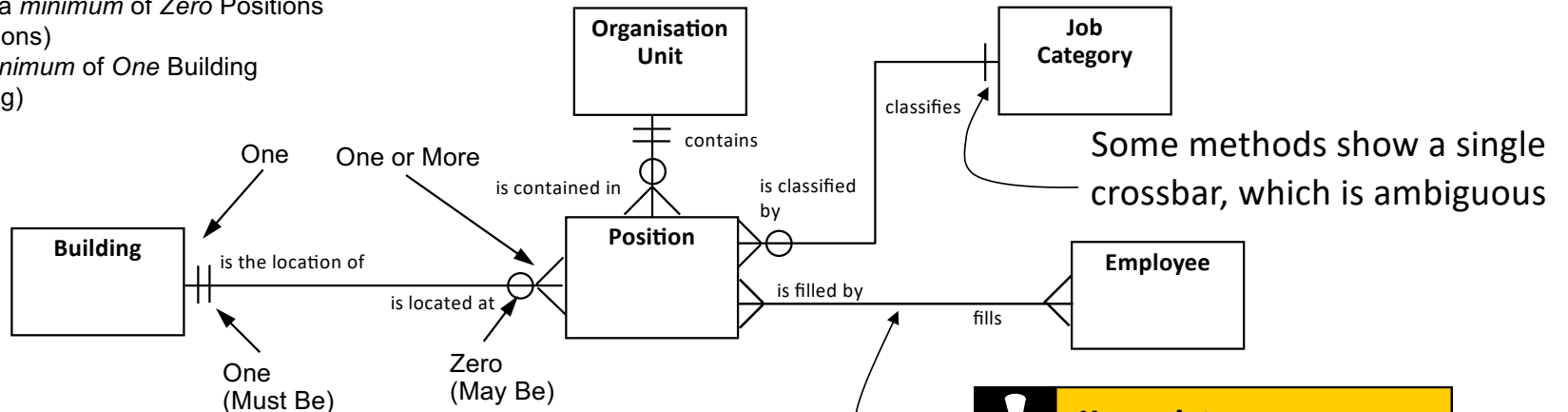| Project	—— is charged to —— funds charges for	G/L Account

- ✖ Failing to account for changes over time
  e.g., an Employee may hold only *one* Credit Card at a time, but *many over time,*
  and we virtually always want history.
  The most common written constraint
  in Concept Modelling is
  "one <u>at</u> a time but many <u>over</u> time."

Employee	—— holds —— is held by	Corporate Credit Card

72

Relationship optionality (logical models only)

Each Building <u>is the location of</u> a *minimum* of *Zero* Positions
(and a *maximum* of *Many* Positions)
Each Position <u>is located at</u> a *minimum* of *One* Building
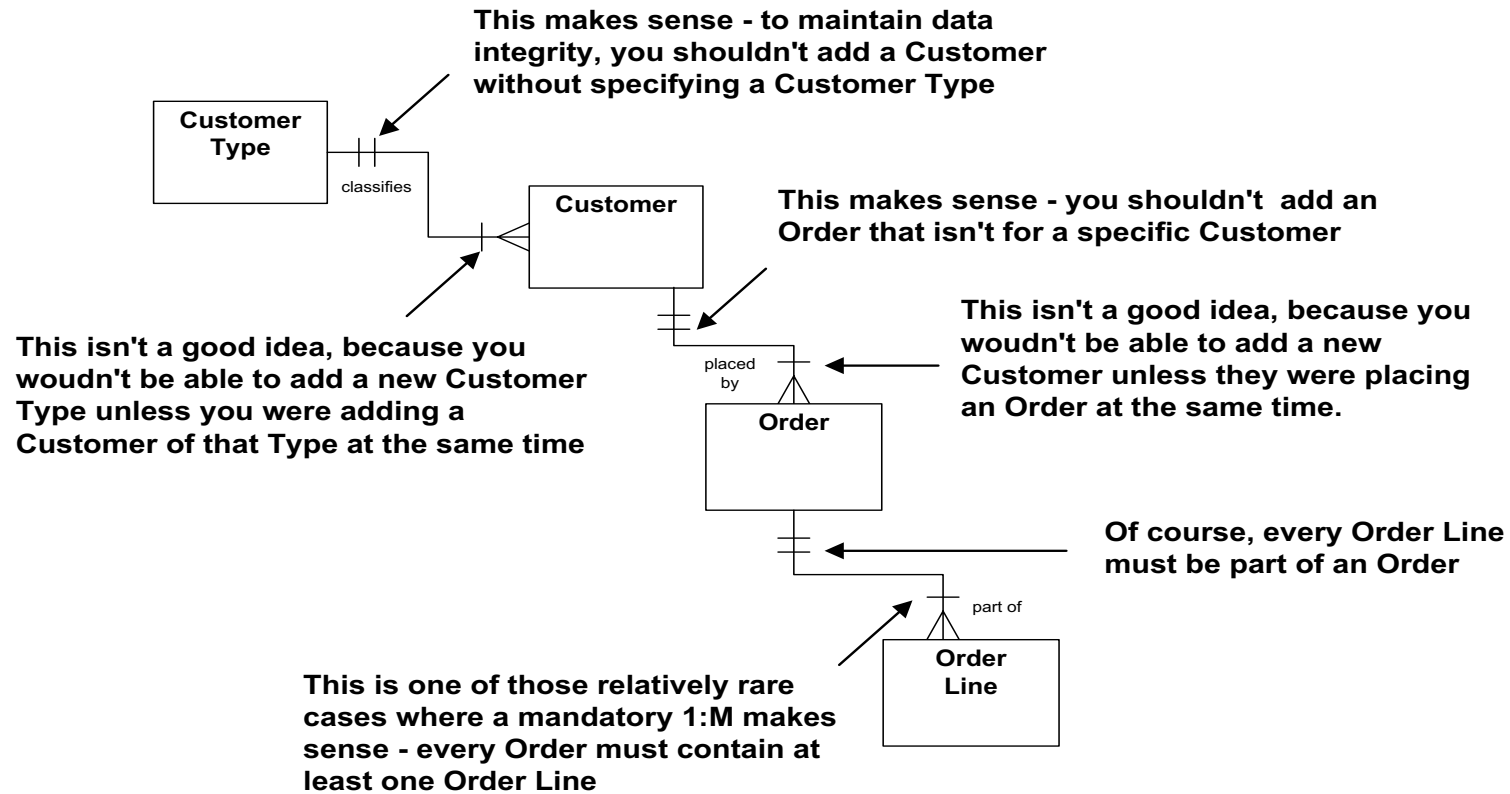(and a *maximum* of *One* Building)

Organisation Unit

Job Category

contains

classifies

is contained in

is classified by

Some methods show a single crossbar, which is ambiguous

One

One or More

Building

Position

Employee

is the location of

is located at

is filled by

fills

One
(Must Be)

Zero
(May Be)

or,
Each Building *May Be* <u>the location of</u> *One or More* Positions
Each Position *Must Be* <u>located at</u> *One* Building

!	**Key point**
	Typically only shown in logical data models – don't bother with optionality on M:Ms

To determine optionality (a.k.a. minimum cardinality)

- for each entity ask "Can one of these be related to a *minimum* of *Zero* <u>or</u> a *minimum* of *One* of the other entity?"

- record the answer – 0 <u>or</u> 1 – at the "other" end
  "zero" means an optional relationship (*May Be*) and
  "one" means a mandatory relationship (*Must Be*)

- easier form: "Each one of these *May Be* be <u>or</u> *Must Be* related to the other?"

73

Mandatory relationships - caution!

This makes sense - to maintain data integrity, you shouldn't add a Customer without specifying a Customer Type

Customer Type

classifies

Customer

This makes sense - you shouldn't add an Order that isn't for a specific Customer

This isn't a good idea, because you woudn't be able to add a new Customer Type unless you were adding a Customer of that Type at the same time

placed by

Order

This isn't a good idea, because you woudn't be able to add a new Customer unless they were placing an Order at the same time.

Of course, every Order Line must be part of an Order

part of

Order Line

This is one of those relatively rare cases where a mandatory 1:M makes sense - every Order must contain at least one Order Line

Don't forget the four Ds of Data Modelling

1 Definition

- "What *is* one of these things?"
- List common and unusual instances
- "Are there any known anomalies?"
- "What are the potential differences of opinion?"

2 Dependency

- "What type of entity is this?"
- "What other entity does it depend on?"
- Essentially
  - is it a free-standing thing?,
  - is it a type of thing?,
  - is it repeating detail about some other thing?

3 Detail

- Don't dive into detail – keep it in its place!
- GEFN!* HPDL!**

  Good enough for now!
  **Hard part, do later!*

4 Demonstration

- Assertions / narrative rules
- Sample data values or instances
- Scenarios or use cases
- Props (e.g., report layouts or common documents)

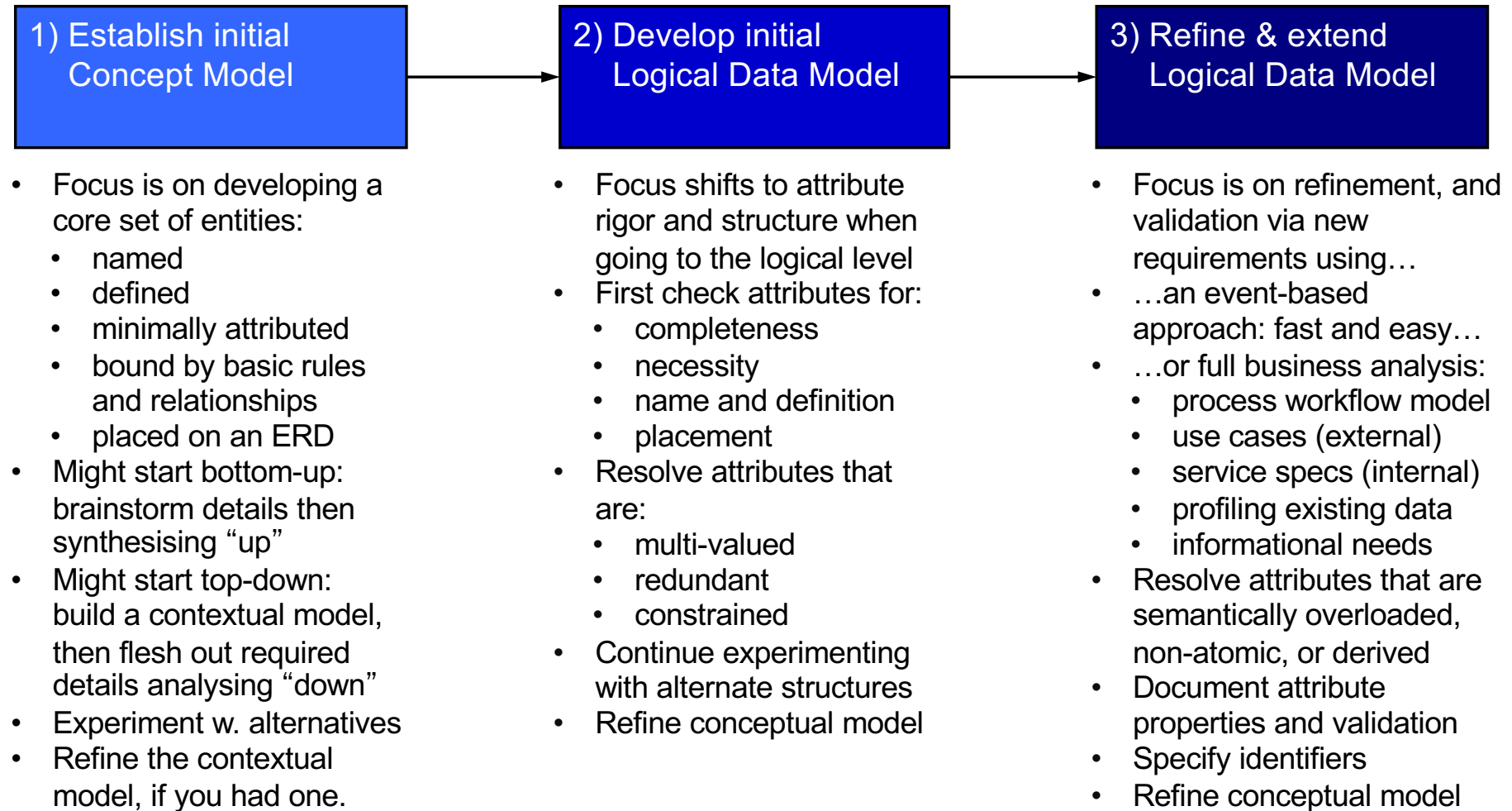# *A blank page to help maintain balance in the universe*

Adding rigor, structure, and detail

→	*Outline*
1.	Essentials of data modelling
2.	Adding rigor, structure, & detail

★	*Topics*

- Exercise – adding detail to the conceptual data model
- Entity types
- Guidelines for drawing the ER Diagram
- Attribute migration and guidelines
- A procedure for meeting new requirements
- The world's shortest course on normalisation
- Rules and guidelines for relationships
- Adding primary identifiers to the model

This is mostly for your reference

Phase 2 of three phases in data modelling

1) Establish initial Concept Model	→	2) Develop initial Logical Data Model	→	3) Refine & extend Logical Data Model

1) Establish initial Concept Model

- Focus is on developing a core set of entities:
  - named
  - defined
  - minimally attributed
  - bound by basic rules and relationships
  - placed on an ERD
- Might start bottom-up: brainstorm details then synthesising "up"
- Might start top-down: build a contextual model, then flesh out required details analysing "down"
- Experiment w. alternatives
- Refine the contextual model, if you had one.

2) Develop initial Logical Data Model

- Focus shifts to attribute rigor and structure when going to the logical level
- First check attributes for:
  - completeness
  - necessity
  - name and definition
  - placement
- Resolve attributes that are:
  - multi-valued
  - redundant
  - constrained
- Continue experimenting with alternate structures
- Refine conceptual model

3) Refine & extend Logical Data Model

- Focus is on refinement, and validation via new requirements using…
- …an event-based approach: fast and easy…
- …or full business analysis:
  - process workflow model
  - use cases (external)
  - service specs (internal)
  - profiling existing data
  - informational needs
- Resolve attributes that are semantically overloaded, non-atomic, or derived
- Document attribute properties and validation
- Specify identifiers
- Refine conceptual model

From conceptual to initial logical

The progression from conceptual to logical is largely based on identifying and dealing with three attribute characteristics

- Multi-valued - the attribute can have multiple different values for one instance of the entity, either "at a time" or "over time"
  E.g., "Employee Name" if aliases or previous names are tracked

  - move it **down** to the "many" end of a 1:M relationship into a characteristic entity

  - if it's a fact about a M:M relationship between entities, move it down to the "many" end of a 1:M relationship into an associative entity

  - this puts the data structure into 1st Normal Form – 1NF

- Redundant - the same attribute value is recorded multiple times, in different entity instances, possibly inconsistently
  E.g., "Company Name" in a "Department" entity

  - move it **up** to the "one" end of a M:1 relationship
    to one of the parent (or higher) entities (2nd Normal Form – 2NF)

  - You might have to create a new parent entity where none existed before

- Constrained - a descriptive attribute needs to be restricted to a set of standard (or "allowable") values to improve integrity and reporting
  E.g., "Employee Type"

  - move it **out** to the "one" end of a M:1 relationship
    to a reference or other related entity (3rd Normal Form - 3NF)

79

One more Conceptual to Logical example, drawn top-down

Conceptual

Beginning the Logical model
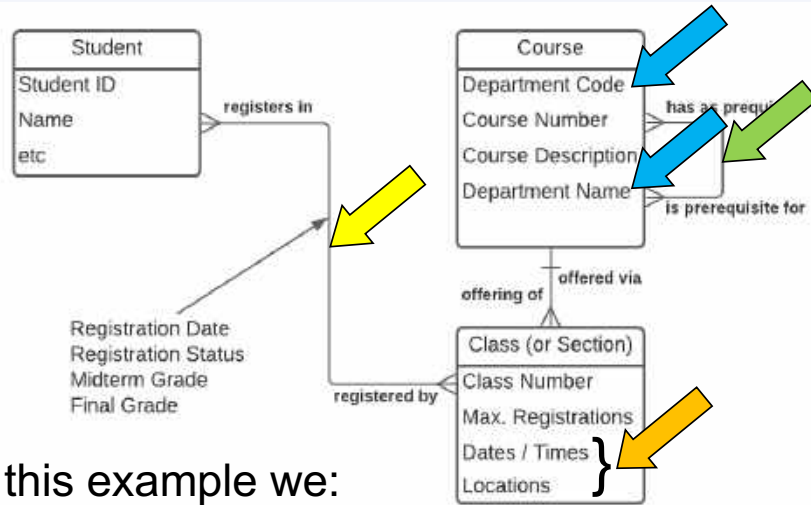
In this example we:

- move multi-valued Class attributes into their own entity – Class Lecture
- resolve the M:M relationship between Student and Class
- resolve the recursive Course to Course M:M relationship
- move redundant Department attributes in Course up into a new Department entity
- move Registration Status into a reference entity
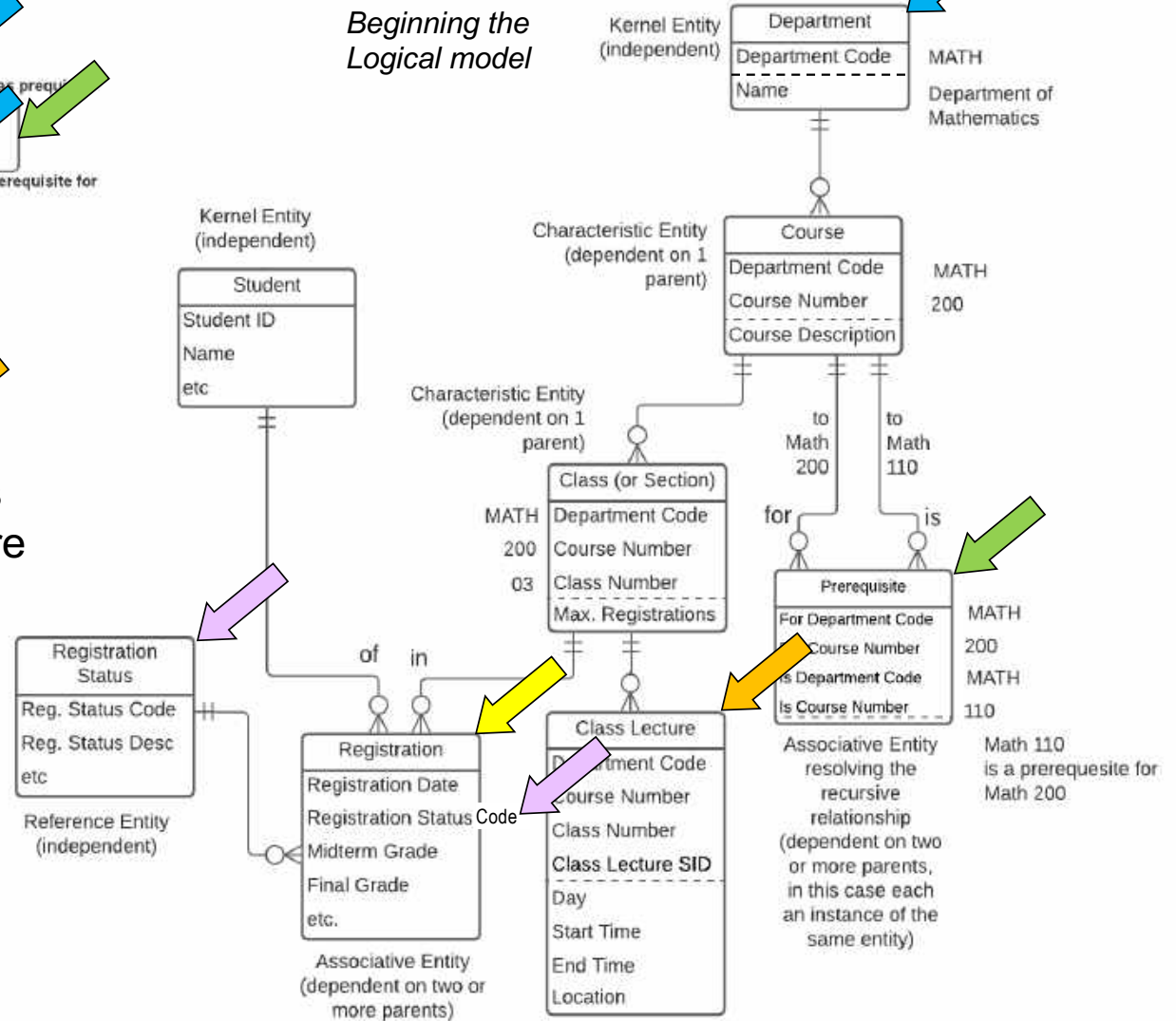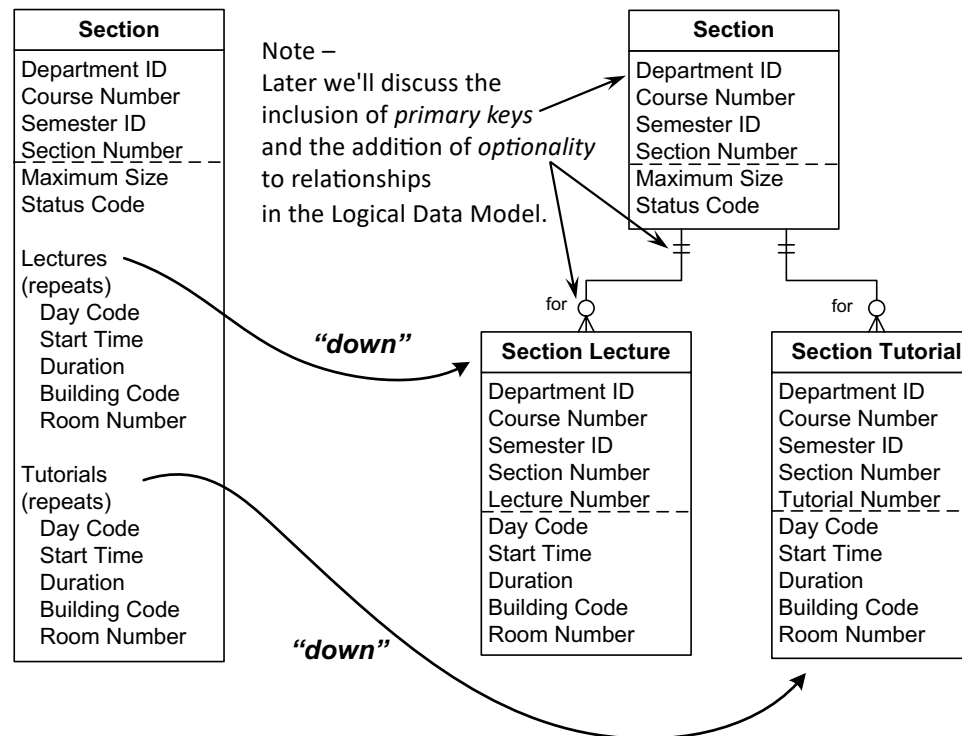
Migrating multi-valued attributes

Attributes can't repeat within an Entity in a Logical Data Model –

- "repeating" or "multi-valued" attributes are moved *down* into a characteristic entity
- this makes the data "reportable"

Section

Department ID
Course Number
Semester ID
Section Number _ _
Maximum Size
Status Code

Lectures
(repeats)
  Day Code
  Start Time
  Duration
  Building Code
  Room Number

Tutorials
(repeats)
  Day Code
  Start Time
  Duration
  Building Code
  Room Number

Note –
Later we'll discuss the
inclusion of *primary keys*
and the addition of *optionality*
to relationships
in the Logical Data Model.

Section

Department ID
Course Number
Semester ID
Section Number _ _
Maximum Size
Status Code

for

for

Section Lecture

Department ID
Course Number
Semester ID
Section Number
Lecture Number _ _
Day Code
Start Time
Duration
Building Code
Room Number

Section Tutorial

Department ID
Course Number
Semester ID
Section Number
Tutorial Number _ _
Day Code
Start Time
Duration
Building Code
Room Number

"down"

"down"

For each Section,
there may be one or more Lectures –
Lecture is a "repeating group"

For each Section,
there may be one or more Tutorials –
Tutorial is a "repeating group"

We must move each
"repeating group"
down into a child entity.

Migrating attributes of relationships

When the multi-valued attribute (or group of attributes) is a *fact about a relationship*, we create an associative entity:

Student
Student Number
Name
GPA

e.g., John Smith

enrolls in

Section
Department ID
Course Number
Semester ID
Section Number
Maximum Size
Status Code

e.g., Math 100, Section 3

Student
Student Number
Name
GPA

by

into

Section
Department ID
Course Number
Semester ID
Section Number
Maximum Size
Status Code

Enrollment
Student Number
Department ID
Course Number
Semester ID
Section Number
Status Code
Enrollment Date
Drop Date
Midterm Grade
Final Grade

"down"

"What date did John Smith enroll in Math 100?"
"What grade did John get in Math 100 at midterm?"
"What was John's final grade in Math 100?"
"What is the average grade for Math 100 Section 3?"

These required facts are not about Student or Section
but about the relationship between a Student and a Section

We need to create a new associative entity
and move the attributes *down* into it

Migrating redundant attributes

We eliminate redundancy by ensuring that every attribute is in the entity it describes,
so the attribute value is recorded only *once*.

- Before migration, attribute values about a Department would be recorded *redundantly*
  in every Course offered by that Department,
  so Department attributes are moved *up* to a new parent entity, Department.
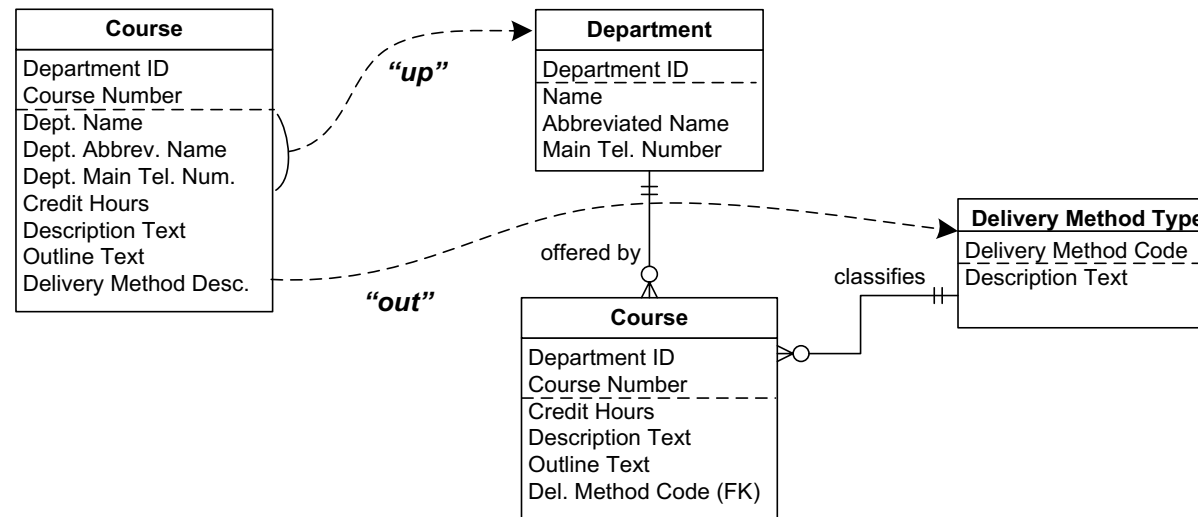
- Before migration, attribute values about a Delivery Method Description would be recorded *redundantly*
  in every instance of Course,
  so Delivery Method attributes are moved out to a new *reference* entity
  (or *type* or *lookup* or *classification* entity,) Delivery Method Type

World's shortest course on normalisation

Unnormalised (UNF or 0NF)

- Contains multivalued attributes (a "repeating group")

First Normal Form (1NF)

- Repeating attributes moved *down* to a dependent Characteristic or Associative entity (create a new dependent entity if necessary.) This makes data "reportable."

Second Normal Form (2NF)

- Only applies to dependent entities

- No attribute in a child entity is really a fact about a parent (or grandparent or...)

- That is, no Characteristic or Associative entity redundantly contains facts from its parent(s) – if it does, move the fact(s) *up* (create a new parent entity if necessary)

Third Normal Form (3NF)

- If any entity redundantly contains facts from a related (non-parent) entity, move the fact(s) *out* to the other entity (create a new entity if necessary)

BCNF (Boyce-Codd NF – "3.5NF")

- Not an issue if you keep your wits about you

Fourth and Fifth Normal Form (4NF, 5NF)

- "Large" (3-way or more) associatives need to be broken down into more granular entities

UNF

1NF

2NF

3NF

4NF,
5NF?...

84
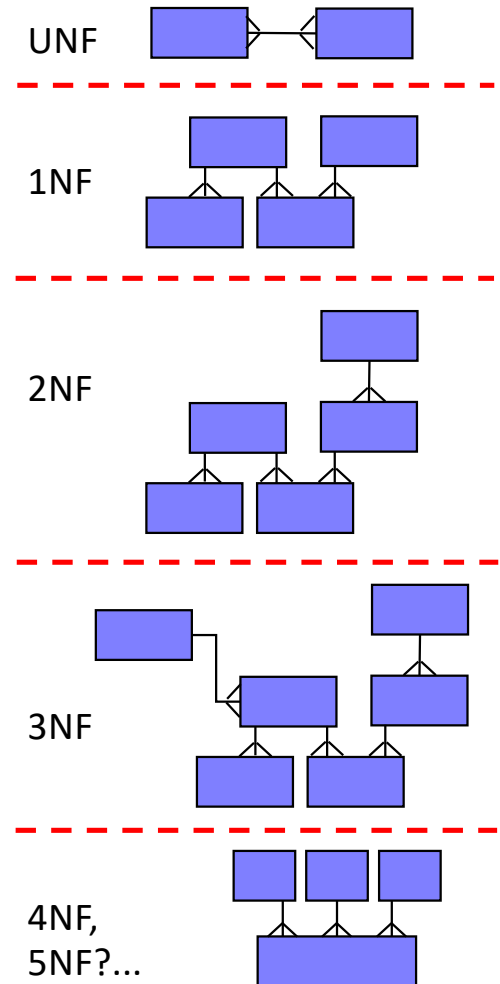
For review: specifics – contextual, conceptual, logical

1 Contextual (Scope)	**2** Conceptual (Overview)	**3** Logical (Detail)

Agree context or "big picture" – the scope in terms of topics or subjects that are in or out, plus core terms and definitions

- May be a simple block diagram of topics/subjects, or primarily textual (a list)
- Optional – not necessary on smaller projects

My most plagiarised diagram ever!

Agreement on basic concepts and rules

- Ensures everyone is using the same vocabulary and concepts before diving into detail
- Overview: main entities, attributes, relationships, rules
- Lots of M:M relationships
- Relationships show cardinality
- No keys
- Few or no reference entities
- Unnormalised – most M:M relationships unresolved, many attributes will be multi-valued, redundant, and non-atomic
- Verified directly by clients plus other techniques: Use Cases...
- A "one-pager"
- 20% of the modelling effort

Full detail for physical design

- Provides all detail for initial physical database design and requirements specification
- Detailed: ~ 5 times as many entities as the conceptual model
- M:M relationships resolved
- Relationship optionality added
- Primary, foreign, alternate keys
- Lots of reference entities
- Fully normalised – no multi-valued, redundant, or non-atomic attributes. All attributes defined and "propertised"
- Verified by other means: sample data, report mockups, scenarios, ...
- May be partitioned
- 80% of the modelling effort

85

Exercise 2 – Adding rigor and structure to the data model

The assignment:
Re-read the scenario, and
- ensure every attribute mentioned in the scenario is placed on the data model, in the correct entity
- identify any attributes or sets of attributes that will repeat within an entity (i.e., be multi-valued)
- identify any attributes whose values will be recorded redundantly, or attributes whose allowable values need to be standardised
(to avoid redundancy / inconsistency)

The instructor will lead the refinement of the data model on the whiteboard.

Amalgamated Automaton… (background text omitted) …has no base of historical data to aid in trend analysis or estimating development effort, nor any effective means of charging back development costs. The proposed solution is to develop a simple Project Tracking System, which will work in conjunction with the existing Personnel and General Ledger Systems.

When a development project is initiated, a project name and a short description are recorded, among other things. Before any further work is done on the project, a new account is created on the G/L System, identified by a G/L account number. Project costs will be charged to this account, and the project budget is recorded as the initial account balance.

Project planners break a project down into many tasks, perhaps hundreds. A typical project task might be "Test Order Entry Module". Some of the facts which are required about tasks include a brief task description, estimated work hours, and the scheduled start and finish dates.

Eventually, individual workers are assigned responsibility for the tasks. Some tasks will be the responsibility of many workers, and a worker might be assigned to many tasks. As each worker is assigned to a project task, their planned start and finish dates, their contribution to the task (not a "kind of work," but their specific duties on the task – e.g., "Develop test scripts"), and the estimated number of hours they are to spend on the task are recorded. Worker information such as the worker name and number are available from the existing Personnel System, although it will have to be modified to record the worker's hourly charge out rate, and the type of worker (contractor, regular, student intern, etc.)

When an IS worker begins work on a new task, their actual start date is recorded. A running total of the number of hours that they have worked on each started task is updated regularly. At the same time, the remaining balance in the project account is updated. When a worker completes a task assignment, the actual completion date is recorded.
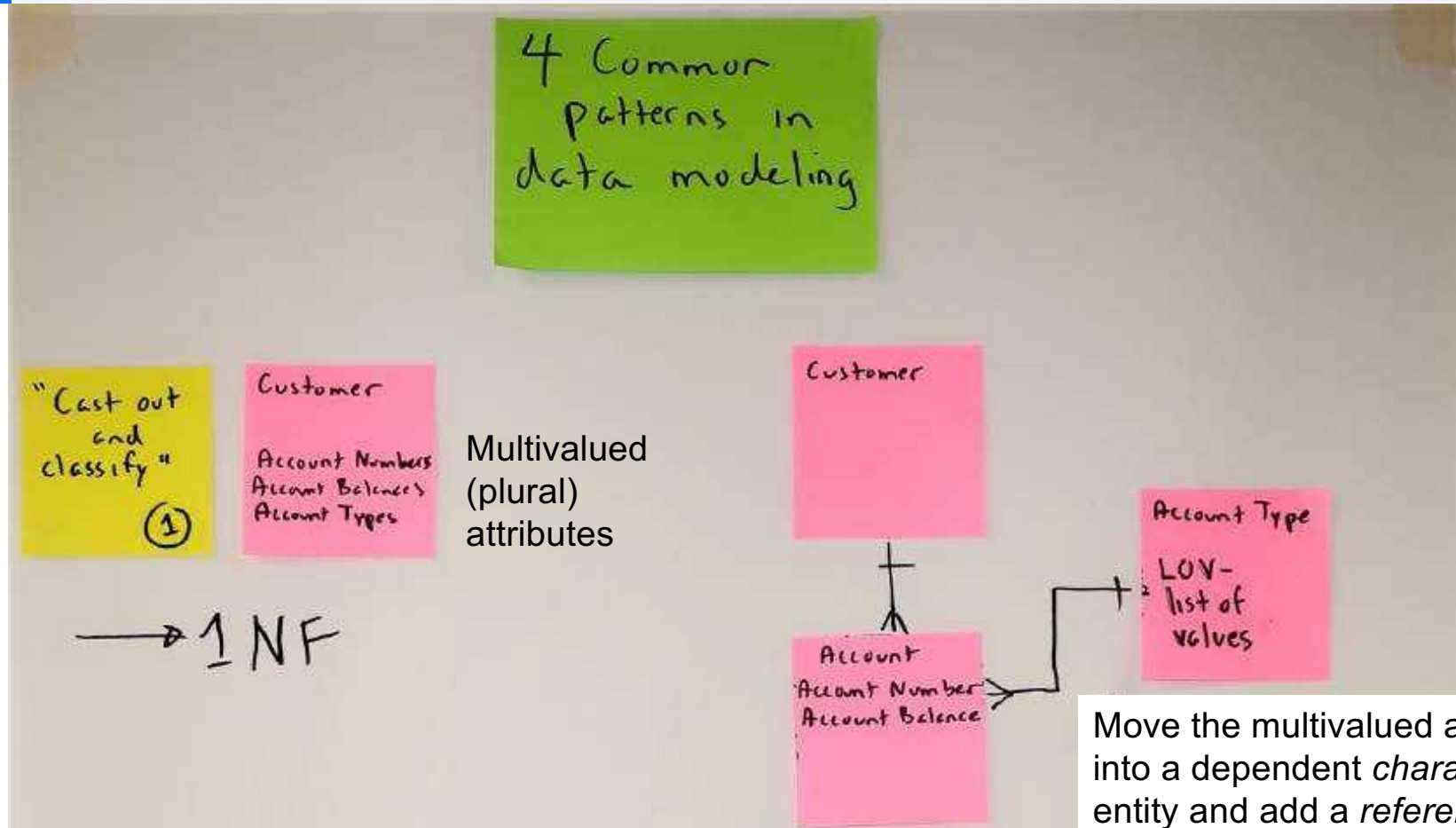
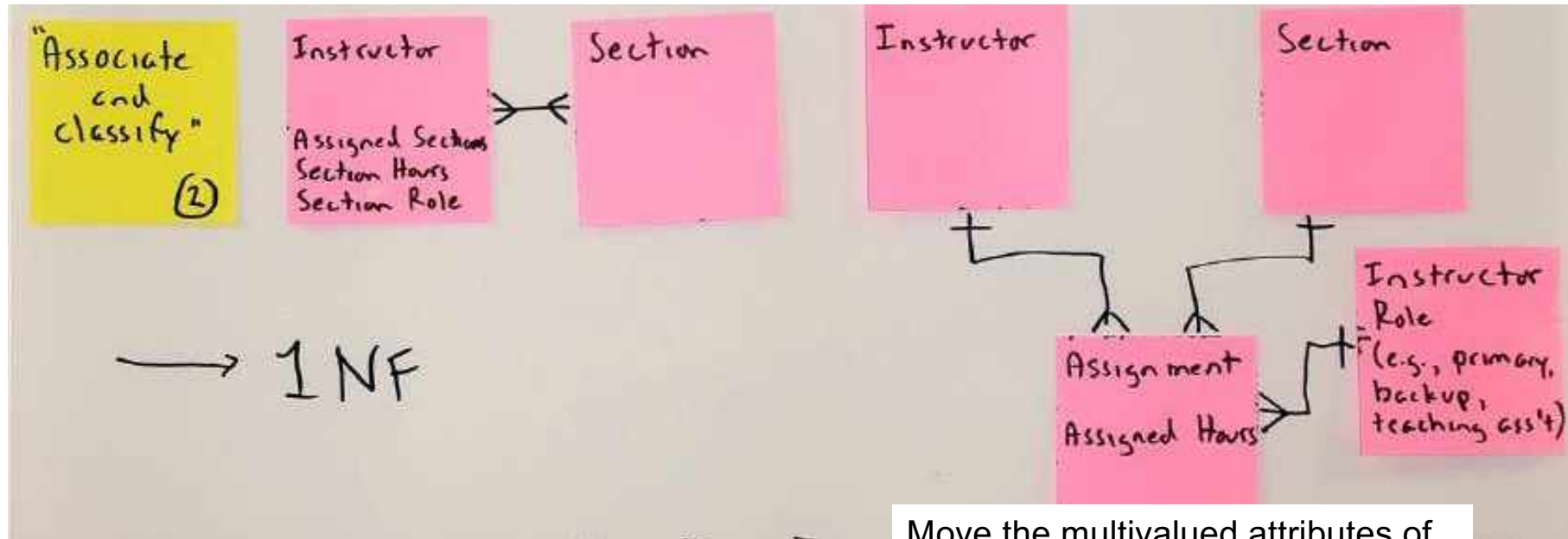# *Exercise 2 – solution (logical, but only core entities)*

Reference (or Type)

Worker Type

Worker Type Code
Description

— describes —

Kernel

Worker

Worker ID
Name
Hourly Charge Rate Amount
Worker Type Code (fk, nn)

Kernel

Project

Project ID
Description
Name
Account Number (fk, nu)

is charged to

collects
charges
from

Kernel

G/L Account

G/L Account Number
Initial Balance Amount
Remaining Balance Amount
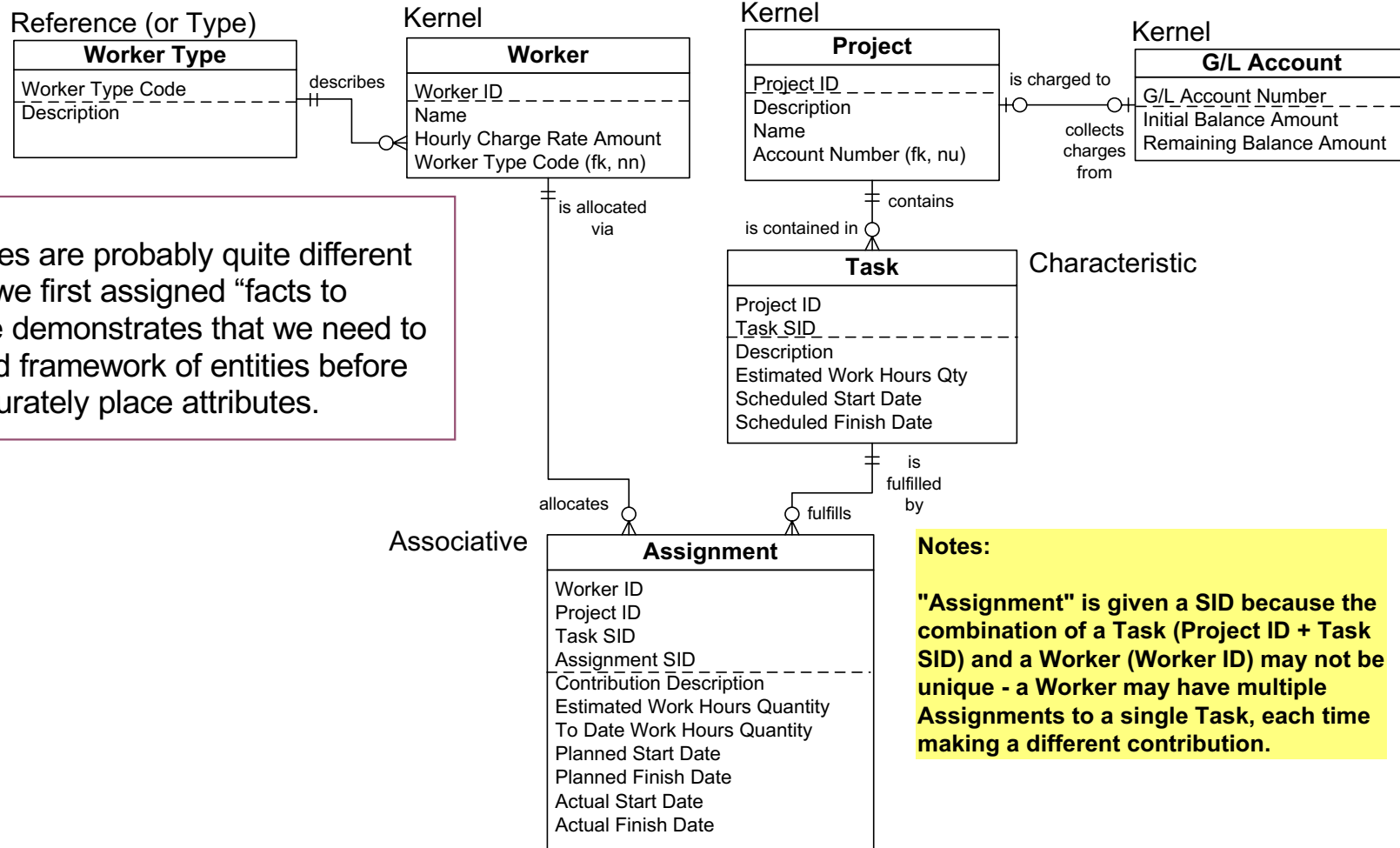
The point -
The attributes are probably quite different than when we first assigned "facts to things." The demonstrates that we need to have a good framework of entities before we can accurately place attributes.

is allocated
via

is contained in

contains

Characteristic

Task

Project ID
Task SID
Description
Estimated Work Hours Qty
Scheduled Start Date
Scheduled Finish Date

is
fulfilled
by

allocates

fulfills

Associative

Assignment

Worker ID
Project ID
Task SID
Assignment SID
Contribution Description
Estimated Work Hours Quantity
To Date Work Hours Quantity
Planned Start Date
Planned Finish Date
Actual Start Date
Actual Finish Date

Notes:

"Assignment" is given a SID because the combination of a Task (Project ID + Task SID) and a Worker (Worker ID) may not be unique - a Worker may have multiple Assignments to a single Task, each time making a different contribution.

87

Four patterns: 1 – "Cast out and classify"

Multivalued (plural) attributes

Move the multivalued attributes into a dependent *characteristic* entity and add a *reference* entity – "cast out and classify."

Four patterns: 2 – "Associate and classify"

Move the multivalued attributes of the M:M relationship into a dependent *associative* entity and add a *reference* entity – "associate and classify."

Four patterns 3 – "Generalise and classify"

Could be *Many*
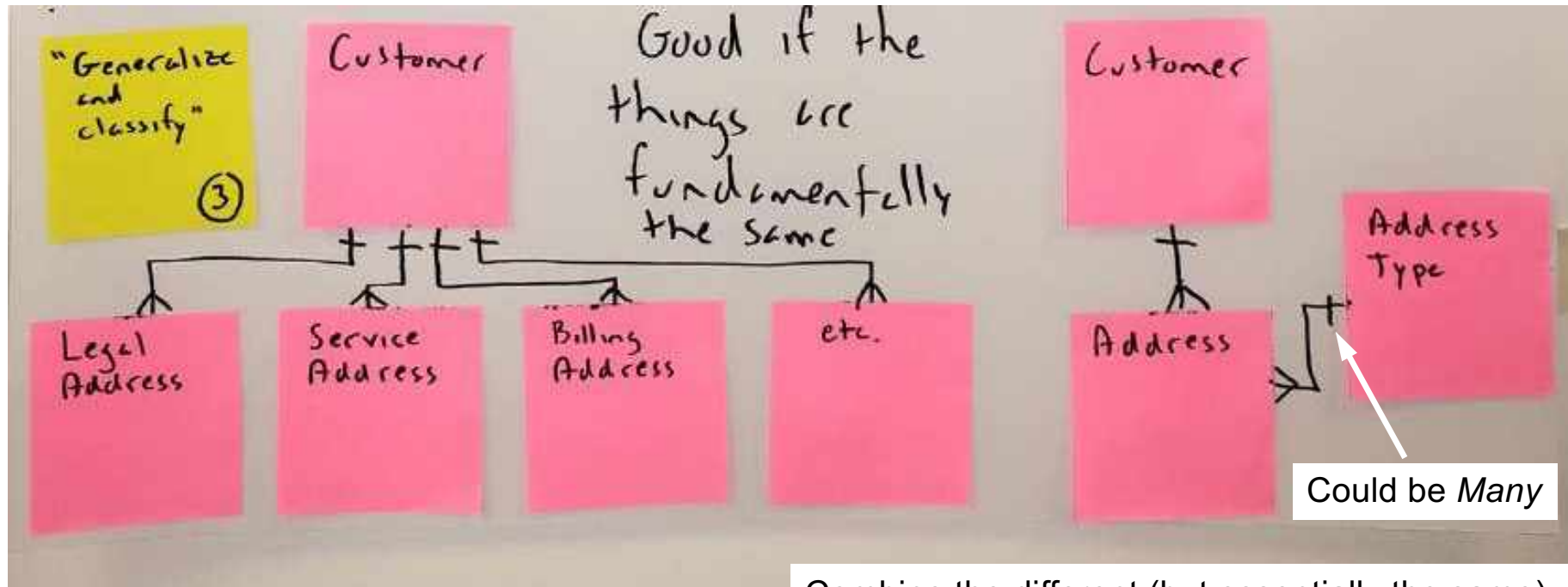
Combine the different (but essentially the same) entities into a generalised entity (in this case a *characteristic)* and add a *reference* entity – "generalise and classify."

Four patterns 4 – "Generalise, recur, and classify"
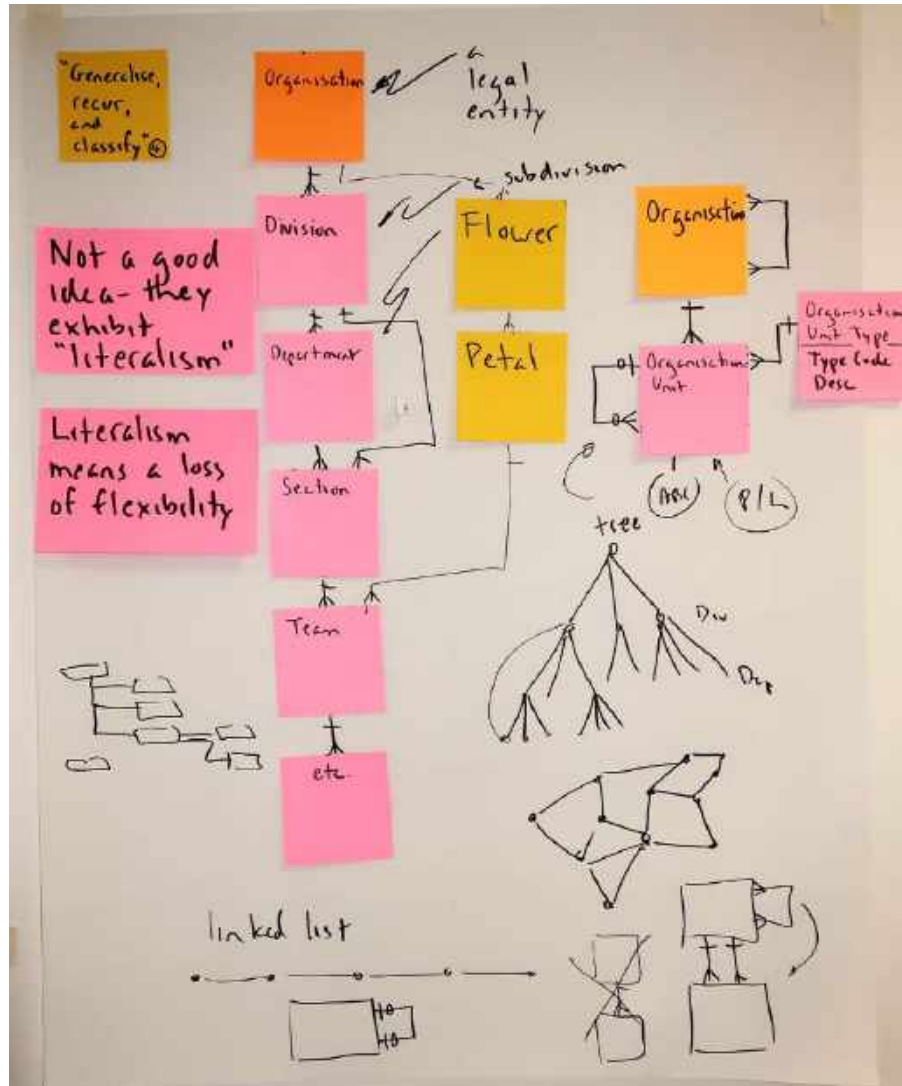
Combine the different (but essentially the same) entities in the *hierarchy* into the generalised entity Organisation Unit (in this case a *characteristic* of the Organisation), then model the hierarchy by adding a recursive relationship, then add a *reference* entity – "generalise, recur, and classify."

Future-proofing – "Challenge the Ones"
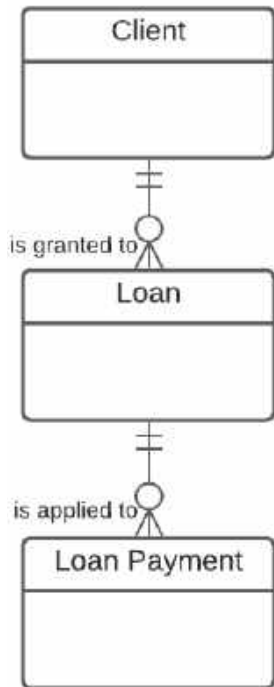
A Loan is granted to
one and only one Customer –
really?

No – multiple Clients can participate
in a Loan via a shared Account.
(A new *requirement*.)

A Loan Payment applies to
one and only one Loan –
really?

No – one Loan Payment could be
distributed across multiple Loans.
(A new *requirement*.)

This M:M would
also be resolved.

repaid via

distributed
to

M:M resolves to a
new associative
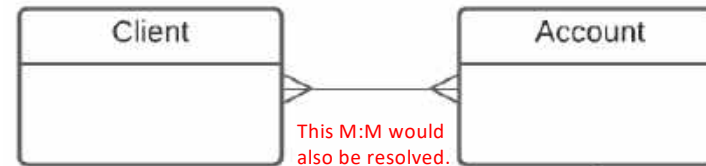
This revised model
meets the new
requirements.
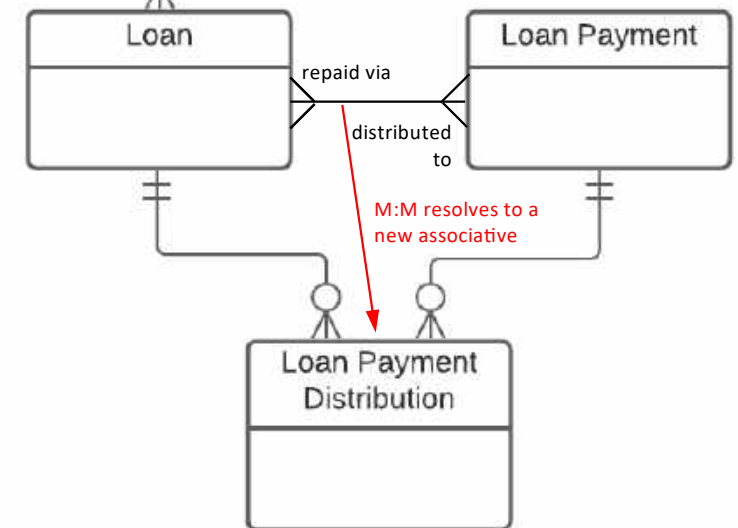
Future-proofing – "Avoid fixed hierarchies"

If we implement this model, what will go wrong?

Company
Division
Department
Section
Unit
Etc.

New types of Organisation Unit will arise

Company
Division — Flower
Department — Petal
Section
Squad
Chapter
Guild
Unit
Tribe
Etc.

New connections will appear

Organisation
Organisation ID
Name
Inception Date

is divided into
is part of

controls
is controlled by

Organisation Unit
Organisation ID
Organisation Unit SID
Parent Organisation ID (FK)
Parent Organisation Unit SID (FK)
Organisation Unit Type Code (FK)

classifies
is classified as

Organisation Unit Type
Org. Unit Type Code
Description

This revised model is *far* more flexible.

This is an example of the "generalise, recur, and classify" pattern.

93

Self-study exercise 3 – from conceptual to logical
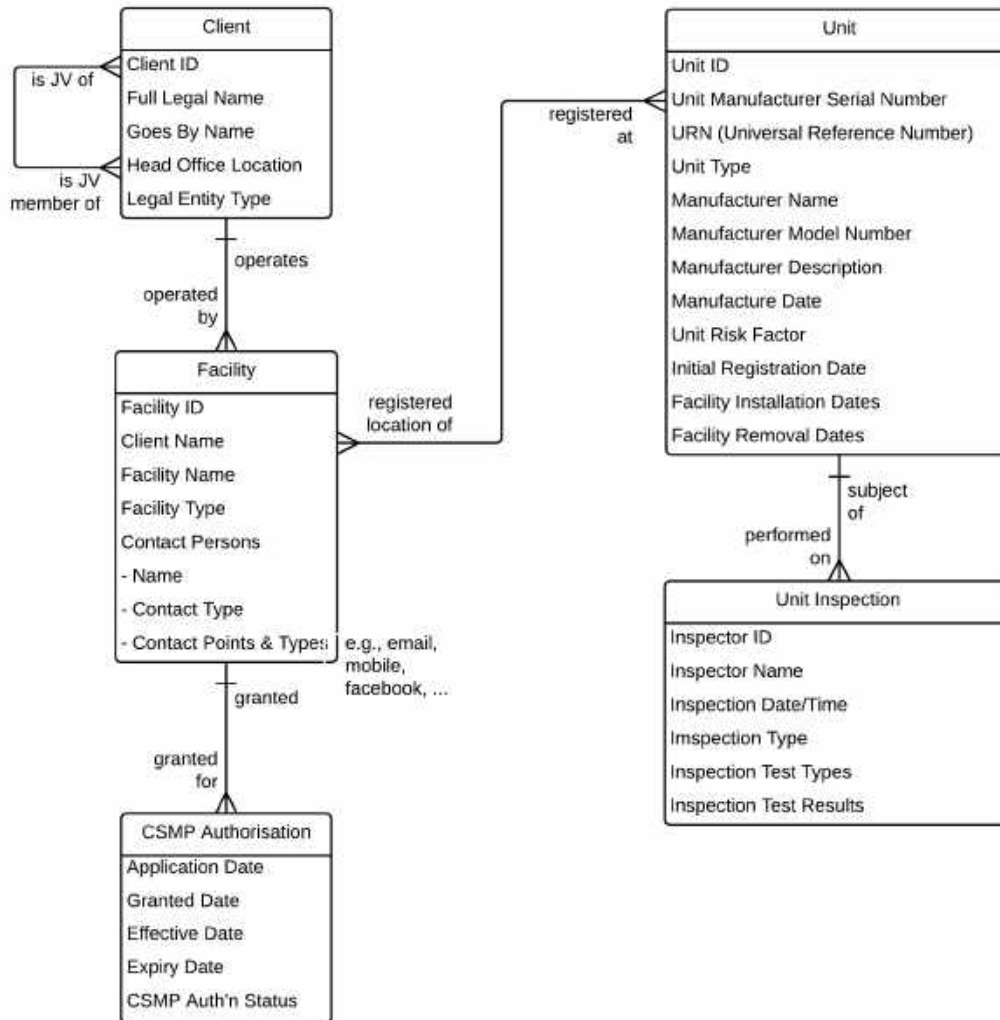
This is unnormalised – it contains multi-valued (repeating,) redundant, and constrained attributes.

First, identify the attributes that are "correct" – they are base attributes of the entity they are in. ✔
Then, normalise it to 3NF (Third Normal Form) by identifying and dealing with attributes that are:

- Multivalued, and need to be moved *down* to a dependent entity. (1NF)

- Redundant and need to be moved *up* to a parent (or higher) entity. (2NF)

- Redundant or constrained and need to be moved *out* (sideways) to a related but non-parent entity, or to a reference entity. (3NF)

94

Entity types – kernels

Building	Organisation Unit	Job	Employee

- The "central" objects in the model
  - "what it's all about"
  - everything else either further describes, associates, or classifies the kernel entities

- *Independent* – its existence is not dependent on another entity
  - "Does it make sense for one of these to exist on its own?"
  - is not a child of another entity

- Ideally, the starting point for modelling

- Drawn at the top of the diagram, or subject area within a diagram

- Primary identifier – a meaningless, system-generated number called "entity name ID"

Entity types – characteristics

Building — Room
A characteristic of a Kernel

Room — Workstation
A characteristic of another Characteristic

Job — Job Qualification, Job Responsibility

Position, Employee — Assignment — Special Duty

A characteristic of an Associative

Drawn below the characterised (parent) entity
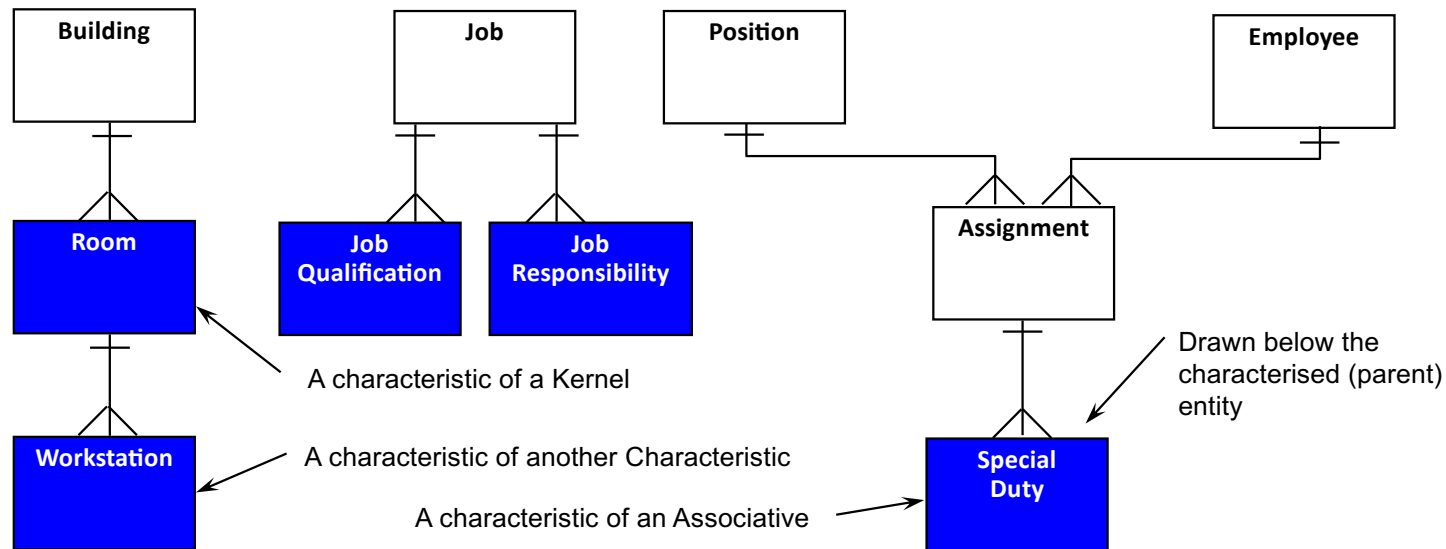
- Records repeating, multi-valued facts about a parent entity that have been "cast out" from the parent entity

- It "characterises" the parent entity (any type of entity)

- *Dependent* on one parent entity, and is drawn below that parent

- Primary identifier – the inherited key of its parent plus a meaningless, system-generated short numeric string called "entity name SID"

Entity types – associatives

Position   **Employee**

filled by   assigned via

fills   assigns

Assignment

Drawn between and below the
associated (parent) entities

Naming the relationships from
an associative is often difficult,
because the associative entity
itself is a relationship
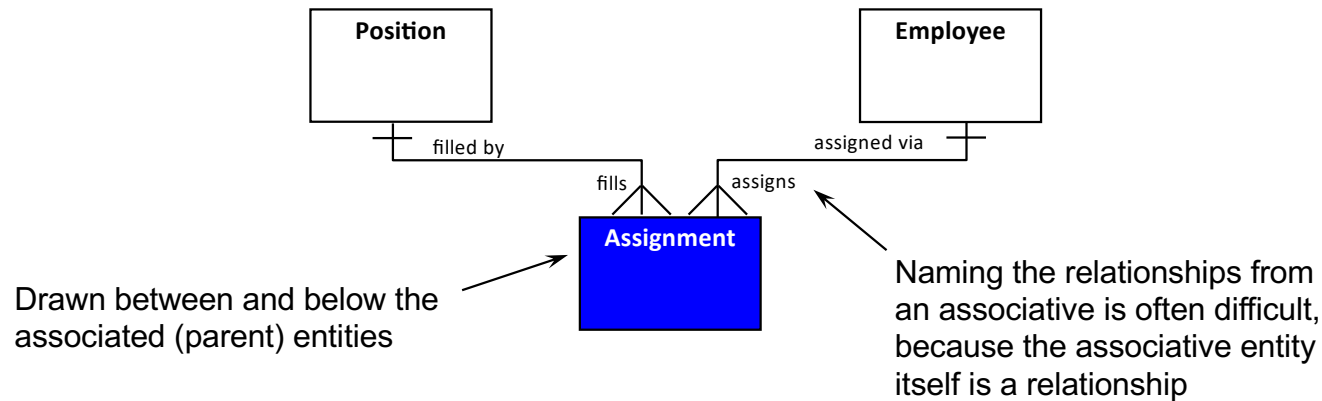
- Relates ("associates") two or more other entities –
  records facts about the association (M:M relationship) between those other entities
  - sometimes so important it is discovered directly (Order, Contract, …) and is shown on the
    Conceptual Model – the remainder are added on the Logical Model
  - other times it evolves from "resolving" M:M relationships
- Can associate any combination of different entity types
- *Dependent* on two or more parent entities, and is drawn between and below the parents
- Primary identifier – the inherited keys of all parent plus an SID if needed
  (if the same parent instances could be associated more than once)

Associatives – notes

Benefit Program —— enrolled by / enrolls in —— **Employee**
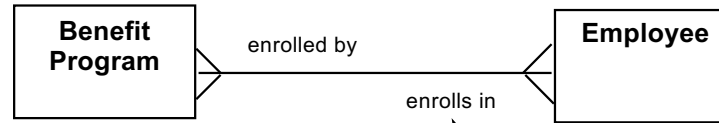
Where will we record facts about the relationship, such as Enrollment Date?

becomes...

The M:M relationship name is often the basis for the associative entity name

Benefit Program — is enrolled through / defines participation in → **Benefit Program Enrollment** ← defines participation by / enrolls through — **Employee**
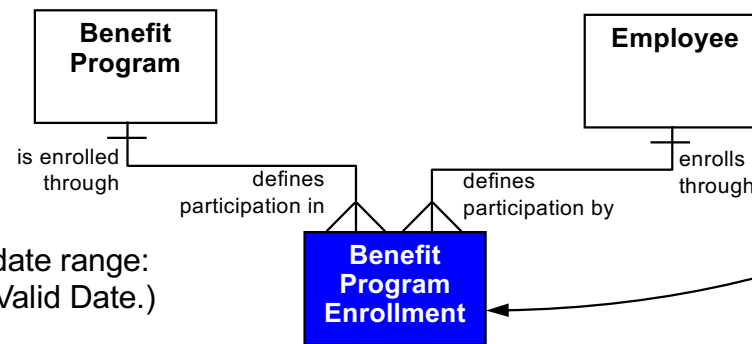
Associative entities often contain a date range:
Effective Date to End Date (or Last Valid Date.)
Always check for the following:

- Gaps are allowed?

| Effective Date | End Date | | Effective Date | End Date |

- Must be contiguous?

| Effective Date | End Date | Effective Date | End Date |

- Overlaps are allowed?

| Effective Date | End Date |
| Effective Date | End Date |

- Effective Dates can begin in the past or future?

- Is the date range *until* or *through* the End Date? (tot en met)

- Must an End Date be specified, and if so, what format is used – "null" or "HighDate – 99991231?"

- Must the date range fit within a parent's date range?

- Do global time zones need to be handled?

98

Entity types – reference or type

"Reference" (or "Type" or "Classification" or "Specification" or...) entities

classifies — **Vehicle Make-Model**

categorises — **Customer Type**

describes — **Job Category**

Vehicle — is a

**Customer** — is a

Job — belongs to
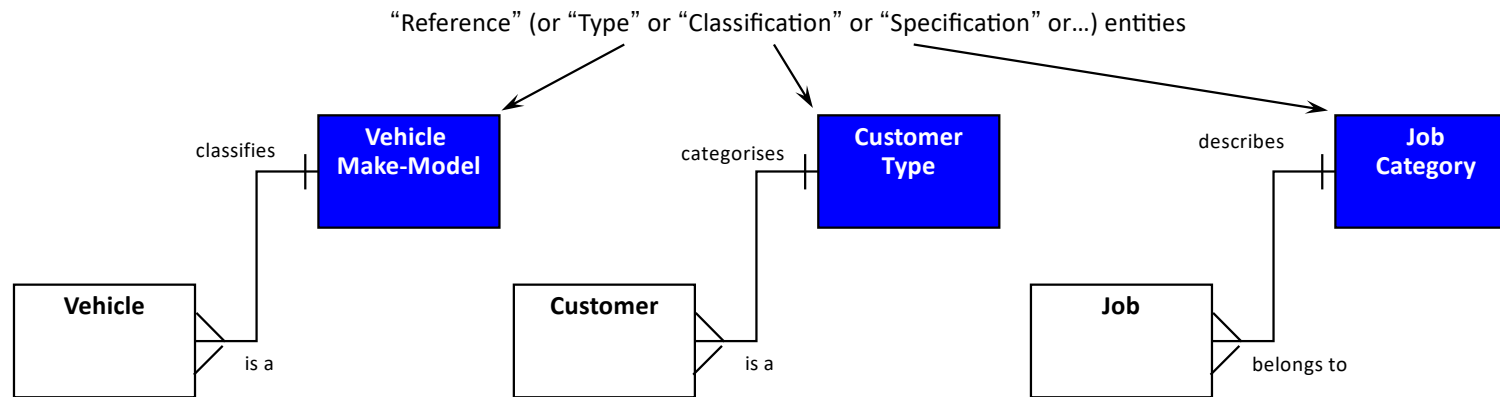
- An entity that classifies or categorises other entities and/or allows the recording of standardised values for a descriptive attribute

- *Independent*

- Drawn beside or diagonally up from the classified entity

- Purpose may be served by an attribute in the Concept Model
  (i.e., a Customer Type attribute in the Customer entity)

- Only critical Reference entities are shown on a Concept Model
  (i.e., when a Reference entity ties together different parts of the model)

- Primary identifier – often, a mnemonic (recognisable) code; otherwise, a meaningless ID

Consistency is a virtue

People pick up data modelling without training if you…

- treat it as a natural way to describe a business,
  not a new technique being imposed on them

- draw the same kinds of things the same way every time
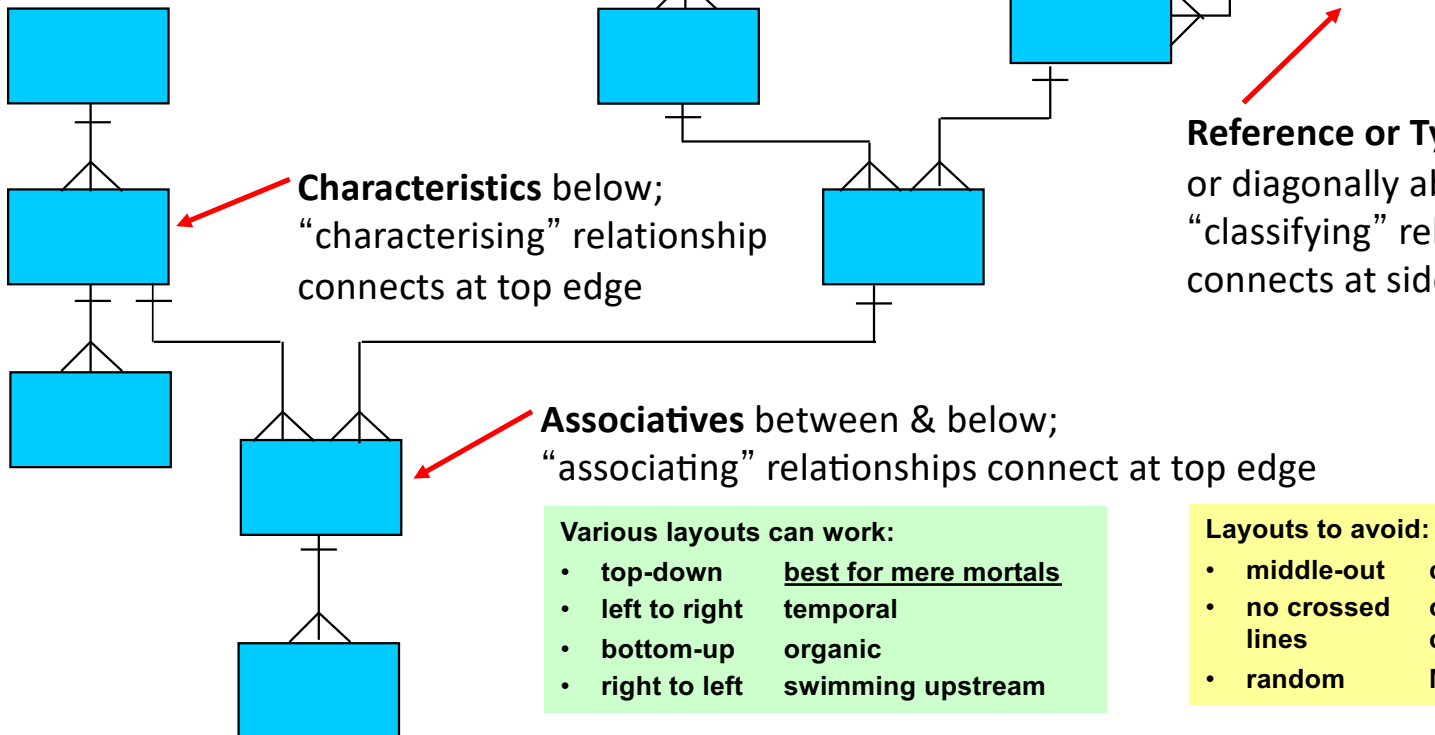
E.g., when drawing an associative entity…
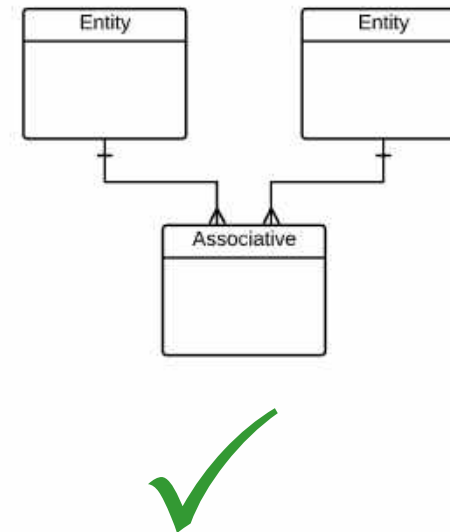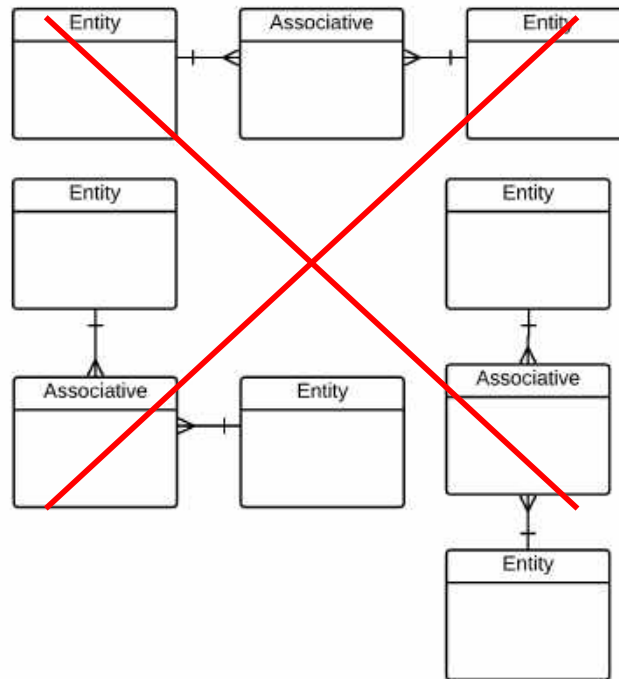
Graphic guidelines – the "no dead crows" principle

!	*Key point*

Entity type is obvious from:
- Placement
- Relationship connections

Kernels across the upper part of the diagram or subject area

Characteristics below; "characterising" relationship connects at top edge

Reference or Types beside or diagonally above; "classifying" relationship connects at side

Associatives between & below; "associating" relationships connect at top edge

Various layouts can work:
- top-down — best for mere mortals
- left to right — temporal
- bottom-up — organic
- right to left — swimming upstream

Layouts to avoid:
- middle-out — cosmic
- no crossed lines — obsessive / compulsive
- random — Mensa-only

101

Summary – entity types and conventions

Kernel
Independent
A fundamental thing of interest to the
enterprise whose existence does not depend
on any other entity – it can "stand alone"
Drawn at the top of its area

Supertype
Contains facts (attributes
and relationships) that are
common to all instances of
the entity. Any kind of
entity can be a supertype.

Business
Party

is either

is a type of
is a type of

Person
Organisation

Building
Type

Building

classified as

Reference or Type
Independent
Classifies or categorises other
entities and/or allows the
recording of allowable values
for a descriptive attribute
*Drawn diagonally out from or
beside the classified entity*

Subtype
Contains facts that are
specific to a particular
subset of instances of
the entity.

part of

Floor

by
of

Occupancy

part of

Area

contains

Characteristic
Dependent on one parent
Records multi-valued facts
about a parent entity that
have been "cast out" from
that entity
Drawn below parent

Associative
Dependent on two or more parents
Records facts about a relationship
(association) between two or more
parent entities – is often the
resolution of a M:M relationship
between the parents
Drawn between and below parents

Recursive relationship
A relationship between
instances of the same entity.
Can be 1:1, 1:M, or M:M

102

(For reference) Primary keys – essential concepts

What they are…	What they are not…

• One or more attributes with a unique value for each instance of an entity
 	
• There might be many identifiers – one is chosen as the primary identifier, the rest are alternate identifiers
 	
• A way to reference an entity instance (e.g., a row of a table)
 	
• Used to establish relationships between entities (or tables) – the primary key of an entity included in another entity is called a "foreign key" and establishes a M:1 relationship
	• The only access or search path
	
	• The fundamental way the business distinguishes:
	  • one instance from an other
	  • a new instance from existing (e.g., Customer applying for credit)

In short, how we relate entities (tables) is not necessarily how the client distinguishes or accesses them

Customer:
Possible keys:
• Customer Name + Postal Code
• Sales Region + Customer Number
• Account Number

Part:
Possible keys:
• Part Category + Manufacturer Prod #

Employee:
Possible keys:
• SIN or SSN
• Name + Address
• Name + Birthdate
• Portrait + Voice

Reservation:
Possible key:
• Room Number + Start Date

103

Meaningless primary keys

Essential characteristics

stable (unchanging)	available
• under your control	• known, or can be assigned, at instance creation
• contains no meaningful data, because it will eventually change	
• contains no "special values" like Customer Number 9999999	
• 'key hierarchy' is unchanging when an inherited key is used as part of identifier	

- *Unique*
- *Non-null*
- Unchanging

A badly designed key:

"Employee Number" – 07X1492

07 – country code

X – contractor

1492 – sequence number

Almost invariably eliminates any choice except keys made up of

meaningless, system-generated ID or Secondary ID (SID) components

Customer:
• Customer ID
…is better than…
• Customer Name + Postal Code
• Sales Region + Customer Number
• Account Number

Part:
• Part ID
…is better than…
• Part Category + Manufacturer Prod #

Employee:
• Employee ID
…is better than…
• SIN
• Name + Address
• Name + Birthdate
• Portrait + Voice

Reservation:
• Reservation ID
…is better than…
• Room Number + Start Date

104

Examples – key not available, not static

Poorly chosen Vehicle key
e.g.,

Vehicle
License Plate Number
Manufacturer
Make/Model
Model Year

... instead of ...

Vehicle
Vehicle ID Number
Manufacturer
Make/Model
Model Year
License Plate Number
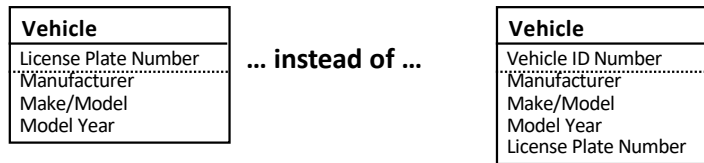
The problems
- Some License Plates move from Vehicle to Vehicle
- License Plate Number not always available,
  so "dummy" number created

Similarly, wouldn't build Owner ID into the Vehicle primary key
- Owners can change – wouldn't want to change primary key
- Some unregistered Vehicles don't have an Owner yet – can't have "null" in key

- Consequences of poorly chosen keys:

  - expense (hard to change key values)

  - errors (hidden meaning, lost relationships)

Foreign keys

- A primary key of one entity included within another entity is called a "foreign key."
  A foreign key is essentially a "pointer" –
  it is how relationships are implemented in a relational database.

- Two types:
  - part of primary key (inherited via an "identifying relationship")
  - not part of the primary key (added as an attribute)

- In a completed Logical Data Model, every relationship is supported by a Foreign Key.

- Which of the two entities holds the F.K.?
  - 1:M – the foreign key is in the entity at the "M" end, and "points to" the "1" end
  - 1:1 – either end, although it should go in the end where it is mandatory or will occur most often. (Rare –1:1 relationships are invalid except in a recursive relationship)
  - M:M – neither; foreign keys don't matter until the M:M relationship is resolved into multiple 1:M relationships

Keys – summary

The **Primary Key** is shown above the dashed line

Organisation Unit
Org. Unit ID

classifies

Job
Job Code (PK)
Title
Description

An alternate method of showing that the identifier of **Job** is **Job Code**

contains

is contained in

Building
Building ID
Name
Address

is the location of

is located at

Position
Position ID
Building ID (FK)
Job Code (FK)
Org. Unit ID (FK)

is classified by

is filled by

Employee
Employee ID
Name
Address
Birth Date
Gov't ID Number

is assigned to

Alternate Key

Building ID is a **foreign key** that implements the relationship to Building

Employee ID is an inherited key that forms part of the **primary key** of Employee Dependent in combination with the SID (Secondary ID). It also acts as a foreign key.)

Employee Dependent
Employee ID
Emp. Dep. SID
Name
Relationship Code
Birth Date

- A means of specifying a particular instance of an entity

- Typically:

  - Kernel – a system assigned ID

  - Characteristic – the key of the parent plus an SID

  - Associative – the key of all parents, plus an SID if necessary
    (if the same parent instances can be associated multiple times)
    - Important associatives are often given their own ID (e.g., Order ID)

  - Reference or Type – a recognisable Code or a meaningless ID

107

Key propagation rules

Item

Item ID

Supplier

Supplier ID

Kernel entities are assigned a meaningless "ID".

By the way, in this example, individual instances (e.g., serialized inventory) of the Items are not recorded. If they were, we might have to call this entity "Item Type" and the instances "Item," or leave this entity as "Item" but call the instances "Item Unit."

Supplier-Item

Item ID
Supplier ID

Packaging Type

Packaging Type Code
Description

Reference or **Type** entities are assigned a "Code", if they are "codesets" ("reference data") or "classifications"

Associative entities inherit the identifiers of the parents. A "SID" is added, if necessary, to make the inherited key unique.

Supp-Item Packaging

Item ID
Supplier ID
Supplier-Item Pack SID

Pack Price Break

Item ID
Supplier ID
Supplier-Item Pack SID
Price Break SID

Characteristic entities inherit the parent's identifier, and a "SID" is added.

etc.

How far to go?

Item

Item ID

Supplier

Supplier ID

Supplier-Item

Item ID
Supplier ID

Supp-Item Packaging

Item ID
Supplier ID
S-I Pack SID

Pack Price Break

Item ID
Supplier ID
S-I Pack SID
Price Break SID

etc.

Item

Item ID

Supplier

Supplier ID

Supplier-Item

Supplier-Item ID
Item ID (fk)
Supplier ID (fk)

Supp-Item Packaging

Supp-Item Pack ID
Supplier-Item ID (fk)

Pack Price Break

Pack Price Break ID
Supp-Item Pack ID (fk)

etc.

This slide left blank by accident

Advanced Data Modelling – Overview

➡	*Outline*

1. Interesting structures
2. Modelling time & history
3. Rules on relationships and associations
4. Presentation techniques for data modellers
5. Relating Dimensional and Entity-Relationship models

⭐	*Topics*

- Types vs. Instances
- Attribute vectors
- Recursion and generalisation, in general
- Recognising lists, trees, and networks, and modelling them with recursive relationships
- Generalisation (subtyping) - when to use it, and when not to
- Modelling difficult rules

!	*Themes*

- Communication!
- Consistency
- Contextual models
- Complexity

111

Exercise: libraries and bookstores

Your local library and your local bookstore share some obvious similarities:

- Libraries loan books to cardholders (what the library calls a customer) and bookstores sell books to customers. Customers get to keep their purchases, but cardholders have to return whatever was loaned to them within a stated time period.

- Bookstores and libraries both keep track of all transactions ("purchase" or "loan"), but:
  - the library always records the cardholder for the transaction
  - the bookstore only records the customer for the transaction if they belong to their "frequent buyer" program.

Some miscellaneous points:

- Purchases and loans can both cover multiple items.

- Both of them use the term "book" somewhat loosely; they also deal with different Format Types - audiotapes, videotapes, CDs, CD-ROMs, and so on.

- They both call a "book" a "Title," and when a Title is available on a specific Format Type, they call that a "Release."

- Both organisations care about the "Book's" title and author

- Both organisations deal solely with "Books" (whatever you decide to call it) – they do not carry other types of products, or at least Books are all we care about.

What are the most important differences between the two models?

Build a simple data model for the library, and one for the bookstore. Make a guess at a few important attributes for each of the entities in your model.

Exercise work area

Solution: differences and notes

Differences – library vs. bookstore

- loan vs. purchase

- loan – we want it back; purchase – one-time

- one book is loaned many times in a library, but sold once in a bookstore

- library: cardholder (mandatory); bookstore: customer (optional)

- library – loan has a return date; bookstore – we hope there is NO return

- bookstore – has a price (but the library may sell books)

- *types vs. instances – bookstore: type (Title); library: each instance (Copy)*

Diane McKellar, civic government – "Get it wrong and live in pain forever!"

An Inventory Management system was selected that only tracked TYPES of Items. During implementation they discovered they also needed to track INSTANCES of certain Items – Items that could be rebuilt and "rotated" in and out of service.
The solution was a complex and expensive "shadow system" built in Excel.
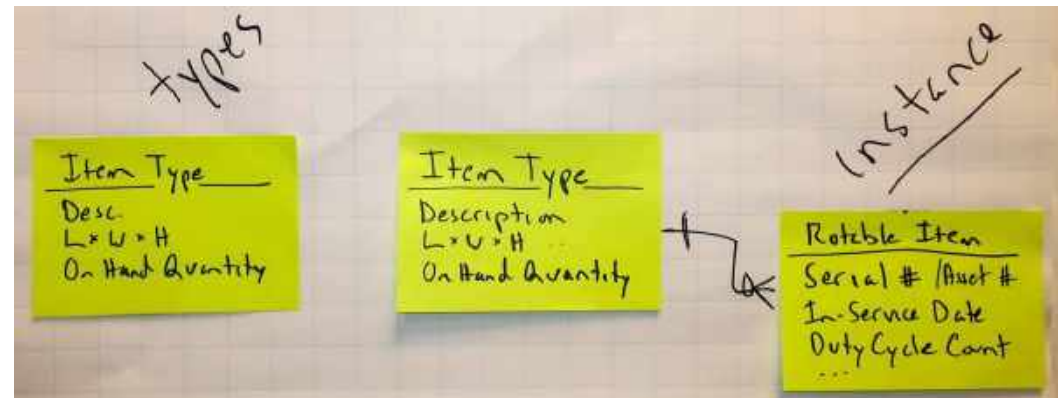
Solution: libraries and bookstores

Library

Format Type
Format Type Code
Description

Title
Title ID
Name
Author

is an instance of

Copy
Title ID
Copy SID
Purchase Price Amt
Acquisition Date
Status Code
Format Type Code (fk)

is classified by

takes

Loan Item
Loan ID
Title ID
Copy SID
Due Date
Return Date
Status Code

Cardholder
Cardholder ID
Name
Number
Member Since Date

is taken by

Loan
Loan ID
Date
Cardholder ID (fk,nn)

is part of

basket
analysis

Book Store

Format Type
Format Type Code
Description

is classified by

Title
Title ID
Name
Author
List Price Amount
Sale Price Amount
On Hand Quantity
Format Type Code (fk)

Customer
Customer ID
Name
Frequent Buyer Number
Enrollment Date

Purchase
Purchase ID
Date
Net Amount (der)
Tax Amount (der)
Total Amount (der)
Customer ID (fk,null)

purchases

is part of

Purchase Item
Purchase ID
Title ID
Quantity
Amount Due (der)

Handling "vectors" of attributes

Vector:
A fixed number of repeating attributes

Divisional Sales (in 1,000,000s)				
Year	Q1	Q2	Q3	Q4
2020	1.45	1.37	1.40	1.67
2021	1.46	1.40	1.63	1.91
2022	2.11	2.32	…	…

← *Each row is a vector*

Examples of vectors:
• 7 days of the week
• 4 quarters in a year
• 12 months in a year
• 52 weeks in a year
• 3 prices for a product (store price, list price, discount price)

116

Modelling vectors

Flight number	Dep from	Time (24 hr)	Arr to	Time (24 hr)	Frequency
SQ017	YVR	1225	SIN	2335+1	1 - - 4 - 6 -
SQ500	SIN	0715	BLR	0900	1 - - - 5 6 -
SQ502	SIN	2000	BLR	2155	1 2 3 4 5 6 7
SQ501	BLR	1015	SIN	1720	1 - - - 5 6 -
SQ503	BLR	2310	SIN	0605+1	1 2 3 4 5 6 7
SQ018	SIN	0950	YVR	1105	1 - - 4 - 6 -

Above are some of the flights you need to know about in order to travel from Vancouver (YVR) to Bangalore (BLR) via Singapore (SIN) on Singapore Airlines (SQ.)

"Frequency" indicates which days of the week the flight operates by using a string of 7 characters, with position "1" representing Monday, "2" Tuesday, through to position "7" indicating Sunday.
If the number is present, the flight operates on that day.
If a dash ("-") is present, the flight does not operate on that day.

SQ017 operates Monday, Thursday, and Saturday, and *not* on Tuesday, Wednesday, Friday or Sunday.

SQ502 operates every day of the week.

Build some alternative data models to record this subset of the flight schedule.

Make note of some of the decisions you have had to make in preparation for class discussion.

117

Modelling vectors – solution

	Flight
SQ	Airline Code
017	Flight Number
YVR	Departure Airport
1225	Departure Time
SIN	Arrival Airport
2335+1	Arrival Time
	Operates Monday Flag
	Operates Tuesday Flag
	Operates Wednesday Flag
	....

Row-wise solution
+ easy to display data
- inflexible
- queries more difficult

	Flight
SQ	Airline Code
017	Flight Number
YVR	Departure Airport
1225	Departure Time
SIN	Arrival Airport
2335+1	Arrival Time

Day
Day Code

Operational Day
Airline Code
Flight Number
Day Code
Operates Flag
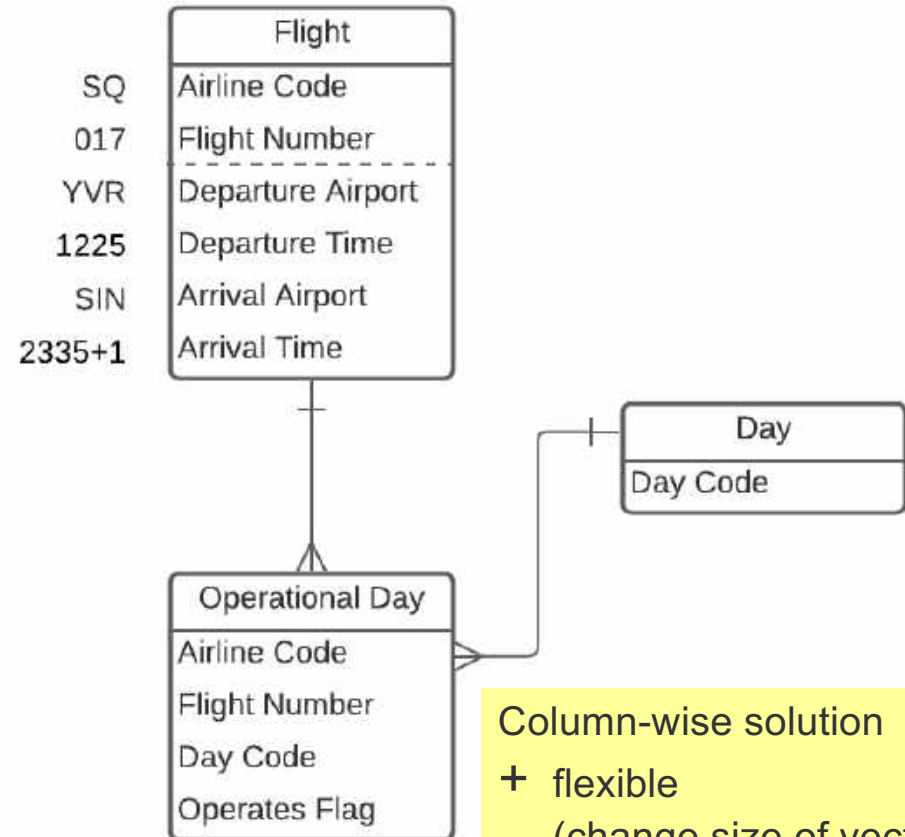
Column-wise solution
+ flexible
  (change size of vector)
+ Many queries are
  much easier

Many "interesting" physical
implementations are possible.

118

Alternatives for modelling vectors

"Row-wise" table	**"Column-wise" table** ✔

- one row per vector; attributes go in separate columns

Division
Division ID _ _ _ _ _ _ _
Name
Budget Center Code

is calculated for

Div. Sales Summary
Division ID
Div. Sales Summ. Year _ _ _ _
Q1 Total Amount
Q2 Total Amount
Q3 Total Amount
Q4 Total Amount
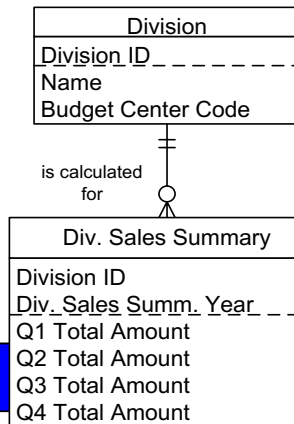
Advantages

- familiar layout
- from "row to screen" is easier
- fewer tables and joins

- multiple rows per vector; attributes go in a single column

Division
Division ID _ _ _ _ _ _
Name
Budget Center Code

is calculated for

Div. Sales Summary
Division ID
Div. Sales Summ. Year
Quarter Number _ _ _ _ _ _ _
Total Sales Amount
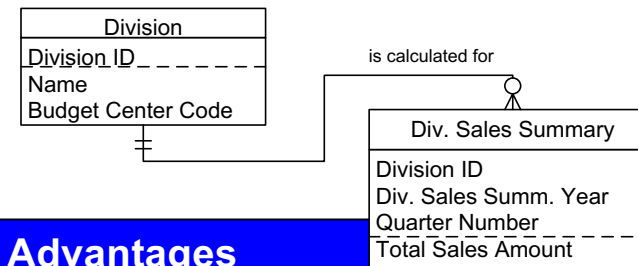
Advantages

- same handling as for other multi-valued attributes
- easier SQL queries (e.g., average sales)
- More efficient for sparse data
- flexible:
  – change vector length
  – add additional attributes (like Top Sales Rep for each Quarter)

119

Recursion

✓ When one entity occurrence can be related to another occurrence of the same entity type

✓ Also know as a "self-referencing" relationship

✓ Three variations:

- 1:1

- 1:M

- M:M

✓ Often involves "generalising"
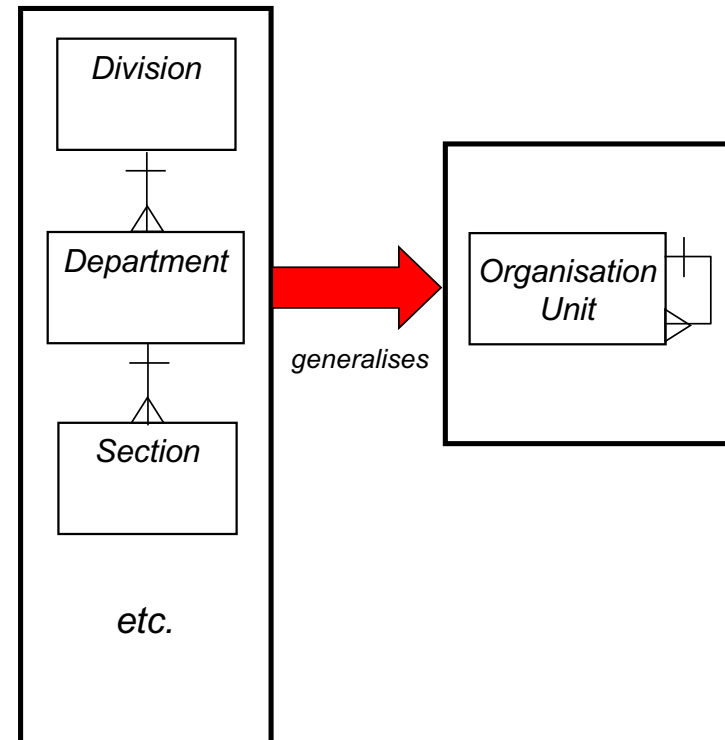
Recursion - recognising the data structure

Items arranged in a _linked list:_

Item	precedes
Item ID _ _ _ _ _ _ _ _ _ _	
Follows Item ID (fk, null)	1:1
other attributes...	
	follows
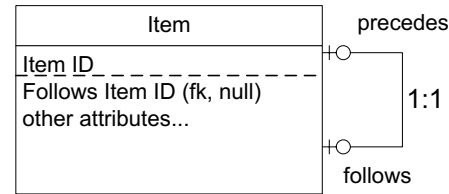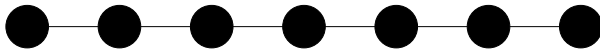
Items arranged in a tree:

Item	parent of
Item ID _ _ _ _ _ _ _ _ _ _	
Child Of Item ID (fk, null)	1:M
other attributes...	
	child of
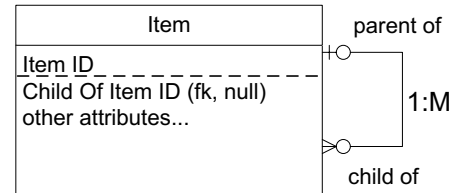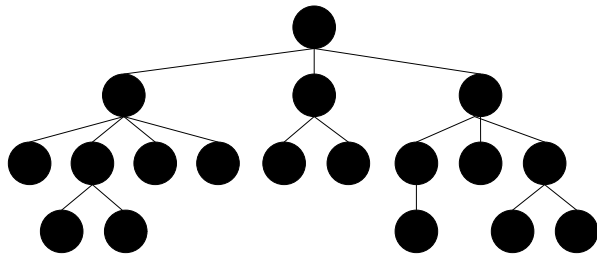
Items arranged in a network:

Item	linked to
Item ID _ _ _ _ _ _ _ _ _	
other attributes...	M:M
	linked to

Item
Item ID _ _ _ _ _ _ _ _
other attributes...

from   to

Item Link
Item ID
Linked Item ID _ _ _ _ _ _
other attributes...
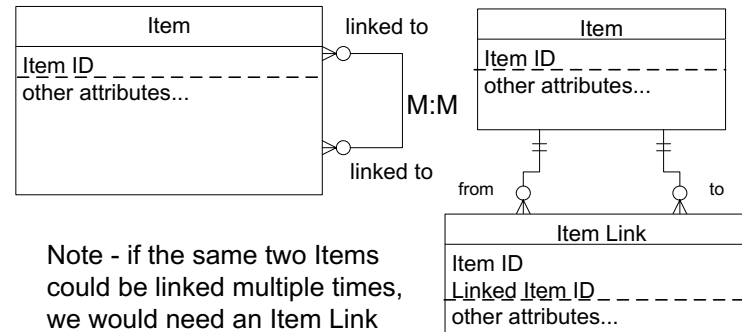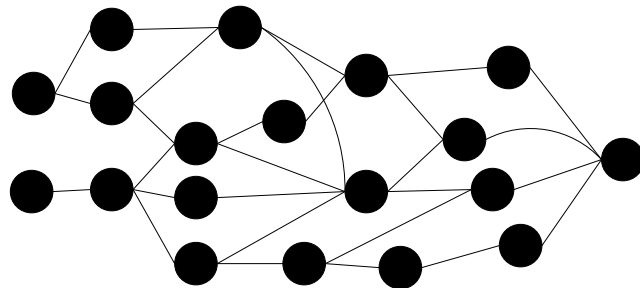
Note - if the same two Items
could be linked multiple times,
we would need an Item Link
SID in the primary key

121

Recursive relationships – 1:1 example

"back"　　　　　　　　　　　　　　　　　　　　　　　　　　**"front"**

| 12345 | 54321 | 24680 | 97531 | 01030 |
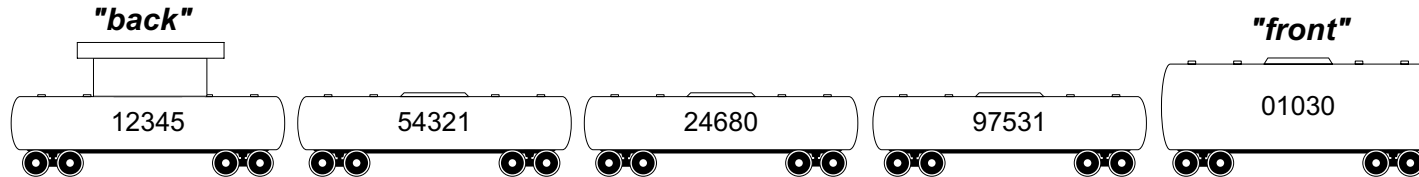
- ✓ The train above is an example of a "linked list"
- ✓ A linked list can be handled with a recursive 1:1 relationship
- ✓ All the items in the list must be generalised into the same type of entity(or a supertype with multiple subtypes - "Rail Car" would subtype into "Freight", "Locomotive", "Passenger", etc.)
- ✓ The foreign key can either "point ahead" or "point back" – depends which end you add new instances from
- ✓ As always, the recursive relationship is "fully optional"
  - • the first car doesn't have a car in front of it
  - • the last car doesn't have a car behind it.

Rail Car

ID

Type Code (fk)
Acquisition Date
Manufacture Date
Next Forward Rail Car ID (fk, null)

connects to
the front of
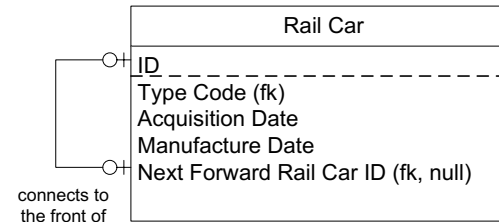
"Rail Car" Sample Instance Table	
Rail Car ID	Next Forward Rail Car ID
12345	54321
54321	24680
24680	97531
97531	01030
01030	null

Can you think of an example where you would use this?

Recursive relationships – 1:M example

Partial Organization Chart - 1999/07/12
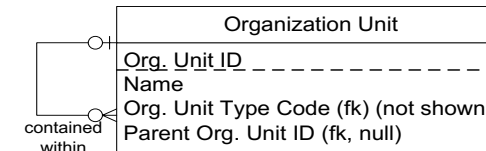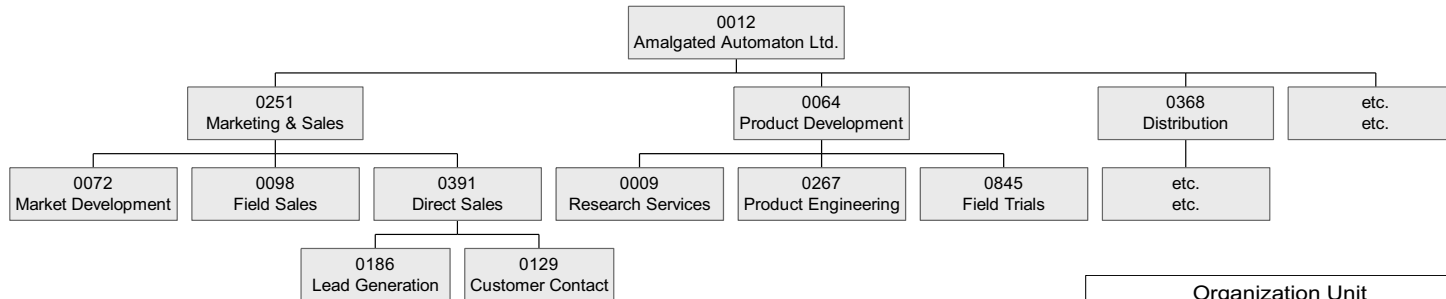
- ✓ The organisational structure is a "hierarchy"
- ✓ A hierarchy can usually be handled with a recursive 1:M relationship
- ✓ Again, this requires all the items to be generalised into the same type of entity
- ✓ The foreign key must be at the "Many" end, so the child "points to" the parent
- ✓ As with all recursive relationships, this is "fully optional"

Organization Unit

Org. Unit ID _ _ _ _ _ _ _ _ _ _ _
Name
Org. Unit Type Code (fk) (not shown)
Parent Org. Unit ID (fk, null)

contained within

"Org Unit" Sample Instance Table	
Org. Unit ID	Parent Org. Unit ID
0012	null
0251	0012
0064	0012
0368	0012
0072	0251
0098	0251
0391	0251
0186	0391
0129	0391
0009	0064
0267	0064
0845	0064
etc.	etc.

Can you think of an example where you would use this?

123

Recursive relationships – M:M example

- ✓ This project plan is a "network"
- ✓ A network is handled with a recursive M:M relationship
- ✓ This resolves to an associative entity linking two different instances of the same entity

follows

precedes

Task

Task ID
Name
Description

resolves to...

Task

Task ID
Name
Description

preceding

following

Task Precedence

Preceding Task ID
Following Task ID

0005 Task D	0012 Task AC
0001 Task A	0002 Task B	0004 Task X	0022 Task K	0007 Task F
0003 Task C			0018 Task N
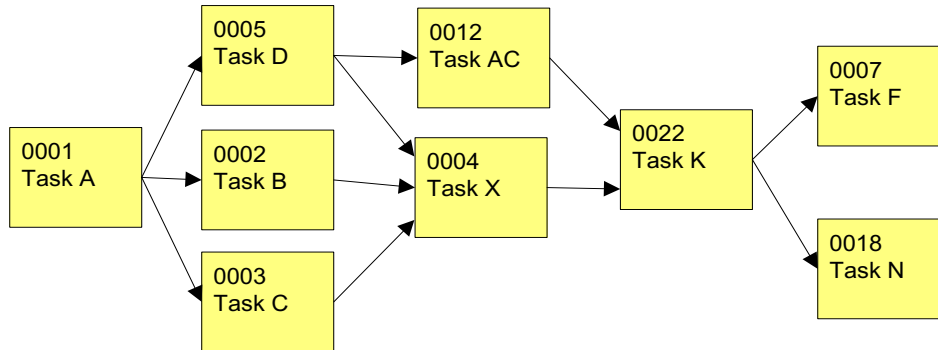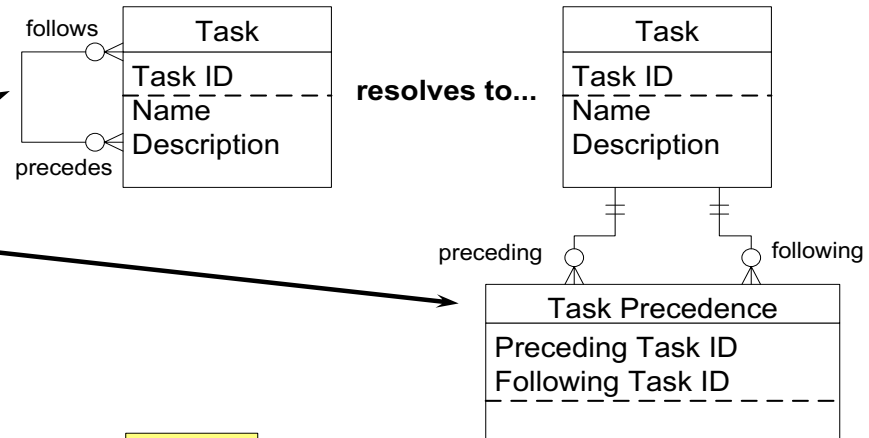
"Task Precedence" Sample Instance Table	
Preceding Task ID	Following Task ID
0001	0005
0001	0002
0001	0003
0005	0012
0005	0004
0002	0004
0003	0004
0012	0022
0004	0022
0022	0007
0022	0018

Can you think of an example where you would use this?

Supertypes and subtypes

Supertype

Subtypes

all jobs

Job

Job Title
Creation Date
Job Type Code

describes
duties of

Employee

performs

**Management
Job**

Salary Amount

**Bargaining
Unit Job**

requires

Certification

required
for

Hourly Wage Amt
Confidential Flag

only B.U. jobs
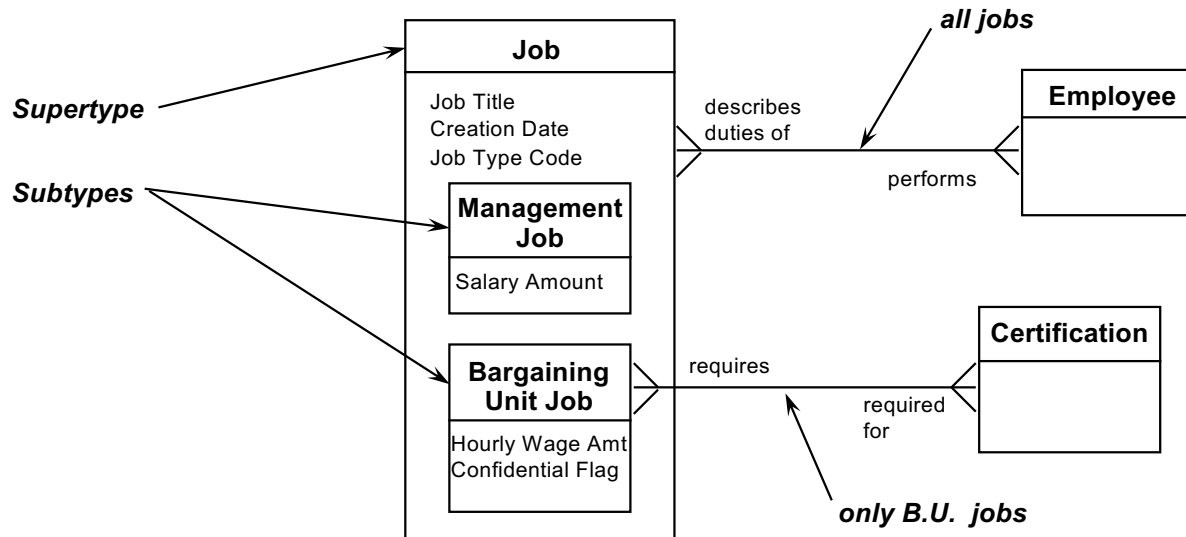
- Breaks an entity down into two *or more* 'subtypes', or generalises two or more into a single 'supertype'
  - common relationships and attributes go into *supertype*
  - unique relationships and attributes go into *subtype*
- Subtypes are mutually exclusive and mandatory – there is exactly one subtype instance for each supertype
- a.k.a., generalisation-specification, or gen-spec

Generalisation vs. subtyping

✓ "*Generalisation - Specialisation*" is typical O-O terminology; "*Supertype - Subtype*" is typical E-R terminology. Gen-spec.

✓ Generalise whenever two or more entities, each with their own *distinct* attributes and relationships, also *share* other attributes and relationships

✓ Automobile, Aircraft, and Vessel have common attributes that could be generalised into Vehicle…

✓ …or, Vehicle could be sub-typed into Automobile, Aircraft, and Vessel, with the same outcome

✓ Note that it's common for a subtyped entity to also be classified by a *type* or *reference* entity. In this example, Vehicle Type Code is the "subtype discriminator."
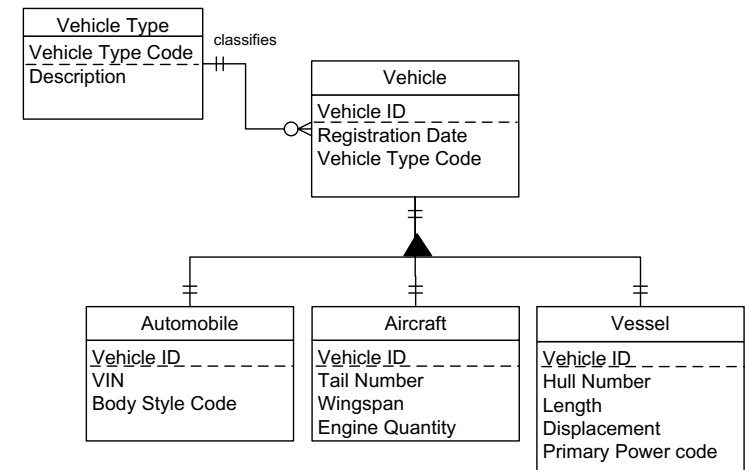
Subtyping - alternative formats

Two different diagramming approaches are widely used -

- "Box in box"
  (e.g., Oracle modelling tools)

- Generalisation hierarchy
  (e.g., most ER- or UML-based modelling tools)

Rail Car

Rail Car ID _
Manufacture Date
Acquisition Date
Rail Car Type Code (fk, nn)
Manufacturer ID (fk,nn)
etc.

Passenger	Locomotive	Freight
Seats Count	Horsepower	Capacity
Has Galley Flag	Fuel Type Code	Tare Weight
etc.	etc.	etc.

Rail Car

Rail Car ID _ _ _ _ _ _ _ _ _
Manufacture Date
Acquisition Date
Rail Car Type Code (fk, nn)
Manufacturer ID (fk,nn)
etc.

is either

Passenger	Locomotive	Freight
Seats Count	Horsepower	Capacity
Has Galley Flag	Fuel Type Code	Tare Weight
etc.	etc.	etc.

127

Don't confuse "roles" and "subtypes"
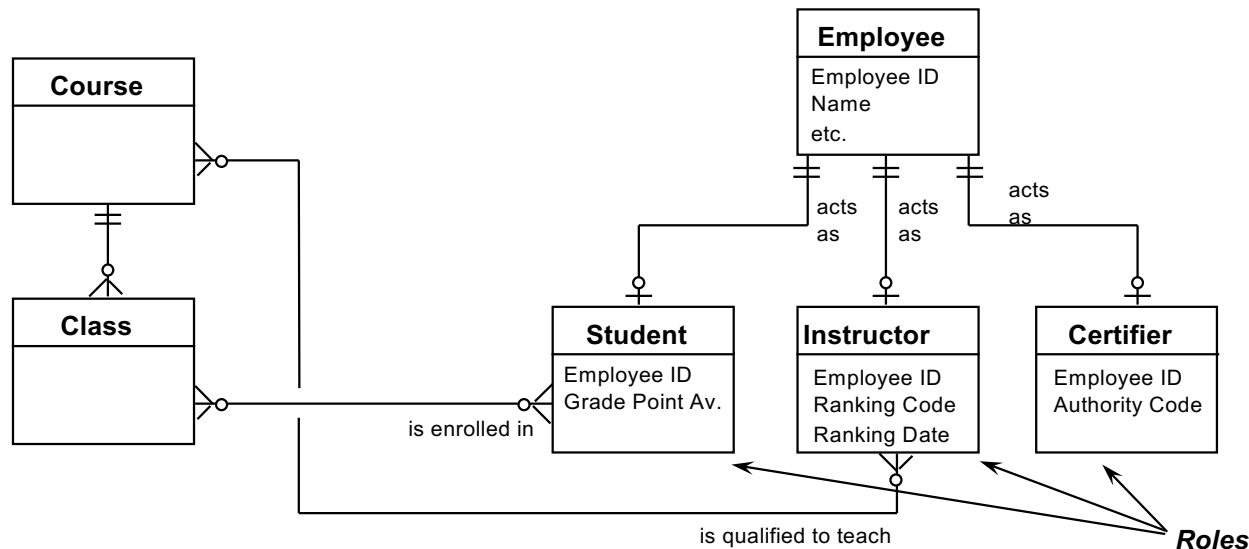
✓ A "role" structure is used when there are two or more different "roles" that an entity type can take on.

✓ *Similar* to subtyping
  • unique attributes and relationships go in the role entity

✓ *Different* from subtyping
  • the roles are *not* mutually exclusive
  • the parent does not necessarily have to take on any role

128

An example with supertype, subtypes, and roles

Note that some roles are valid for one *subtype* (Person or Organisation.)
If a role is valid for both, we connect it to the *supertype* (Party.)

Supertype

Subtype Subtype

What I *AM*

Roles

What I *DO*
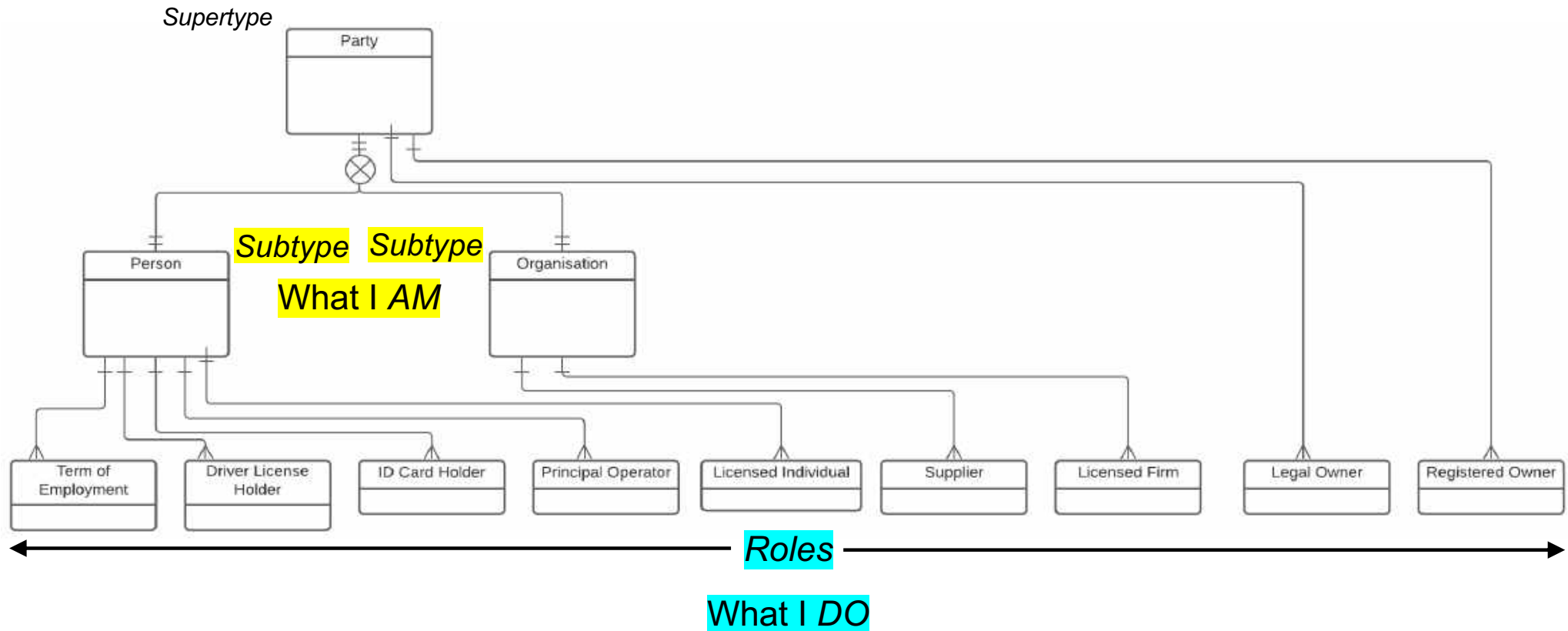
When to generalise

Generalisation is <u>good</u> if generalised items -
- ✓ *are fundamentally the same*
- ✓ *share common facts and behaviours*

Without specification | **With specification**

130
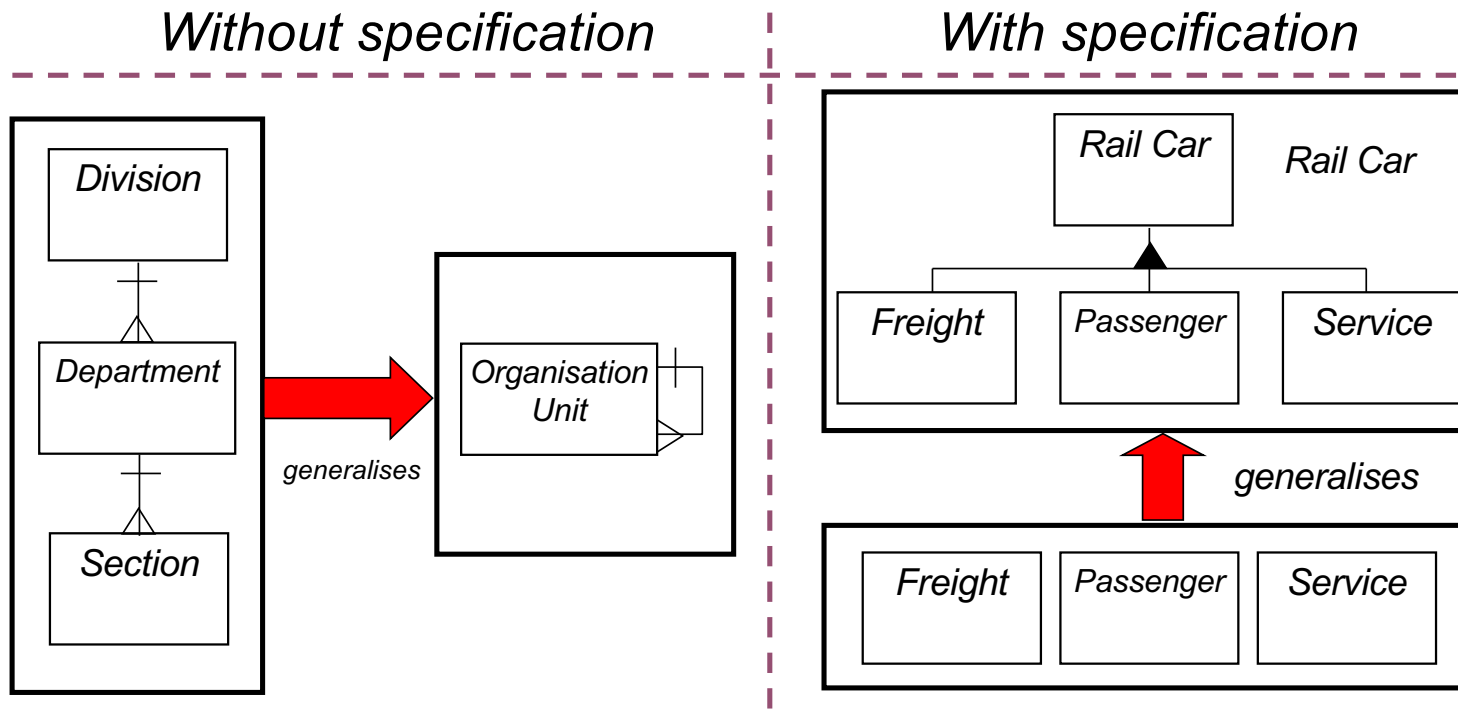
Generalising "too soon"

"Too soon" | *"Just right"*
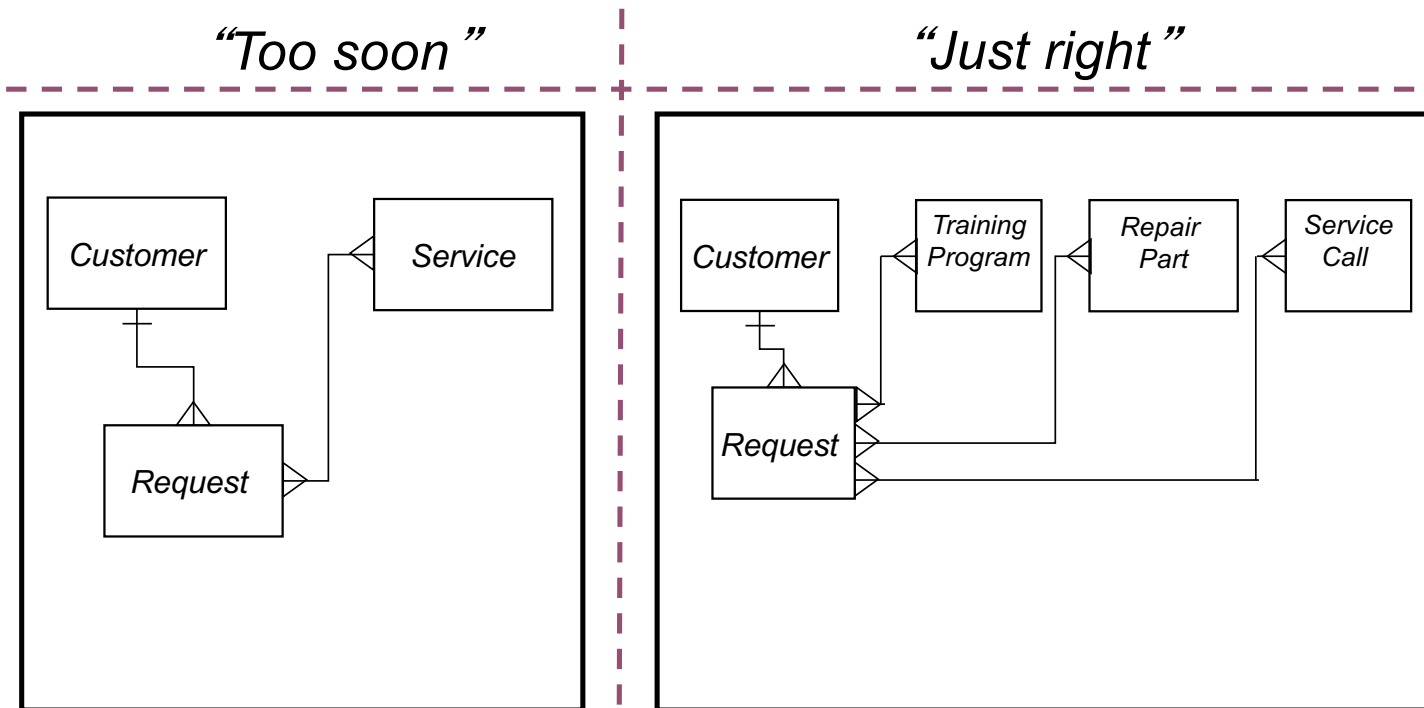
- ✓ Confuses the client
- ✓ Reduces the chance of discovering *"specifics"*
- ✓ Specifics first, generalisation later

Generalising "too much"

"Good"

Person

Person ID
Name
Eye Colour
Height
Etc.

Good!

Address
Type

Good!

Address

"Way over the top"

Person

Person ID

A truly esoteric data structure indecipherable to mere mortals

Characteristic

?
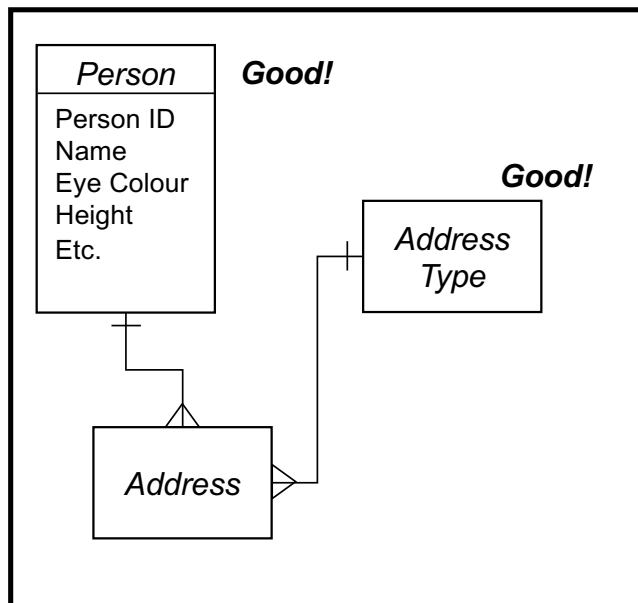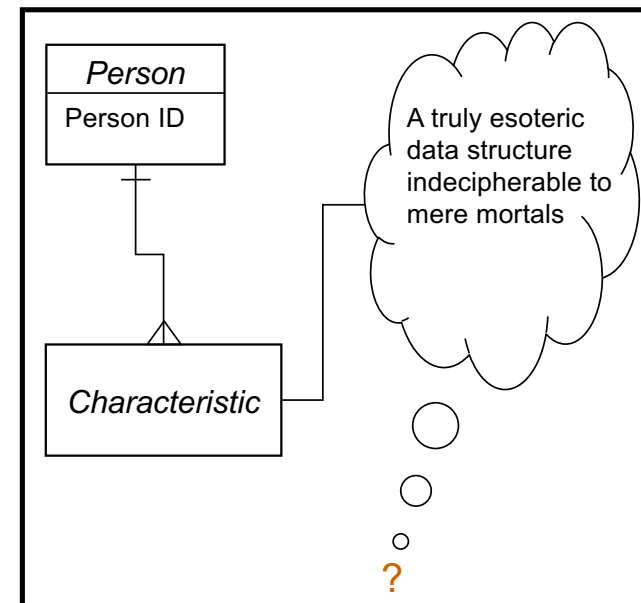
✓ **Really** confuses the client

✓ Flexibility is an **illusion** because programming and reporting are so difficult

Combining recursion and generalisation

✓ *Business rules can often be handled by a recursive relationship involving supertypes and subtypes:*

Between subtypes

Between super and subtype
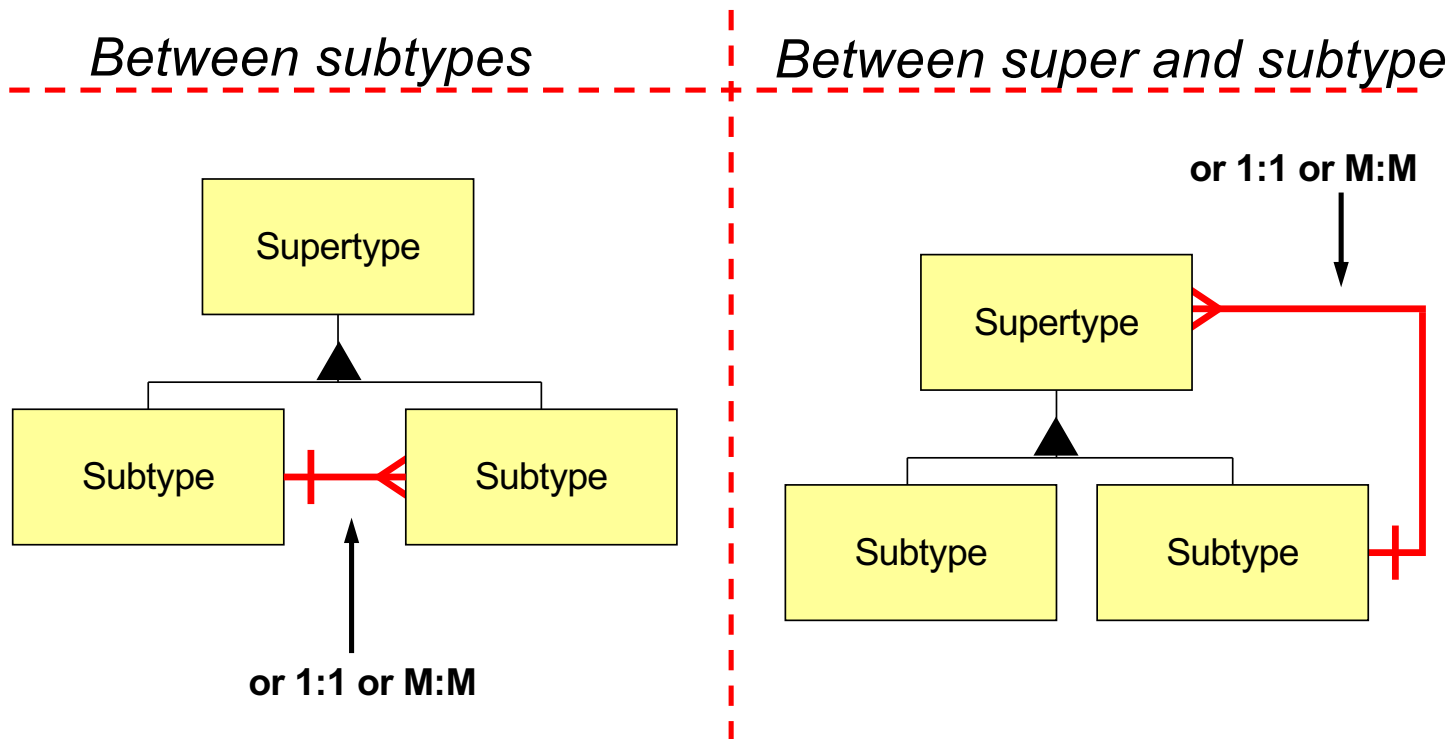
Meeting a new requirement…

Confirm and extend the model:

✓ *discover new requirements, using a variety of techniques*
   e.g., look for multi-valued attributes

Philosophy

✓ *don't dive in – start simple, add detail in layers*

✓ *start out in "natural language"*

See example on the next page from our "Good entity?" exercise

1) State the new requirement as an *assertion*	✓ Start out using the *client's* language ✓ *Then*, ensure that the assertion uses terms from the data model (entity names, relationship names, etc.) This "leads" you to the solution. ✓ ***Confirm it!***
2) Develop a conceptual solution	✓ Look for the simplest option first: no change needed, a new reference attribute, a multi-valued attribute(s), M:M relationship, new entity ✓ Explore rules, like "what is the basis for multi-valued?" ✓ ***Confirm it!***
3) Develop a logical solution	✓ Fully normalised, fully attributed ✓ Follow an "orderly script" – *don't get ahead of yourself or the client* ✓ ***Confirm it!,*** possibly using other easy-to-follow formats such as screen or report mock-ups.

Another level-setting exercise
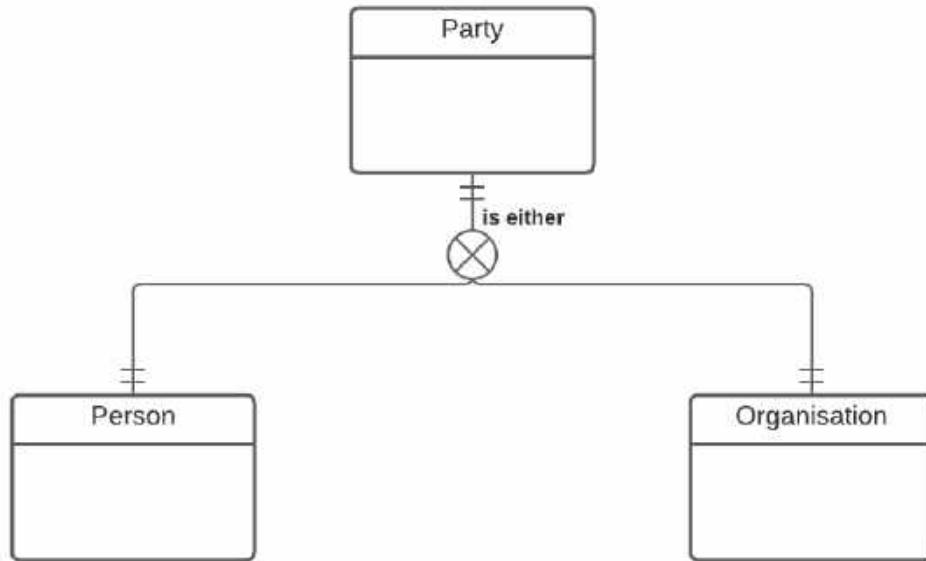
Key point:

State the constraint or fact you are trying to model in plain language before drawing the model.

Extend the model so that it can record these additional facts:

1 – Organisational structure
An Organisation must be one of the recognised types, such as Corporation, Partnership, or Society.

An Organisation may be made up of multiple Organisation Units (an internal subdivision,) each of which might break down further into lower-level Organisation Units, and so on.

Each Organisation Unit has only one parent Organisation Unit. Some Organisation Units have no parent Organisation Unit, because they depend directly on an Organisation. That is, they are the highest level of Organisation Unit.

2 – Rules on Organisational structure
Each Organisation Unit is of a specific type, such as division, department, area, team, section, etc. Only certain relationships between types are valid. E.g.,
- a division can contain a department, but a department cannot contain a division.
- a team can be contained within an area or a division
Note – only certain types of Organisation Units can be immediately subsidiary to an Organisation, but we won't model that constraint at this time.

3 – Organisation ownership
An Organisation may have multiple owners, each of which could be another Organisation or a Person.

136

Exercise work area

Solution
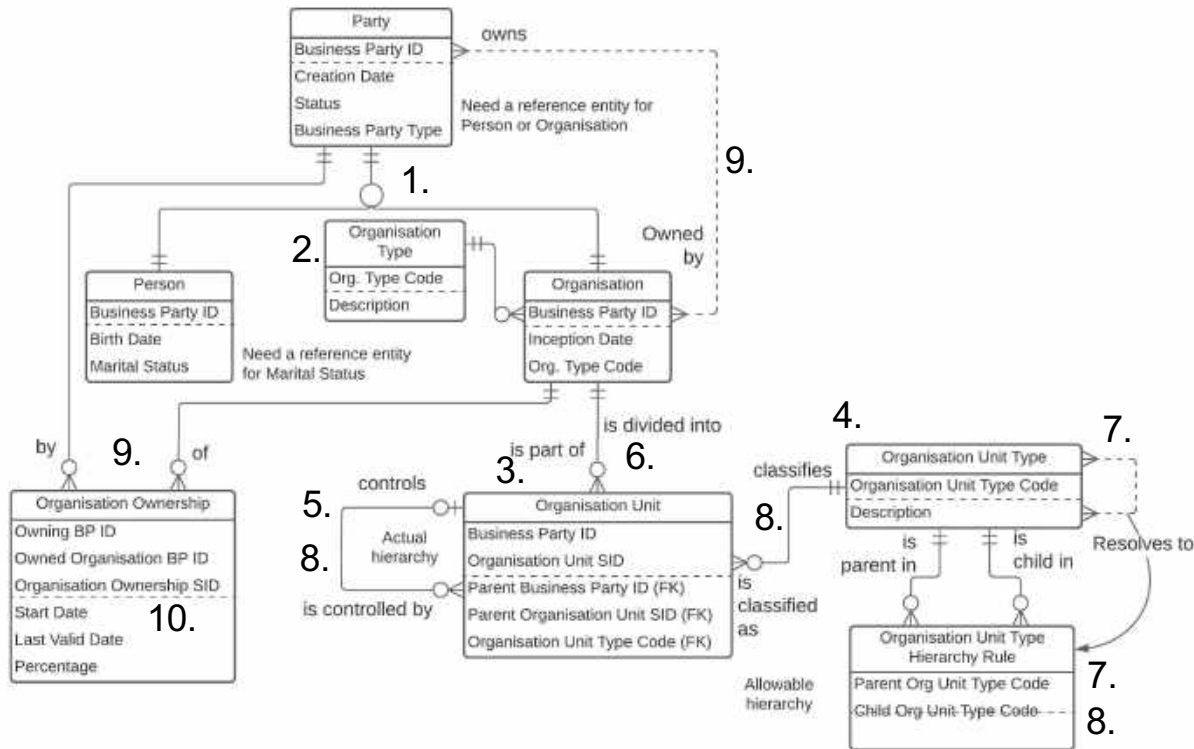
Assertions:
1. Each Party is either a Person or an Organisation
2. Each Organisation must be classified as one Organisation Type and Each Organisation Type may classify one or more Organisations
3. Each Organisation may divide into one or more internal Organisation Units and
   Each Organisation Unit must be part of exactly one Organisation
4. Each Organisation Unit must be classified as one Organisation Unit Type and
   Each Organisation Unit Type may classify one or more Organisation Units
5. Each Organisation Unit may control one or more other Organisation Units and
   Each Organisation Unit may be controlled by one other Organisation Unit
6. The controlled and the controlling Organisation Units must be part of the same Organisation
7. Each Organisation Unit Type may control one or more other OU Types and Each Organisation Unit Type may be controlled by one or more other OU Types
8. The Organisation Unit Type of the controlled Organisation Unit and the Organisation Unit Type of the controlling Organisation Unit must be a pair found in Organisation Unit Hierarchy Rule
9. Each Organisation may be owned by one or more Persons or Organisations (which is to say one or more Parties) and Each Party may own one or more Organisations
10. A Party may own an Organisation multiple times over a period of time

138

Exercise: stock exchange trading

Please build a data model from the following facts:

Companies issue shares in various stocks. For instance, Algonquin Industries has issued common stock, preferred A stock, and preferred H stock.

Each stock may be listed on multiple stock exchanges. For instance, Algonquin's common stock is listed on the Vancouver and Toronto exchanges, but its preferred stocks are only listed in Vancouver.

When a customer wishes to buy or sell shares of a particular stock, they place an order on one of the exchanges. The order says, in effect, that "I am offering to buy (or sell) X quantity of stock Y for price Z for the next W days" If it is a sell order, the customer must also (by law) indicate if they are short selling (Short Sale Flag is set)

The Automated Trading System matches up buy and sell orders if the prices are within certain parameters. A complex algorithm determines the actual price of the sale. Note that an order may not be satisfied all at once (i.e., with one sale). For instance, an order to sell 10000 shares may be matched with a buy order for 5000 shares, another for 3000 at a later time, and it may expire before the remaining 2000 shares sell.

Build an initial E-R data model illustrating all the relevant entities, and their relationships.

1. Identify the main entities

2. Agree simple definitions

3. State assertions that describe the scenario

4. Arrange entities by dependency

5. Add and name relationships

   • name the relationship first

   • only then add cardinality (ones and manys)

Assertions and clarifications

- Each Company may issue one or more Stocks and each Stock must be issued by one Company

- Each Stock must be classified as one Stock Type and each Stock Type may classify one or more Stocks

- Each Stock may be listed on one or more Exchanges and each Exchange may list one or more Stocks
  (The Company chooses which Exchanges to list on)

- Each Customer may place
  one or more Buy or Sell Trade Orders &
  Each Buy or Sell Trade Order must be placed by one Customer

- Each Trade Order is either a
  Buy Trade Order or a Sell Trade Order

- Each Buy Trade Order may be filled by one or more Sell Trade Orders and each Sell Trade Order may be filled by one or more Buy Trade Orders

- The Buy and Sell Trade Orders for a Trade must be placed by different Customers

- Each Trade must be a match of one Buy Trade Order and one Sell Trade Order

- A Listing is an authorisation to buy or sell a specific Stock on a specific Exchange

- A Buy Trade Order is an offer to buy …
  A Sell Trade Order is an offer to sell…
  - a stated quantity
  - of a specific Stock
  - on a specific Exchange
  - at a specified price
  - during a specified time period

- The matching of a Buy Trade Order and a Sell Trade Order is referred to as:
  - a Sale
  - a Match
  - a Fill
  - ***a Trade***
  - a Buy/Sell Transaction…

Sample Instance Model
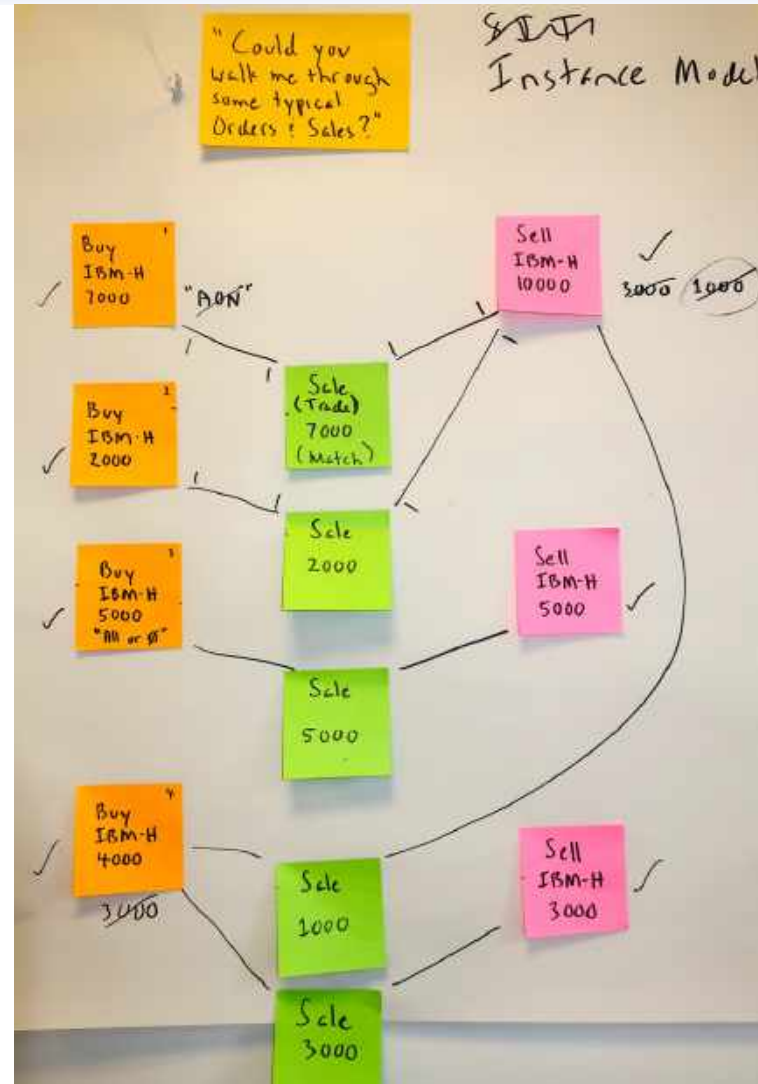
Possible entities for the Stock Exchange model

Initial Concept Model for Stock Exchange Trading

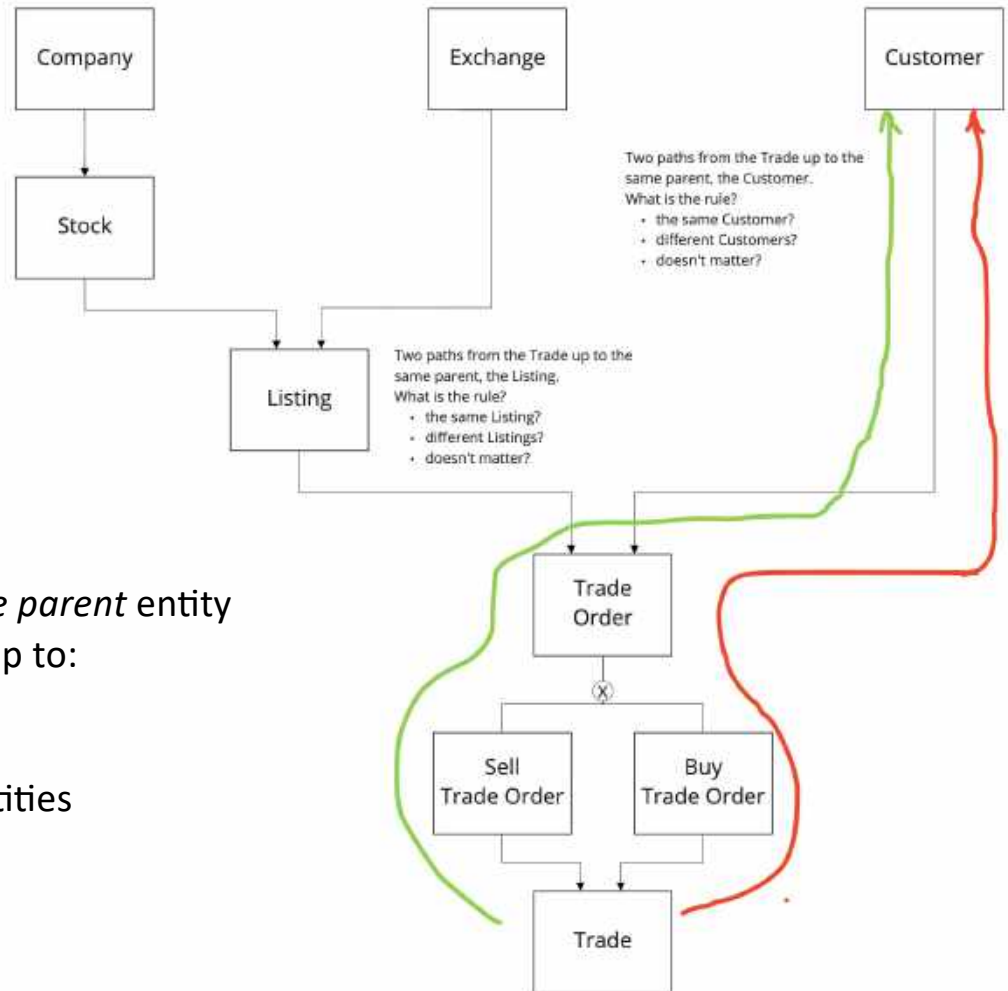Instead of the "hash mark" (One) and "crowsfoot" (Many) this uses the symbol Miro offers – an arrowhead
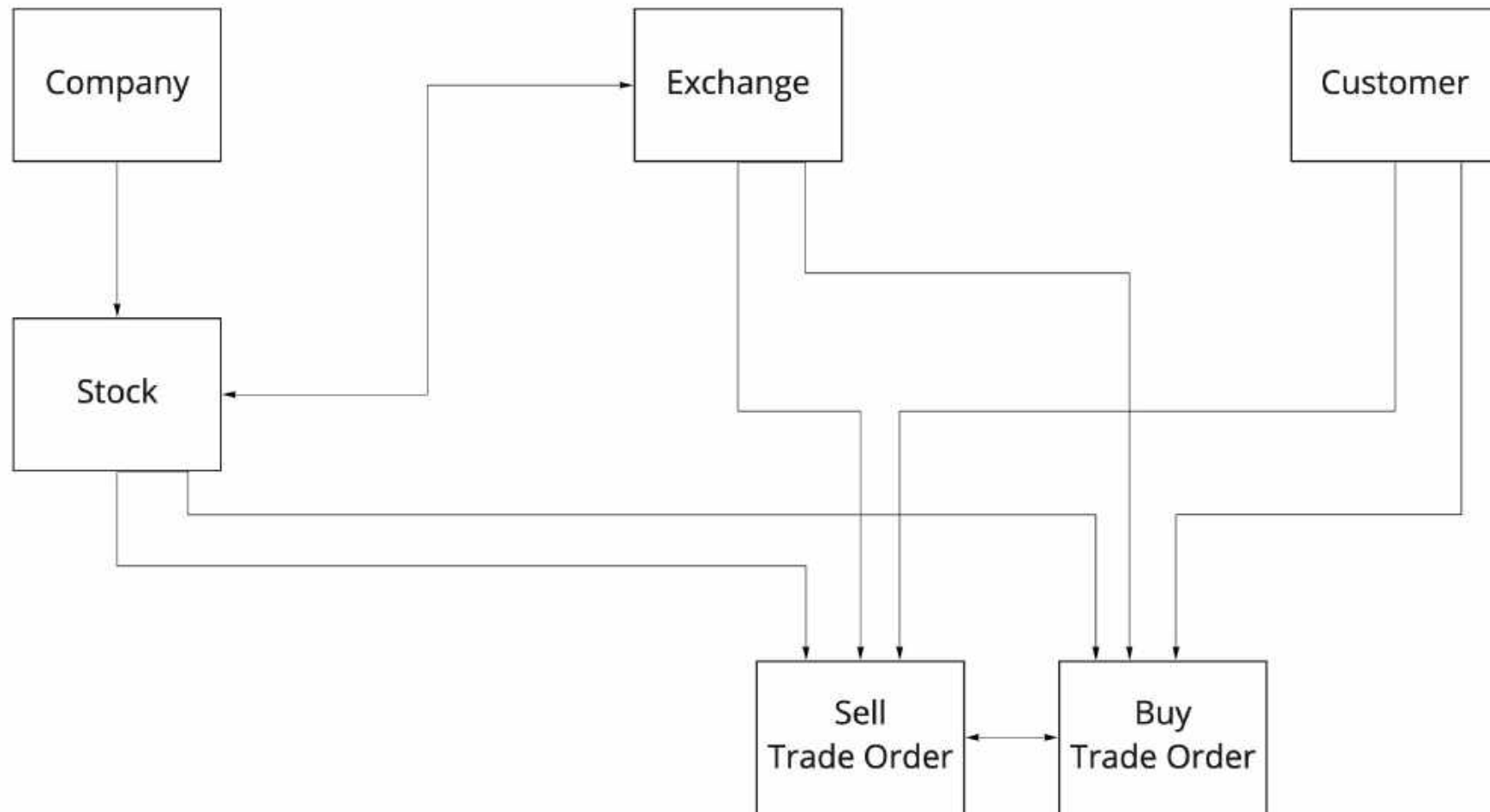
Second iteration Concept Model for Stock Exchange Trading

Key point!
Resolving the M:M between *Stock* and *Exchange*
(creating *Listing*) and
generalising *Sell Order* and *Buy Order* into
Trade Order makes the model *easier* to understand
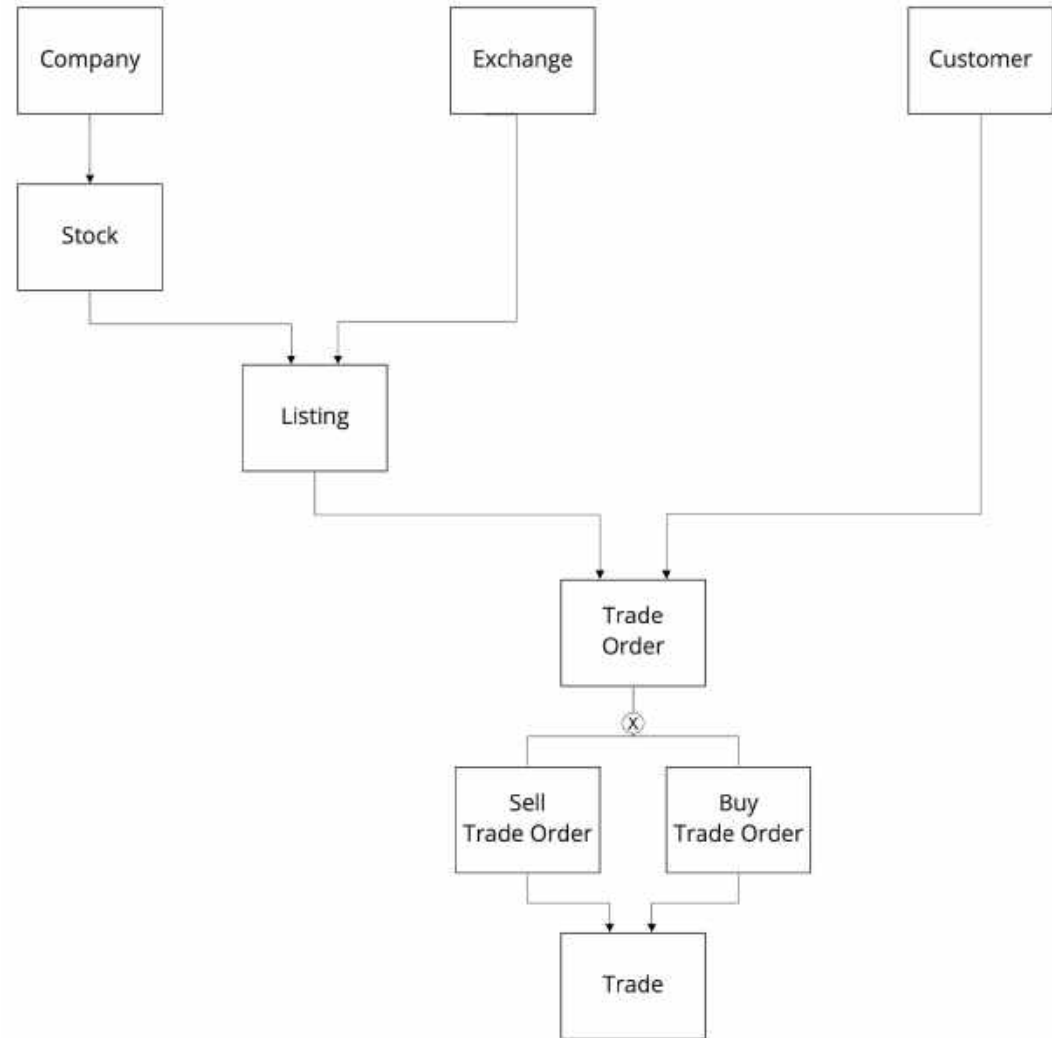
An important constraint to check for

Company

Exchange

Customer

Stock

Two paths from the Trade up to the
same parent, the Customer.
What is the rule?
- the same Customer?
- different Customers?
- doesn't matter?

Listing

Two paths from the Trade up to the
same parent, the Listing.
What is the rule?
- the same Listing?
- different Listings?
- doesn't matter?

Trade
Order

Sell
Trade Order

Buy
Trade Order

Trade

When there are *two* paths up to the *same parent* entity
always check if the two paths must lead up to:
- the *same* parent entity
- *different* parent entities
- either the *same* or *different* parent entities
  (*it doesn't matter*)

We must check the paths from
- Trade to Customer and
- Trade to Listing

Don't generalise too soon! Specifics first, generalisation later

Client:
"Wow! Buy and Sell
Trade Orders
look very similar!"

Me:
"Omigosh!
That's amazing!"
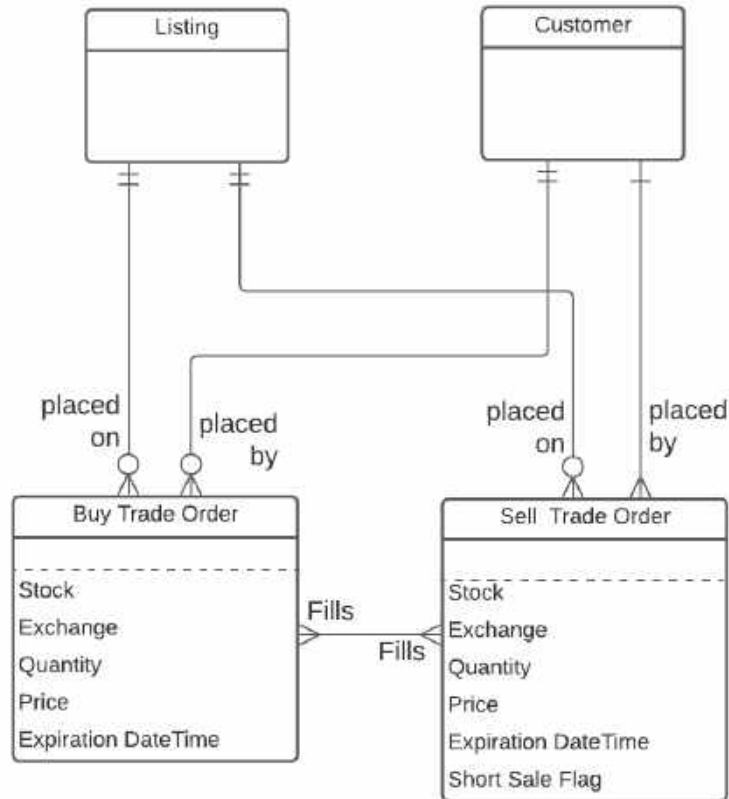
Specifics

Generalisation
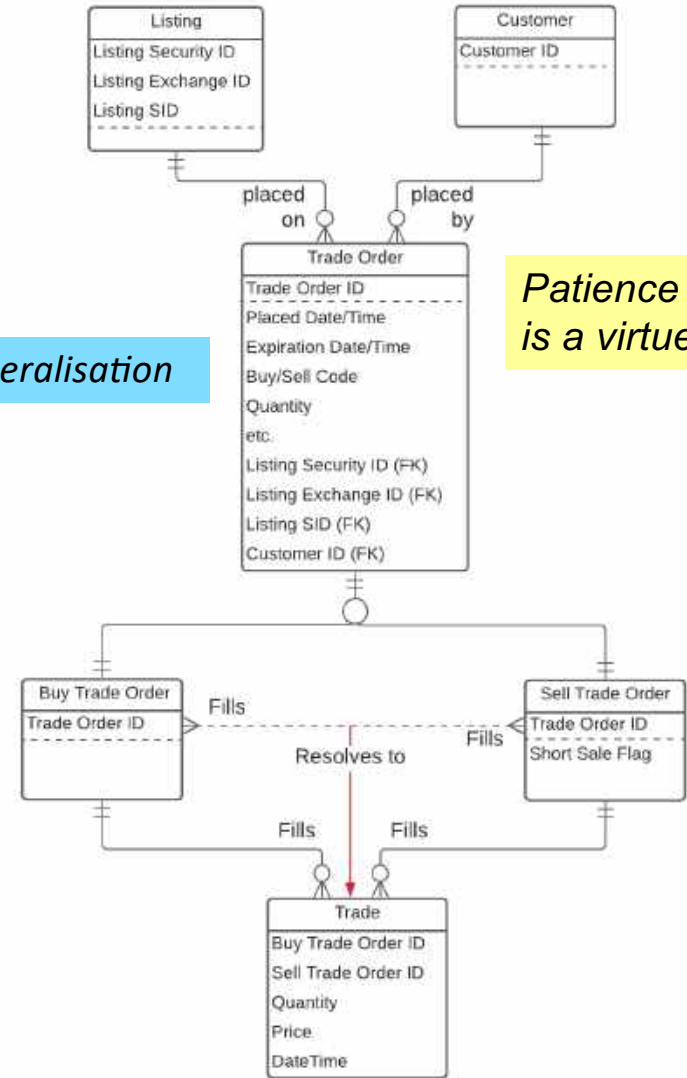
Patience is a virtue!

146

Complete solution: stock exchange trading

Company

Company ID
Name

Exchange

Exchange ID
Name
Abbreviated Name

Customer

Customer ID
Name

trades

placed by

issued by

Security

Security ID
Name
Issued By Company ID (fk, nn)

Trade Order

Trade Order ID
Buy/Sell Code
Original Quantity
Remaining Quantity (der)
Price
Expiration Date/Time
Listing Security ID (fk, nn)
Listing Exchange ID (fk, nn)
Listing SID (fk, nn)
Customer ID (fk, nn)

is either

of   on

Listing

Security ID
Exchange ID
Listing SID
Status Code
Effective Date
Expiration Date
Symbol

Buy

Sell

Short Sell Flag

fills   fills

Trade

Buy Trade Order ID
Sell Trade Order ID
Quantity
Price
Timestamp

Constraints

- Buy Trade Order and Sell Trade Order involved in a Trade must reference the same Listing. Using fully inherited keys for Trade Order and Trade would make this easier to enforce.

147

This slide left blank to maintain balance in the universe

Modelling time & history

➡	Outline
1.	Interesting structures
2.	**Modelling time & history**
3.	Rules on relationships and associations
4.	Presentation techniques for data modellers
5.	Relating Dimensional and Entity-Relationship models

★	Topics
•	Issues in modelling history
•	History vs. audit trail
•	"What's wrong with this model?"
•	Modelling for "as of" reporting
•	Physical time and Business time
•	Time-specific considerations

History in data models
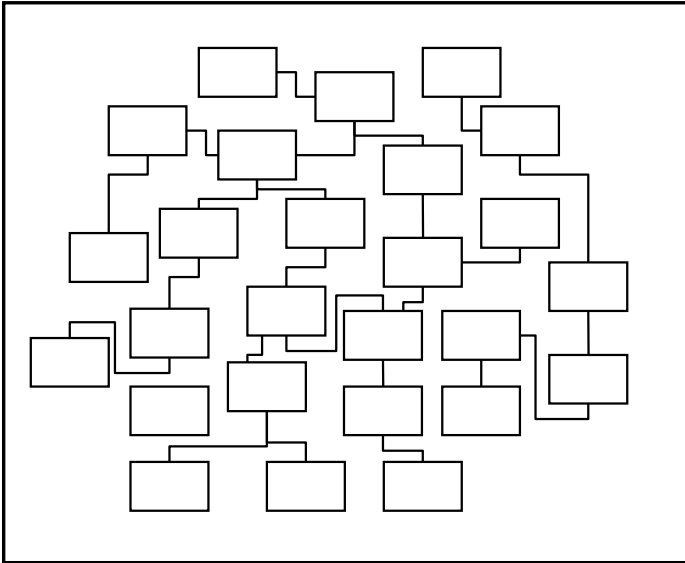
✓ Ability to record previous values for facts

✓ Facts: attributes or relationships

✓ Ability to record history, with care, gives ability to record future cases

Issues

- E-R diagramming techniques don't support time-based rules, e.g., "1:1 at a point in time, 1:M over time"

- Two types of change – business data change, and correction

- The client might *initially* say they don't need history, but…

History vs. audit trail

History	Audit Trail
• Previous (or future) values for attributes or relationships	• Information about the transaction (event) that caused the change, together with before/after image
• "Employee 51943 was named John Bartlett until 2020/05/12; from 2020/05/13 onward his name is Bart Johnson"	• "On 2020/05/20 a Change Name transaction was processed for Employee 51943 from Node SX1749 by Employee 42778"
• Must decide whether model should reflect time dependency – the norm is to include it	• Must decide whether model should depict audit trail – the norm is to exclude it unless it is needed for operational processes

Exercise: time dependent data

Jim needs to track List Price and Store Price over time for each Title the store carries. It is very important for him to be able to list the history of changes to either price, and the date, so that pricing information can be compared to sales figures.

Either or both prices may change at any time. Often, a price change is known before it is due to take effect.

The model to the right has several flaws – try to list at least five.

Then, construct a model (or a few alternatives) that will support Jim's needs.

Title
Title ID
Name Artist Name Publisher Name etc. etc. etc. List Price Amount Store Price Amount

has

Title Price History
Title ID Effective Date
List Price Amount Store Price Amount

Exercise: time dependent data

Title

Title ID
- - - - - - - - - - - - - - - -
Name
Artist Name
Publisher Name
etc. etc. etc.

List Price Amount
Store Price Amount

Data modellers only know three numbers –
0, 1, and Many – but this model has two Price types

List and Store Prices are in
two places. It's unclear which
is current, previous, or future.

History
is mandatory

has

"Has" is not a descriptive relationship name

Title Price History

"History" is not good in an entity name.

Title ID
Effective Date
- - - - - - - - - - - - - - - -
List Price Amount
Store Price Amount

"Effective Date" should not be in the primary key –
it is business data so it can change.

If we change and correct a price on the
same date it is not clear which is which

There is no "End Date." There is only one
situation where you do not have to include an
End Date / Expiry Date / Last Valid Date.

Three solutions: time dependent data

Least desirable

Current price data

Title
Title ID
Bar Code Number
List Price Amount
List Price Date
Store Price Amount
Store Price Date
On Hand Quantity

Prev List Price	Prev Store Price
Title ID	Title ID
Prev List Price SID	Prev Store Price SID
Amount	Amount
Effective Date	Effective Date

Previous price data

A possible de-normalisation

Current price data

Title
Title ID
Bar Code Number
List Price Amount
List Price Date
Store Price Amount
Store Price Date
On Hand Quantity

List Price	Store Price
Title ID	Title ID
List Price SID	Store Price SID
Amount	Amount
Effective Date	Effective Date

Current & previous price data

Almost the best

No price data

Title
Title ID
Bar Code Number
On Hand Quantity

List Price	Store Price
Title ID	Title ID
List Price SID	Store Price SID
Amount	Amount
Effective Date	Effective Date

Current & previous price data

154

Best solution: time dependent data

The most general solution

No price data

Title

Title ID
Bar Code Number
On Hand Quantity

List Price

Title ID
List Price SID
Amount
Effective Date

Store Price

Title ID
Store Price SID
Amount
Effective Date

Two types of Price –
and *two* is not a number
Data Modellers recognise!

Price Type

Price Type Code
Description

Title

Title ID
Bar Code Number
On Hand Quantity

Title Price

Title ID
Title Price SID
Amount
Effective Date
End Date
Price Type Code (fk)

*Current & previous prices
of any type*

155

Future-proofing – "Avoid a fixed number of repeating attributes"

This model shows two types of Prices – List and Store.
Tomorrow a third will arise…
and a fourth and a fifth…

Data modellers only know
three numbers –
0, 1, and Many.
We don't recognise 2.
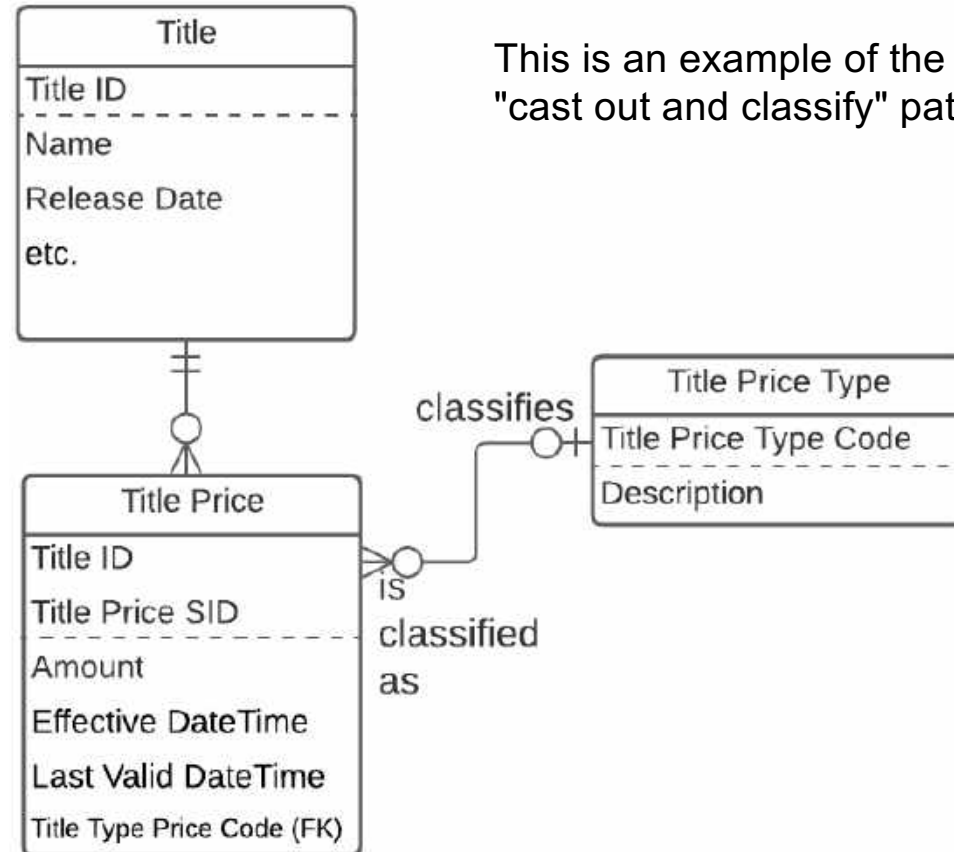
This revised model offers greater
flexibility and supports richer queries.

This is an example of the
"cast out and classify" pattern.

Two important time concepts

Logical Time	*Physical Time*

Logical Time

✓ Effective date/time, Start date/time, Begin date/time,etc.

✓ Time that data reflects the intent of the business at the time of update

✓ *Reality*

⚠ **Remember**
• **Can** be updated
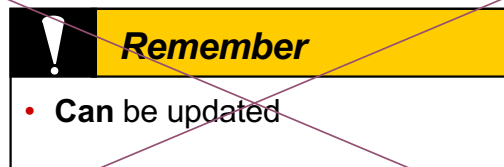
Wrong – with developments like Sarbanes-Oxley, we *don't change* stored data, we *add new* records.

Physical Time

✓ Recorded date/time, Transaction date/time, Update date/time,etc.

✓ Time when a record was written to the database

✓ *Representation*

⚠ **Remember**
• **Cannot** be updated

157
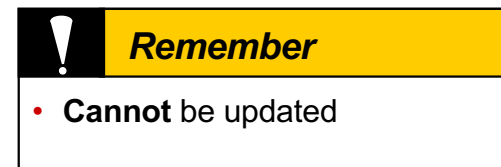
Change and correction

The situation…

Employees are given a credit limit, which is checked whenever they attempt to make a purchase to ensure that the purchase amount does not exceed the credit limit.

Purchases are approved or rejected based on the credit limit; a record of the transaction is always made.

The credit limit changes over time, and changes to the credit limit can be entered into the system before or after the effective date of the change.

Mistakes in entering data about credit limits are common, leading to frequent corrections. Currently, the business uses a "correct in place" approach and erroneous values are overwritten with correct ones.

The problem is that after an erroneous credit limit data has been corrected, it's often difficult to see in the data why a particular transaction was approved or rejected. We need a data model that will resolve this by recording pre-corrected and corrected data.

For example…

- Rochelle, a new employee, was given a credit limit of $5000, effective Sept 01 2019. This was entered into the system on Sept 04 2019, with an End Date of Aug 31, 2020.

- Rochelle's attempted $3000 purchase on Sept 03 2019 was rejected.

- A year later, Rochelle's credit limit was raised to $10,000 effective Sept 01, 2020 with no end date specified (that credit limit was to be in place indefinitely.) However, the new limit was mistakenly entered into the system as $100,000 on Aug 22, 2020.

- That error was corrected on Sept 15 2020, when the credit limit was revised to $10,000.

- Before the correction, Rochelle's $17,000 purchase on Sept 08, 2020 was approved.

- In late September of 2020, auditors were not able to understand why the $3000 Sept 20190 purchase was rejected, nor why the $17,000 Sept 2020 purchase was approved.

158

Change and correction solution

Employee

Employee ID
Hire Date
Termination Date
etc.

Employee Credit Limit

Employee ID
Emp. Credit Limit SID
Amount
Effective Date
End Date
Final End Date
Recorded Date
Corrected Date
…
audit attributes
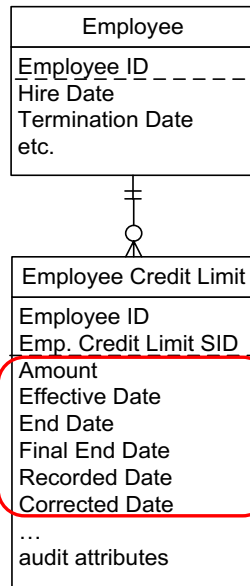
	Initial	*Erroneous*	*Corrected*
Amount	$5000	$100,000	$10,000
Effective Date	2019-09-01	2020-09-01	2020-09-01
End Date	2020-08-31	9999-12-31	9999-12-31
Updated End Date	null	null	null
Recorded Date	2019-09-04	2020-08-22	2020-09-15
Corrected Date	null	null **2020-09-15**	null
Notes	Effective and End Dates record business intent; Recorded and Corrected Date reflect database activity.	An open-ended record has an End Date set to a "high date" value. If a specific End Date is later set, it is recorded in Final End Date if you don't want to overwrite End Date.	The Corrected Date of the corrected (previous) record is set to the Recorded Date of this correcting record.

159

Four approaches to time dependent data

Show only the *current state* of the entities

1

- *No history, no future*

Show how the data *was recorded* at various times in the past

2

- *Shows database activity - physical DB update dates*
- *Done "by accident" in older systems that used system clock as effective date*

3

Show how the data *was intended* at various times in the past

- *Shows business intent - effective/end dates*
- *This is the most common approach*

4

Show both how the data *was intended*, and how the data *was actually recorded* at various times in the past

- *Shows intention and correction*
- *Supports "as of" reporting*

Time dependent data – key points

- Facts that change independently
  should be recorded independently

- Never name the entity "History" –
  it probably includes present and future values

- Distinguish between
  - *business* Effective Date
  - *database* Recorded Date

- It's tempting to put "Effective Date" in the key,
  but it might change, and might prevent two records from having
  the same Effective Date even when that makes sense

- Be sure to define what End Date / Expiry Date date *means*
  ("tot en met")

- Capture the need (the "reality") *first* in the model,
  then factor in performance considerations

- You might need to consider time zones
  - GMT / UMT / UTC
  - Local offset

This slide left blank in support of the pulp & paper industry

Rules on relationships and associations

➡	*Outline*

1. Interesting structures
2. Modelling time & history
3. Rules on relationships and associations
4. Presentation techniques for data modellers
5. Relating Dimensional and Entity-Relationship models

★	*Topics*

- V-A-K – using all the tools to build a model
- Multi-way associations and complex rules
- Advanced normal forms – resolving circular relationships and cyclic dependencies

!	*Themes*

- *Say* the requirement to be sure you understand it
- *Draw* schematics and examples

Four key points about complex associations

1. You can't tell whether a model is correct or not simply by inspecting it – you must have business involvement

   This gives rise to the other three points…

2. You must draw the model in a top-down fashion (or other systematic approach) so you can actually *see* dependencies

3. You must state your assumptions or understanding in narrative form as *assertions*, using terms (entity names, relationship names, and attribute names) from the data model

4. You must *illuminate* the data model by using sample data, schematic diagrams, scenarios, or some other understandable form

A quick exercise…

1. The company decides which items will be carried at which stockrooms.

2. The company qualifies suppliers to provide specific items.
   (A supplier can be qualified to provide multiple items, and an item may be provided by multiple suppliers)

3. The company enters into a contract with qualified suppliers for each item they will provide to a specific stockroom.

Will this model satisfy the business constraints?
If not, identify specific problems and develop a better model

Stockroom		Item		Supplier	
Stockroom ID _ _ _ _		Item ID _ _ _ _ _ _		Supplier ID _ _ _ _ _	

at   for   with

Supply Contract
Stockroom ID
Item ID
Supplier ID _ _ _ _ _ _
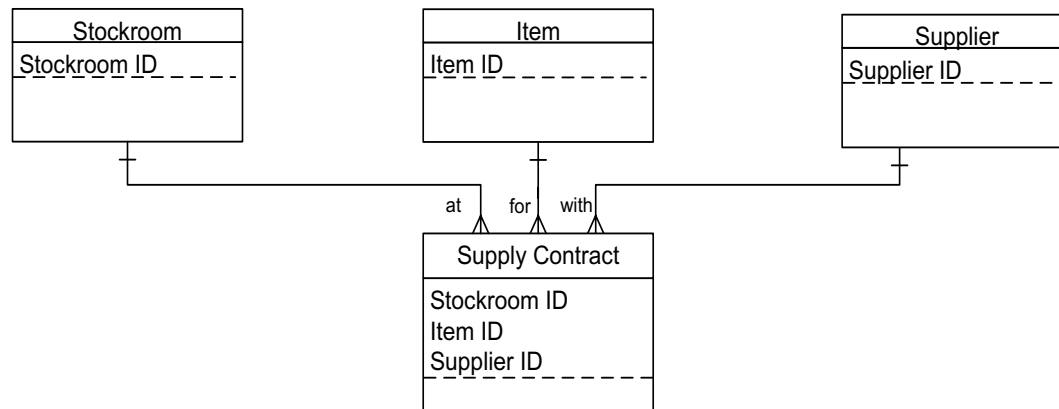
Can't record independent Supplier-Item relationship without including Stockroom – ~~"Stockroom #9999999" – a dummy Stockroom~~

Can't record independent Stockroom-Item relationship without including Supplier – ~~"Supplier #9999999" – a dummy Supplier~~

Insert anomalies

Delete anomalies

165

First, record independent relationships
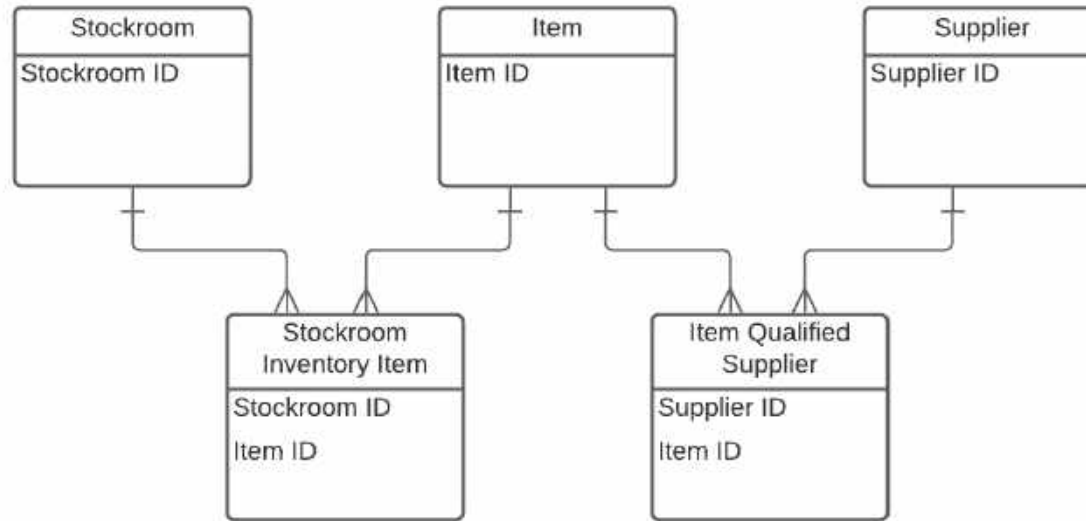
Associate the associatives

Stockroom

Stockroom ID

Item

Item ID

Supplier

Supplier ID

Stockroom Inventory Item

Stockroom ID

Item ID

Item Qualified Supplier

Supplier ID

Item ID

Supply Contract

Stockroom ID

Item ID

Supplier ID

Item ID

Supply Contract

Stockroom ID

Item ID (fk1, fk2)

Supplier ID
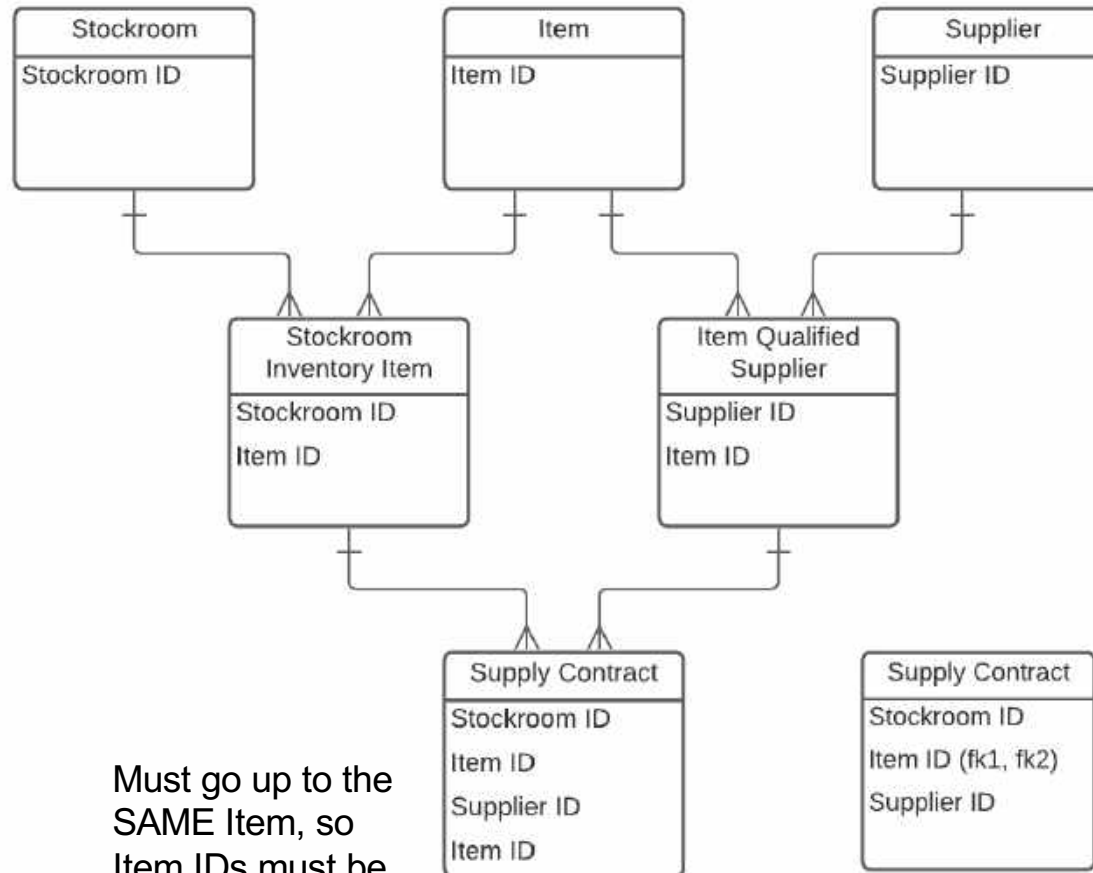
Must go up to the SAME Item, so Item IDs must be the same!

Or use one Item ID as part of two foreign keys

167

Fourth Normal Form and Fifth Normal Form

4NF and 5NF are violated when a 3-way or higher order associative entity should be broken down and made more granular.

X   Y   Z

XYZ   ?

Fourth and Fifth Normal Form are all about making multi-way associatives (three or more parents) more granular, if necessary

Customer   Vehicle   Salesperson
A   B   C

to   of   by

This might be valid...   ABC   ... or it might be a

Sale

4NF or 5NF Violation? Spoiler alert - it doesn't matter!

All that matters is that the multi-way associative needed to be more granular.
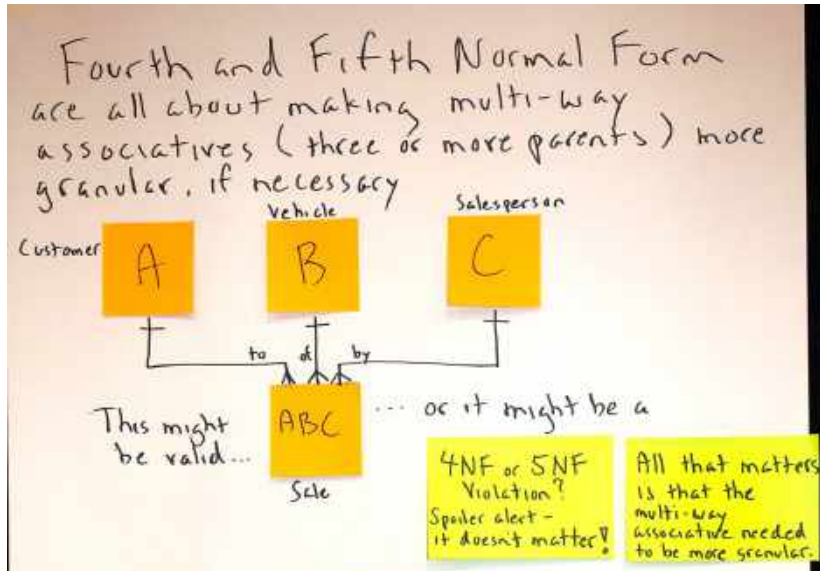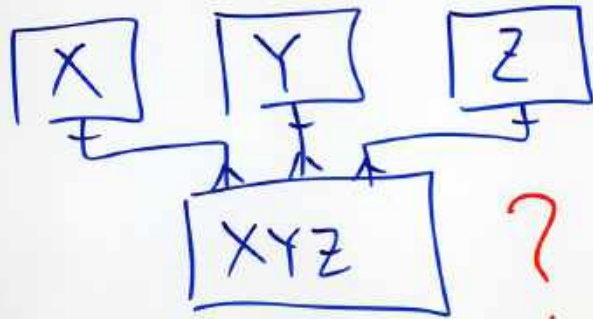
More possibilities…
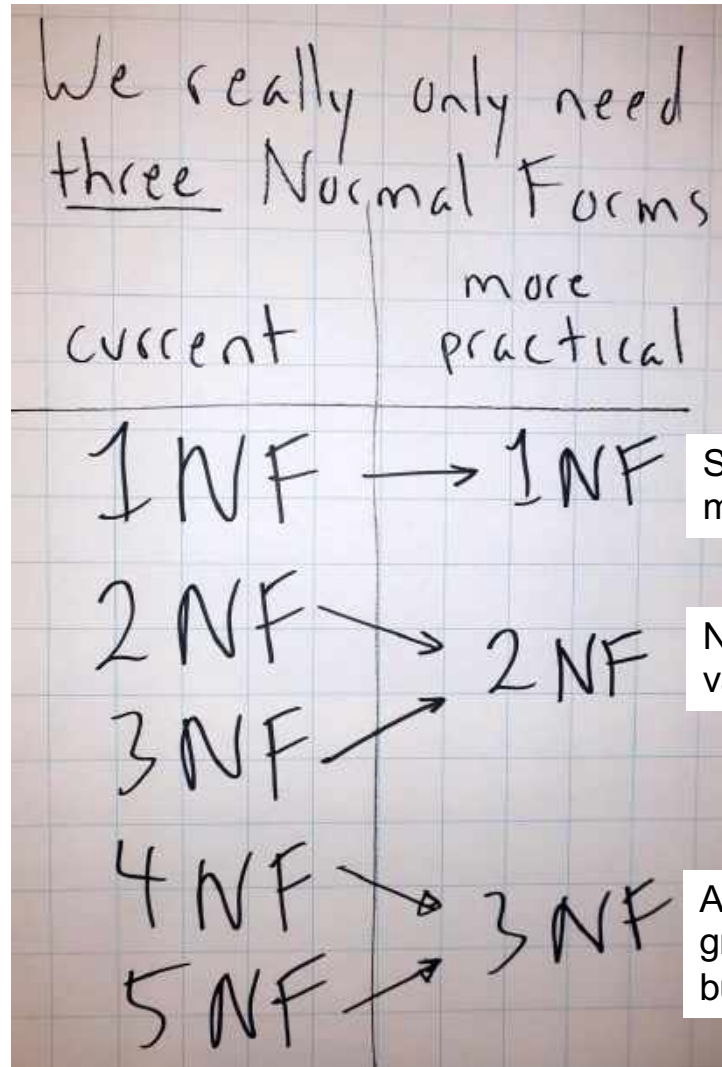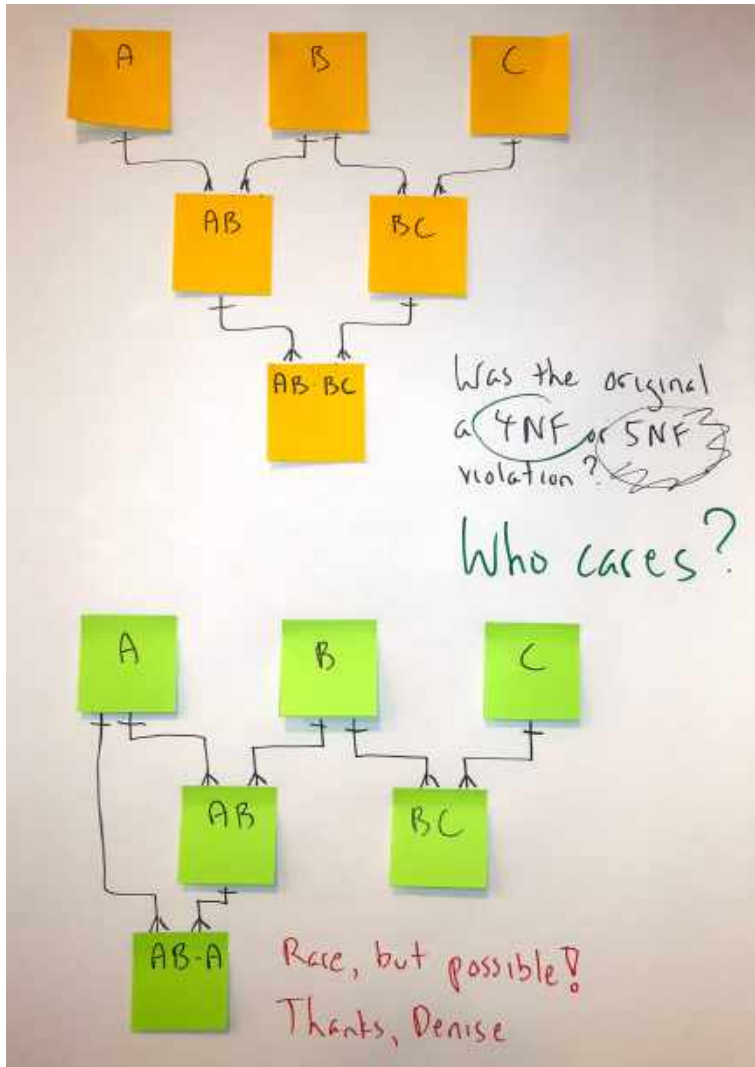
Skip to 174

Single-valued attributes make data more "reportable."

Non-redundant – each attribute value is recorded only once.

Associative entities are as granular as necessary to reflect business rules.

169

4th Normal Form

- 4NF - "Primary Key cannot contain 2 or more independent, multivalued attributes of another entity"

- The classic example:
Employees may have Skills and/or Languages

This version is incorrect, because Skill and Language are independent

This version is correct

170
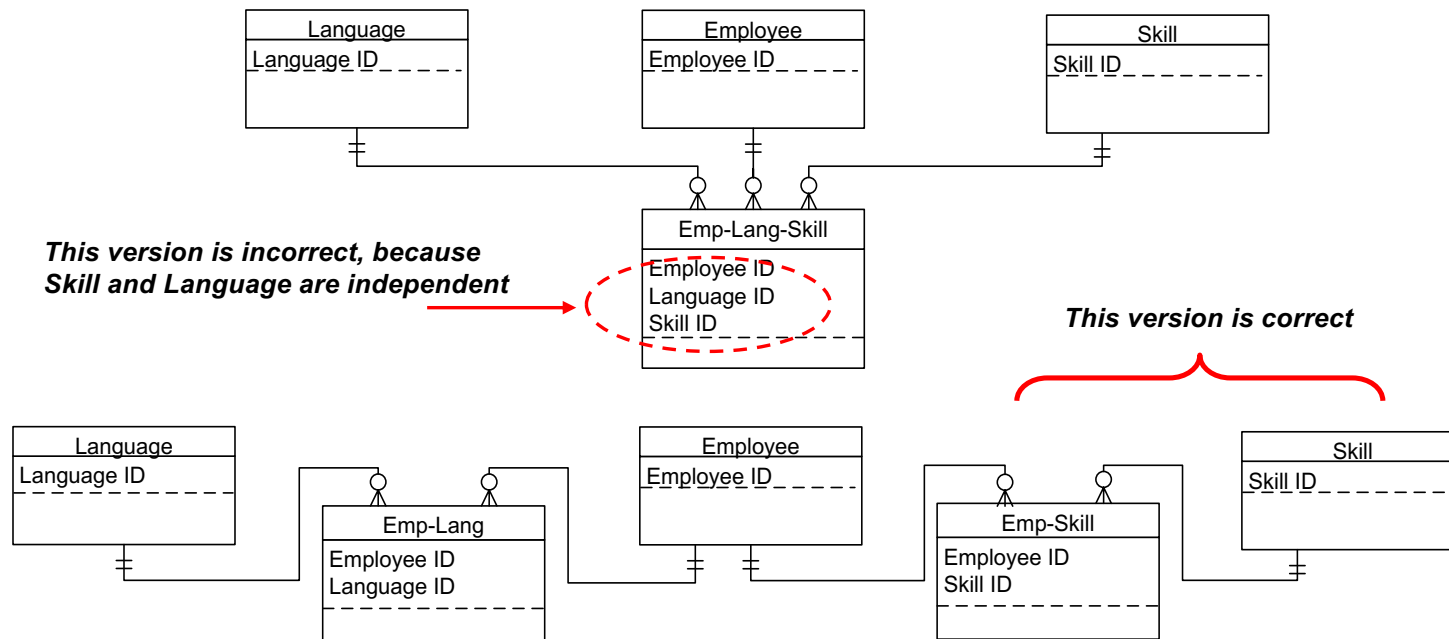
5th Normal Form

- How we model three or more related
   entities depends on the rules

- Agents represent Manufacturers in Regions -
   if any combination is valid, the model to the
   right is fine

- What if there are *additional* constraints?

   – "business rules"

   – only certain combinations are valid
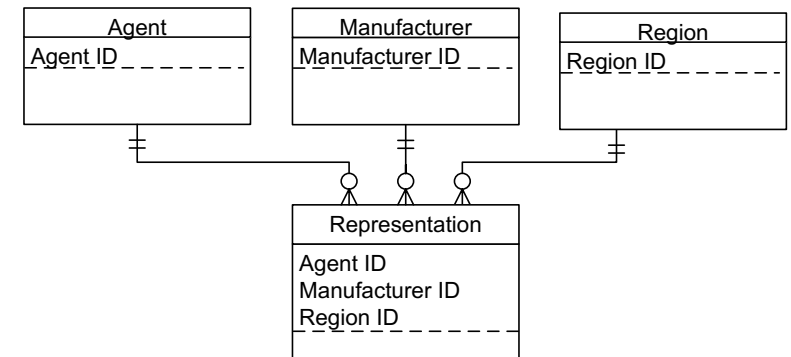
5th Normal Form

- ## Assume the following constraints:

  – Agents only represent certain Manufacturers

  – Manufacturers only distribute in certain Regions

  – Regions are only covered by certain Agents

- ## Now we have a *"cyclic dependency"* within the key of Representation

  – violates 5NF
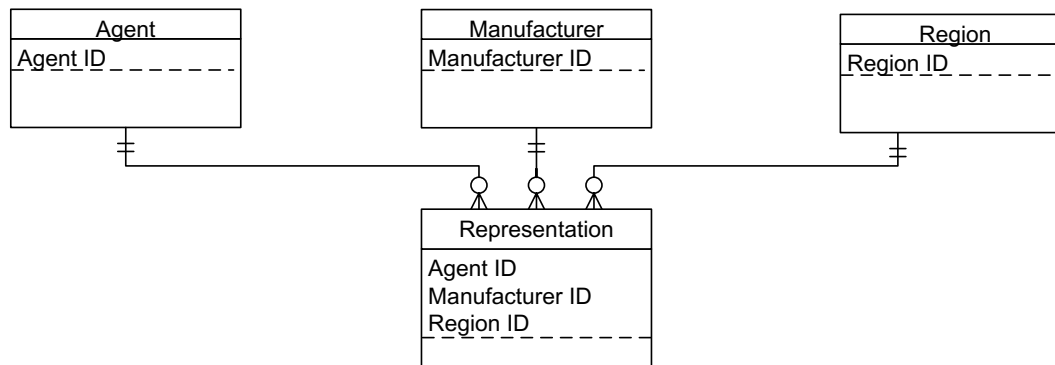
"Cyclic dependency":
Agents are related to Manufacturers,
Manufacturers are related to Regions,
and Regions are related to Agents

172

Self-study exercise: Fifth Normal Form
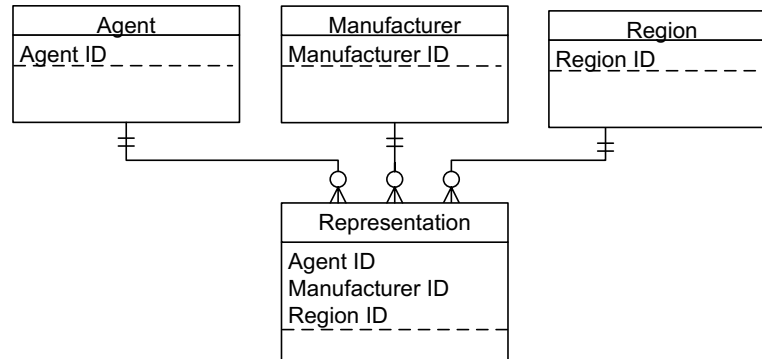
The problems with the above example include:

- Certain combinations will be recorded redundantly. For instance, if Bob-GM-Central and Bob-GM-Eastern are valid combinations, then Bob-GM is recorded twice

- Some valid two-way combinations won't be recorded at all if they aren't part of a valid three way combination. For instance, Joe-Central may be valid, but if Joe doesn't represent a Manufacturer who sells in Central, Joe-Central won't get recorded

In other words, it doesn't accurately reflect the rules or facts

Over ➡

Self-study exercise: Fifth Normal Form

Redraw the Agent - Manufacturer - Region model assuming the following

Agents only represent certain Manufacturers

Manufacturers only distribute in certain Regions

Regions are only covered by certain Agents

When an Agent represents a Manufacturer, and that Manufacturer distributes in a Region, and the Agent covers that Region, then the Agent represents the Manufacturer in that Region

Now, redraw the model assuming the following changes to the rules

Agents only represent certain Manufacturers

Manufacturers only distribute in certain Regions

The Manufacturer decides which of their Agents will cover which of the Manufacturer's regions

Solutions: Fifth Normal Form

Agent
Agent ID

Agent-Mfr
Agent ID
Manufacturer ID

Manufacturer
Manufacturer ID

Mfr-Region
Manufacturer ID
Region ID

Region
Region ID

Agent-Region
Agent ID
Region ID

Agent
Agent ID

Agent-Mfr
Agent ID
Manufacturer ID

Manufacturer
Manufacturer ID

Mfr-Region
Manufacturer ID
Region ID

Region
Region ID

Constraint:
Both Manufacturer IDs
must be the same

Agent-Mfr-Region
Agent ID
Agent-Mfr Manufacturer ID
Mfr-Region Manufacturer ID
Region ID

or, ...

Agent-Mfr-Region
Agent ID
Manufacturer ID (FK1, FK2)
Region ID

175

Self-study exercise: Folsom Foundries

FF is a casting and machining shop that makes a variety of industrial products, such as valves and hydraulic cylinders.

Some products are in turn made up of other products - a hydraulic switch body, for instance, contains a number of hydraulic cylinders that are available separately as well.

As is often the case, a product like a hydraulic cylinder may be used in many other products, for instance in the 4-way hydraulic switch and the 8-way hydraulic switch.

They are all treated by the company as "products", with the only difference being that some products are manufactured by FF, and others are obtained from suppliers (none come from both sources.)

FF is only interested in the makeup of the products it manufactures and the operations that must be carried out in order to make them. A product manufactured by FF could be made up of either manufactured or supplied products.

FF has decided that it needs to do a better job of planning its manufacturing operations, and the scheduling of production to satisfy customer orders.

When a product is being manufactured, it goes through a strict sequence (step 1, step 2, step 3, etc.) of standard operations such as casting, welding, drilling, de-burring, assembly, test, plating, painting, or polishing. FF has a number of workstations, each of a particular type, such as an Emco Drill/Punch. Because of standardisation, an operation can only be carried out at one type of workstation, but a particular type of workstation may support multiple operations.

Over ➡

Self-study exercise: Folsom Foundries

Customers place orders, which (like all orders in all data modelling exercises) can contain multiple order lines, each specifying a desired quantity of a particular product. Customer orders can request both the products that FF manufactures, and those that FF obtains from suppliers.

Production Planning schedules a "lot" (also known as a "batch" or a "run") of a single product when it needs to satisfy some orders . A lot may consolidate many order lines, and an order line may be split across various lots. Each lot has a scheduled start and finish date, and a quantity.

Production Planning schedules a "lot" (also known as a "batch" or a "run") of a single product when it needs to satisfy some orders . A lot may consolidate many order lines, and an order line may be split across various lots. Each lot has a scheduled start and finish date, and a quantity.

Where it gets tricky is when Planning starts to allocate the production steps for a lot across the available workstations. A step in the manufacturing process for a lot may be split across several workstations, each with different planned dates/times and quantities.

The assignment:
1. Draw a contextual data model (approx. 5 subject areas)
2. Identify the main entities within each,
   along with an approximate definition and main facts
3. Lay out an initial ER diagram according to dependency
4. Finish the initial ERD, but don't worry about attributes and keys, except those attributes that will be necessary to state important constraints.

Solution - Folsom Foundries
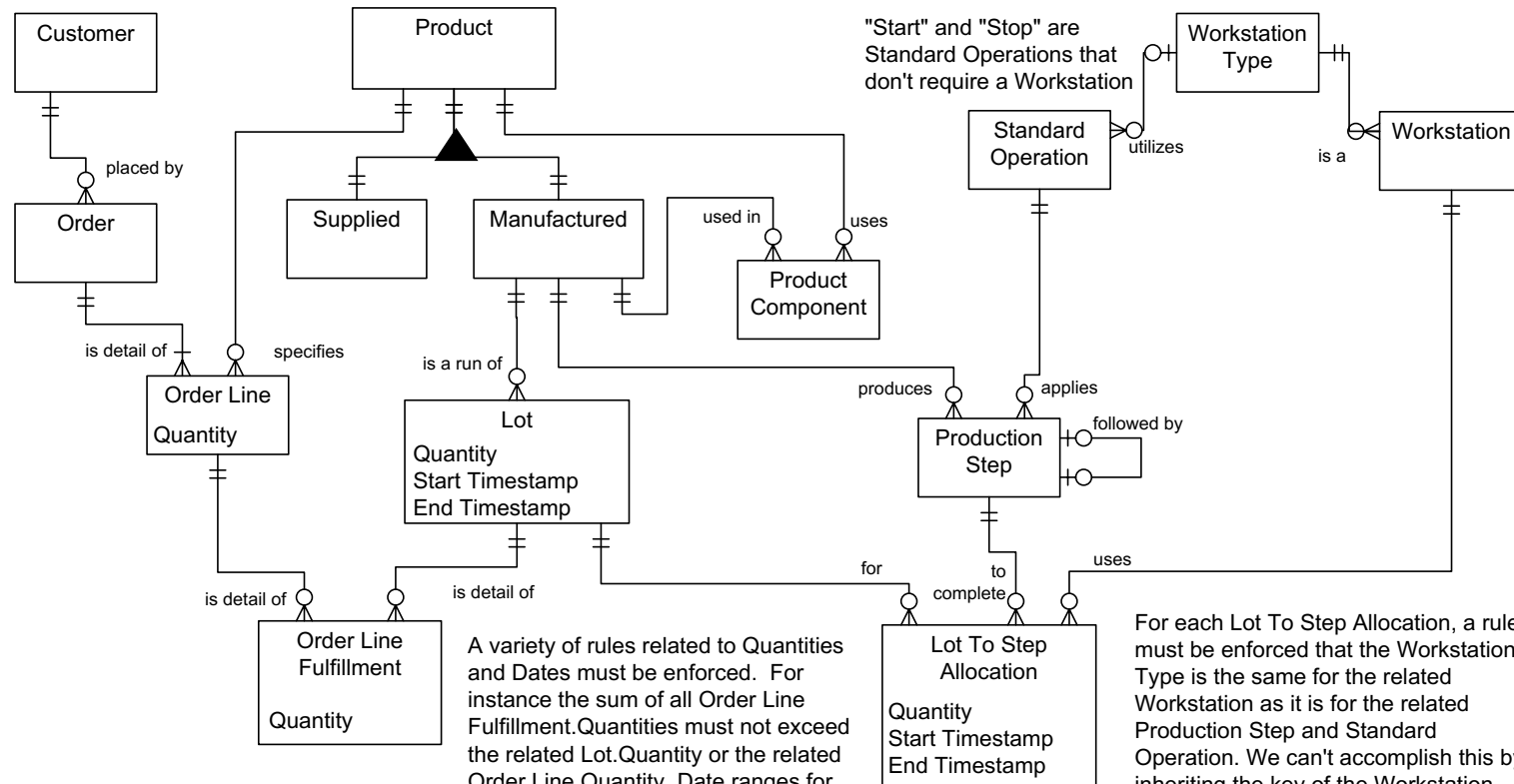
"Start" and "Stop" are Standard Operations that don't require a Workstation

A variety of rules related to Quantities and Dates must be enforced. For instance the sum of all Order Line Fulfillment.Quantities must not exceed the related Lot.Quantity or the related Order Line.Quantity. Date ranges for the Lot To Step Allocation must be within the Date Range for the Lot.

For each Lot To Step Allocation, a rule must be enforced that the Workstation Type is the same for the related Workstation as it is for the related Production Step and Standard Operation. We can't accomplish this by inheriting the key of the Workstation Type because the relationship from Strandard Operation to Workstation Type is optional.

178

Presentation techniques for data modellers

➡	*Outline*

1. Interesting structures
2. Modelling time & history
3. Rules on relationships and associations
4. **Presentation techniques for data modellers**
5. Relating Dimensional and Entity-Relationship models

★	*Topics*

- A learning experience
- The rhetorical context and the storyboard
- A demonstration
- Five key techniques

!	*Themes*

- Context – your audience, and why the model matters to them
- It's a story, not a data model!
- It's marketing, not a data model!
- A few key techniques, not hundreds of "tricks"

Presentations – my "new Customer data model" experience…

Road show version 1

"Here's the new Customer model. How do you like it?"

"So what?" "Obvious!" "Yawn." "ZZZZZZzzzzzzzz…."

- -

VP of IS:

"You're dyin' out there, kid! I want you to drag them through all of the pain and misery of our current files and databases. *Then* show the new model."

- -

Road show version 2

"Let's try answering some important questions using the current model, and then the new model"

"Fantastic!"

"When do we get it?"

"Do you need funding? We can help!"

180

First lesson: know the "rhetorical context"

✓A framework for alignment

- known: brainstorm, reduce, prioritise
- unknown: research (survey, ask, …)

Audience

✓preferences
✓experiences
✓knowledge
✓biases

Purpose

✓your objective
✓their objective

Occasion

✓mood
✓time
✓recent events
✓environment
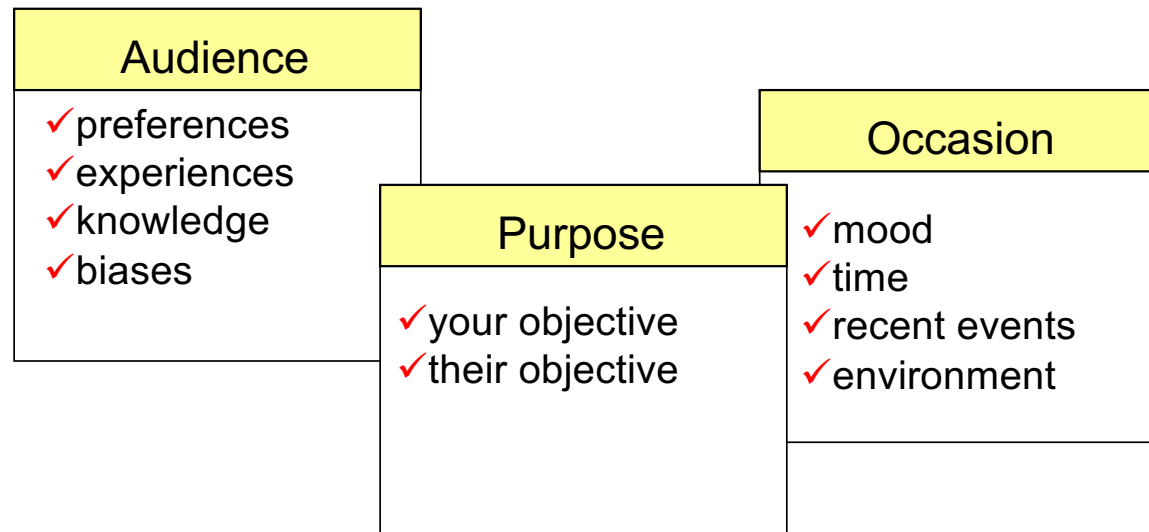
It's a story, so storyboard it

Business issues have changed…

… and therefore systems have changed…

Details
Operational to informational

Distributed, component-based

Cross-functional

…so methods for building have changed

Making these changes will be difficult…

…but it is vital to our survival
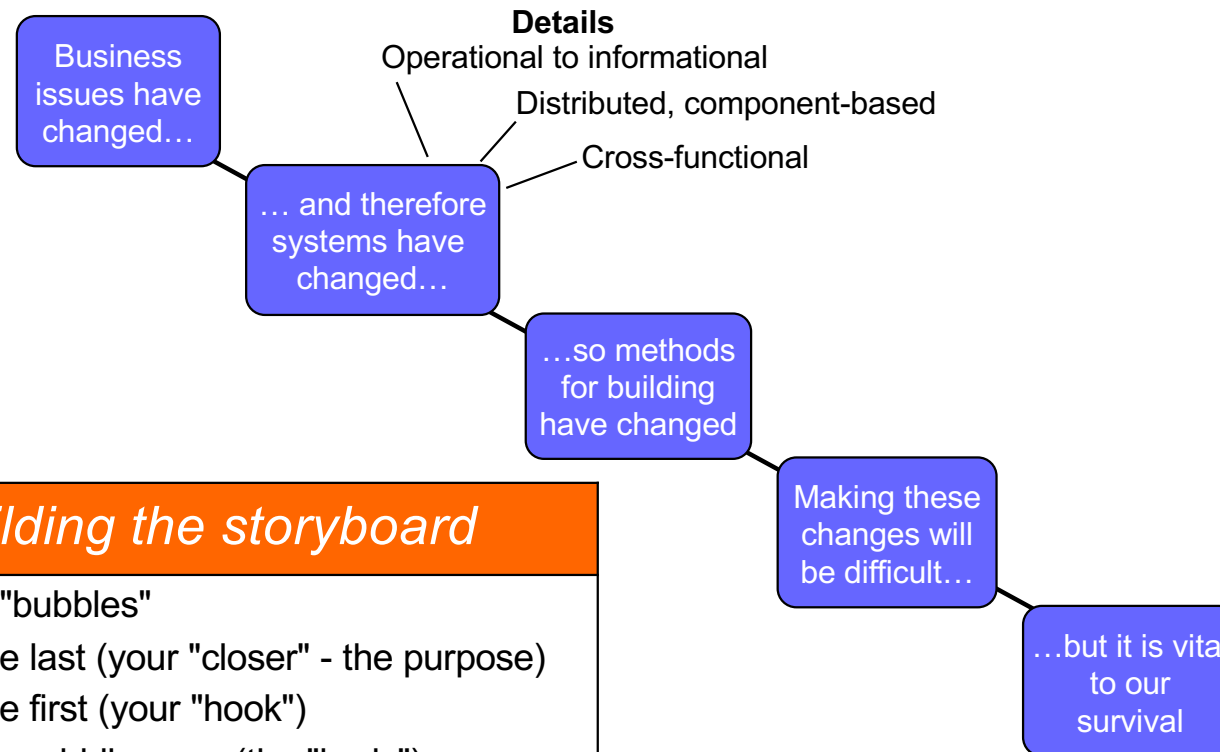
Building the storyboard

1. Draw 5 "bubbles"
2. Fill in the last (your "closer" - the purpose)
3. Fill in the first (your "hook")
4. Fill in the middle ones (the "body") – add or subtract bubbles as needed
5. Allocate details to bubbles
6. Iterate until it flows and builds properly

Only include detail that matters!

Presenting data models

Try not to call it a *data model*
- I often call it a "world view"
- Or… "This is how Application XYZ sees the world."

Start simple, and add details in layers
- Begin with two or three fundamental things
- Work "across" the model, not a "deep dive" in one area
- Draw the model on a whiteboard as you speak to it
- Save detail like optionality until later, and primary/foreign keys until *much* later

Speak exclusively in the language of the *business*
- Don't use terms like entity, relationship, attribute, optionality, supertype, subtype, recursion, etc. Remember, you're describing a *business,* not a *database.*
- Point to the relevant entity while addressing a concept
- Someone overhearing your presentation should not realise you are presenting a data model

Make it *real*
- Back it up with sample data, queries, and scenarios
- Identify specific business issues or opportunities, and show how the data model helps

We'll now walk through a successful data model presentation, followed by a discussion of key points

183

Storyboard for the "Flight Data Model" presentation
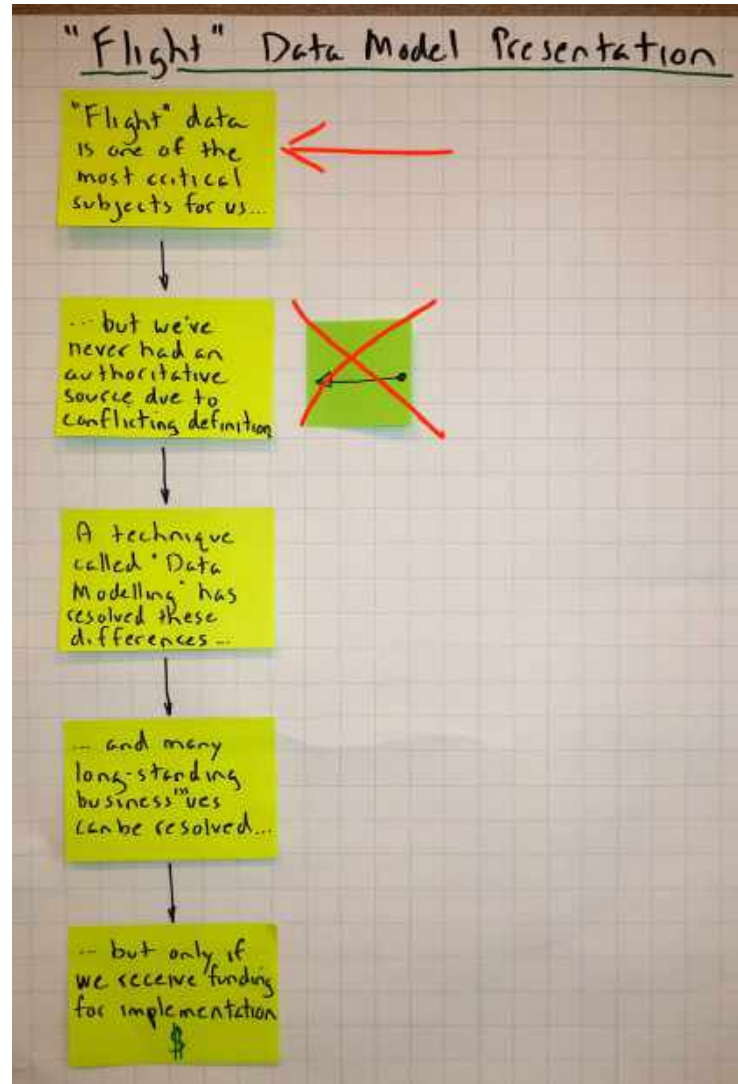
"Flight Data Model"

"Flight Data Model"

186

BODM-MC:
Business-
Oriented
Data Modelling
Masterclass

"Flight Data Model"

"Flight Data Model"

"Flight Data Model"

It became a 25-entity concept model, and the businesspeople loved it!
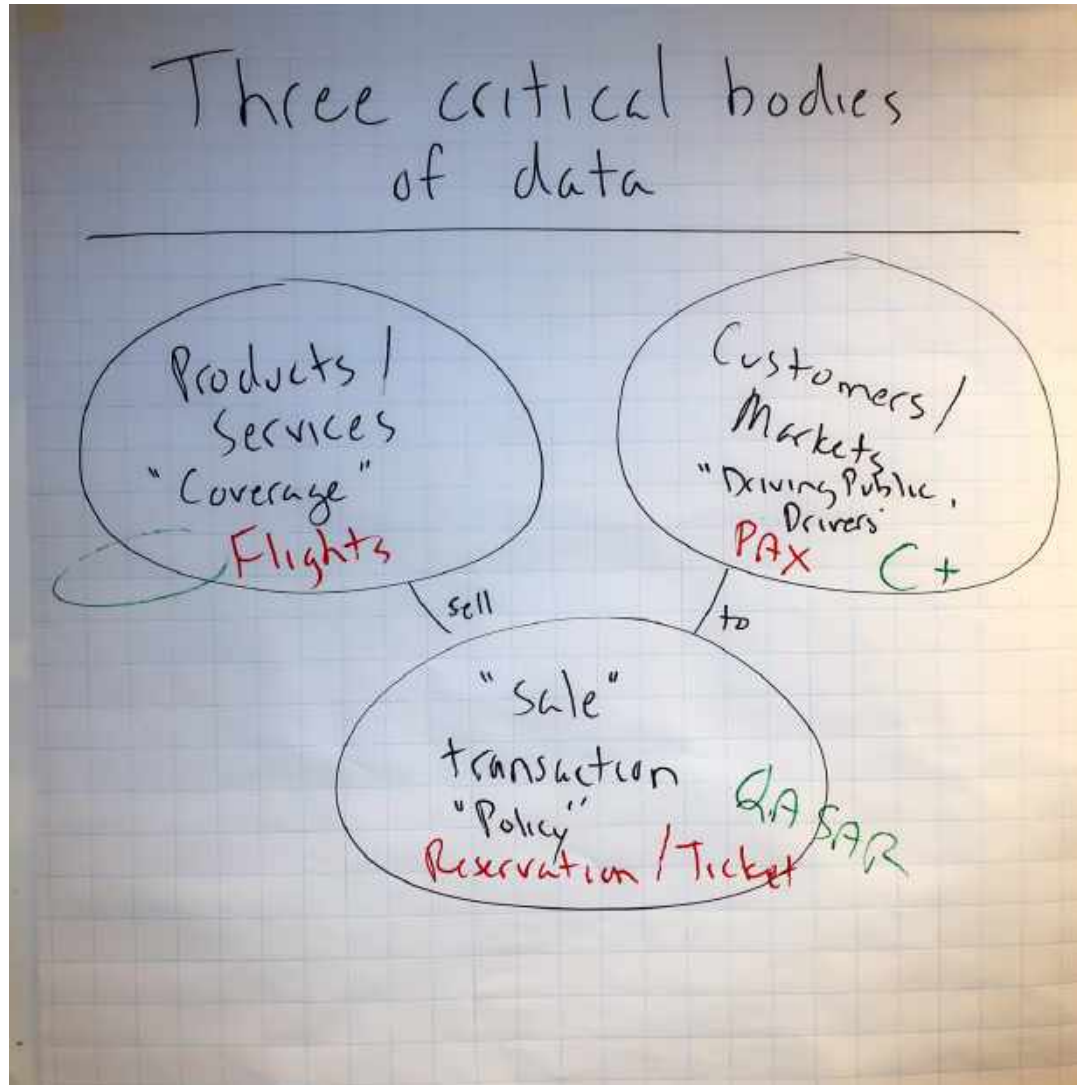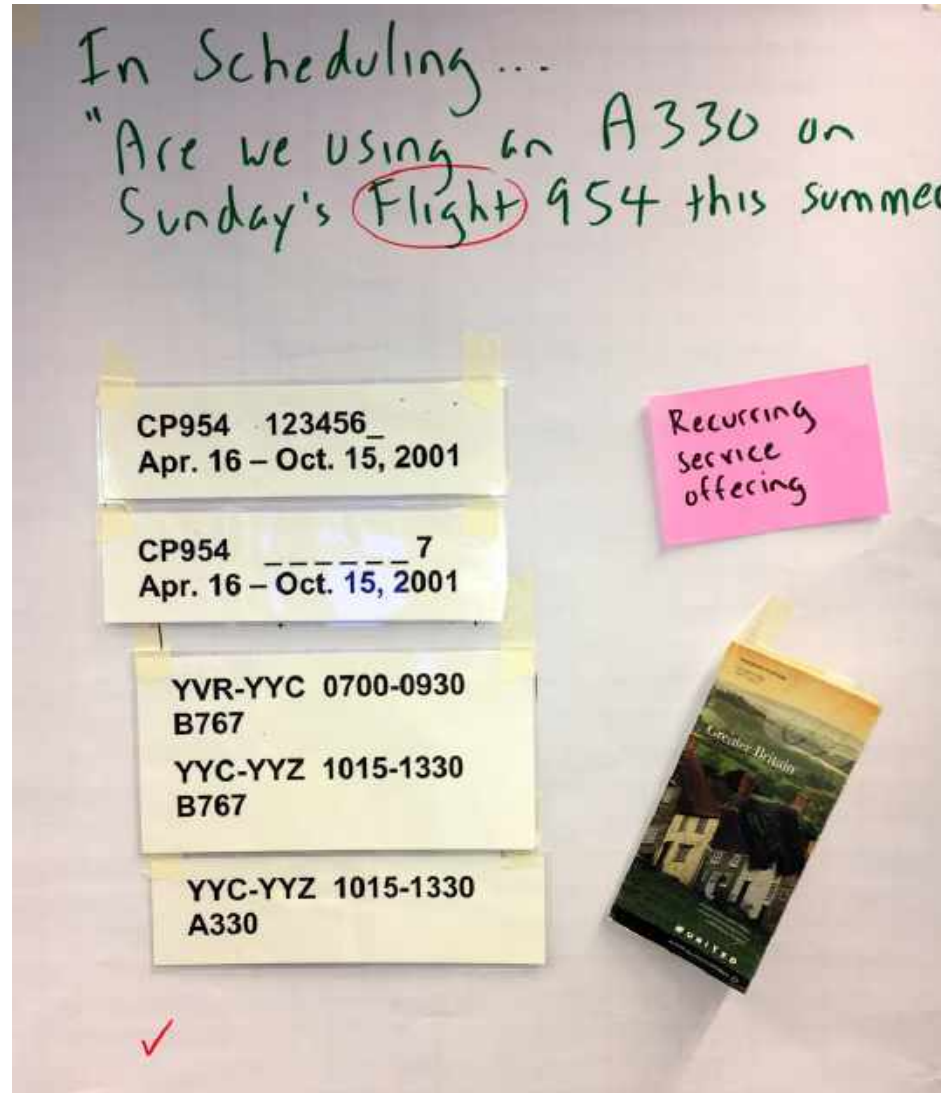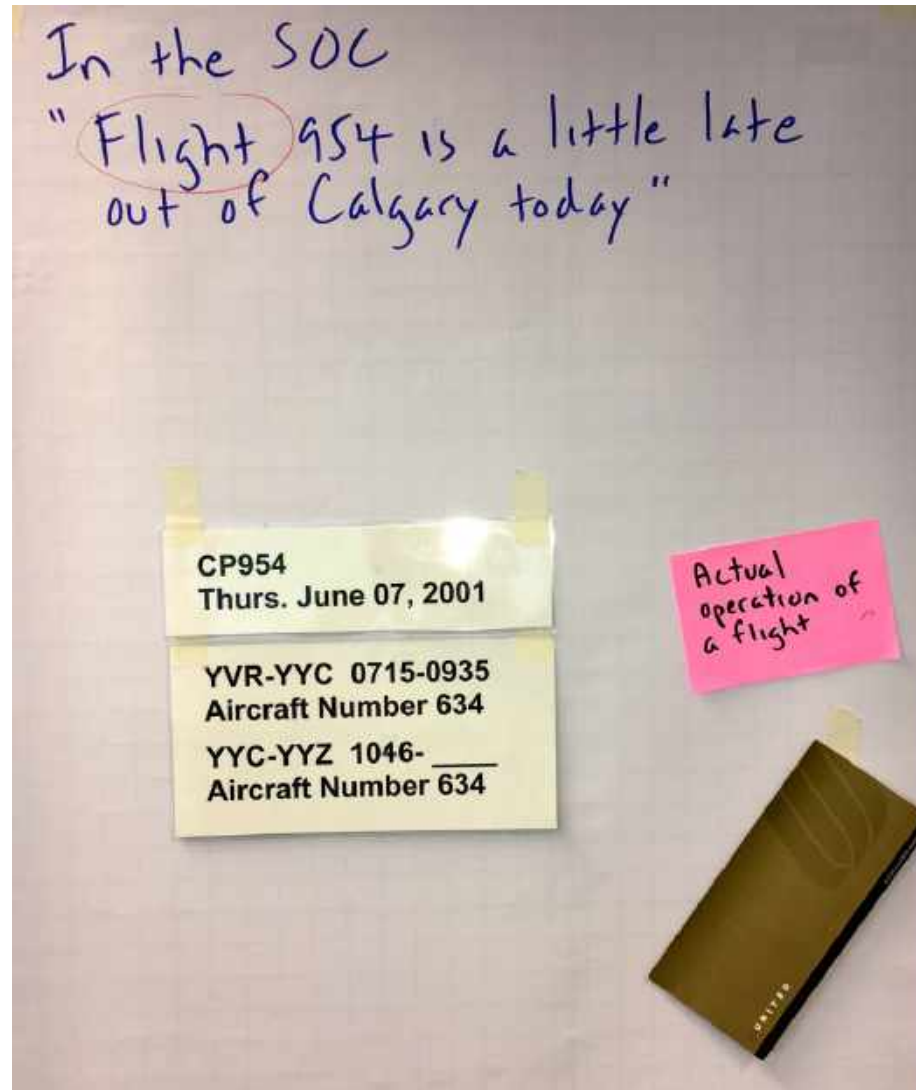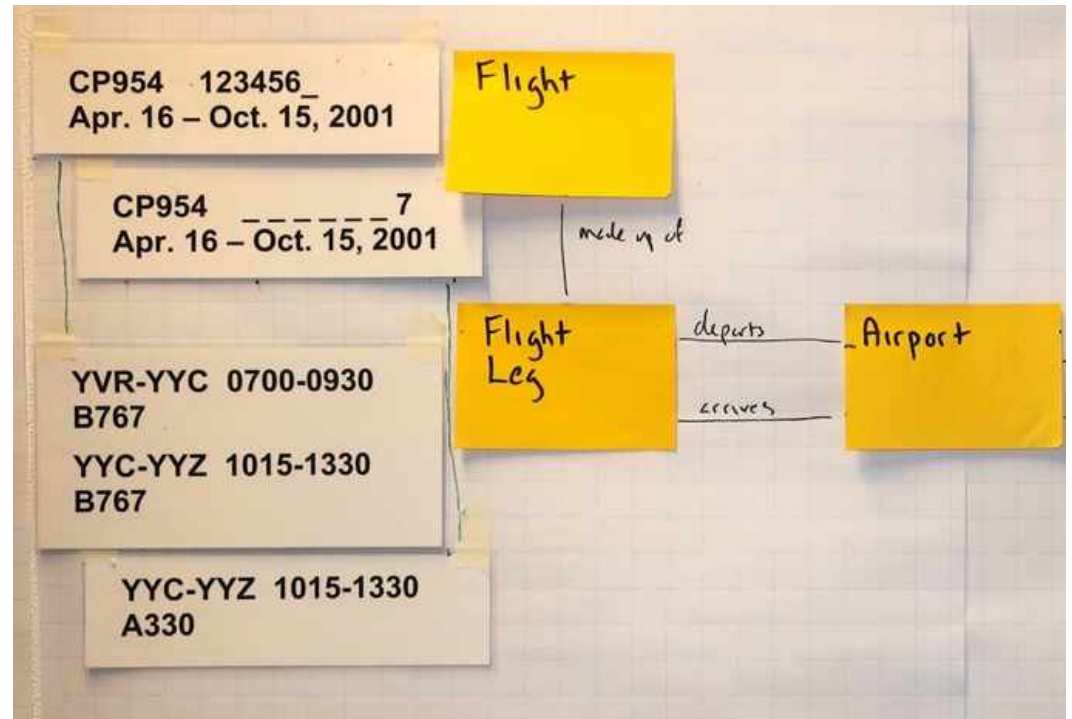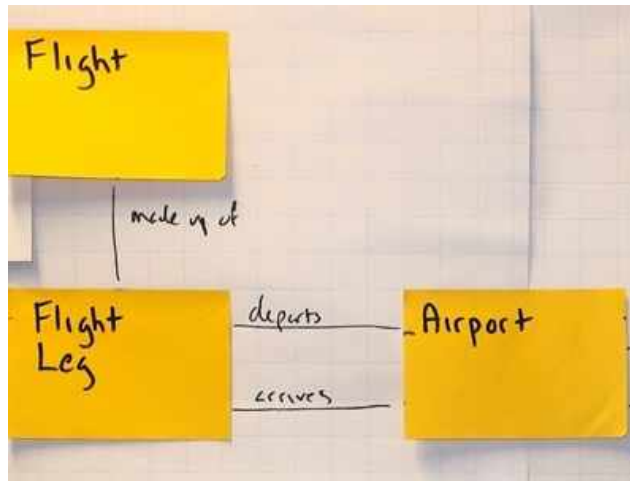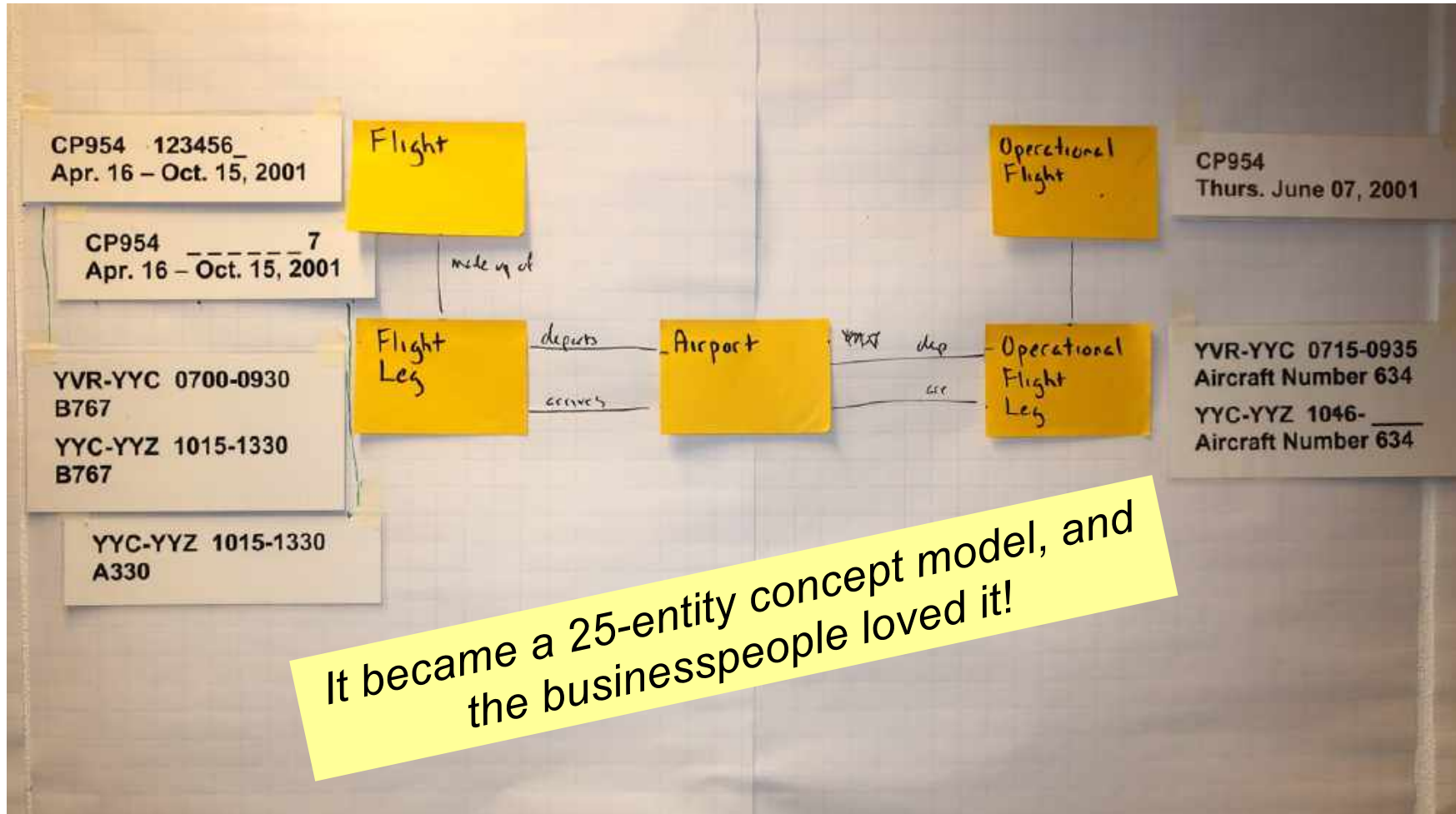
The five techniques that really matter

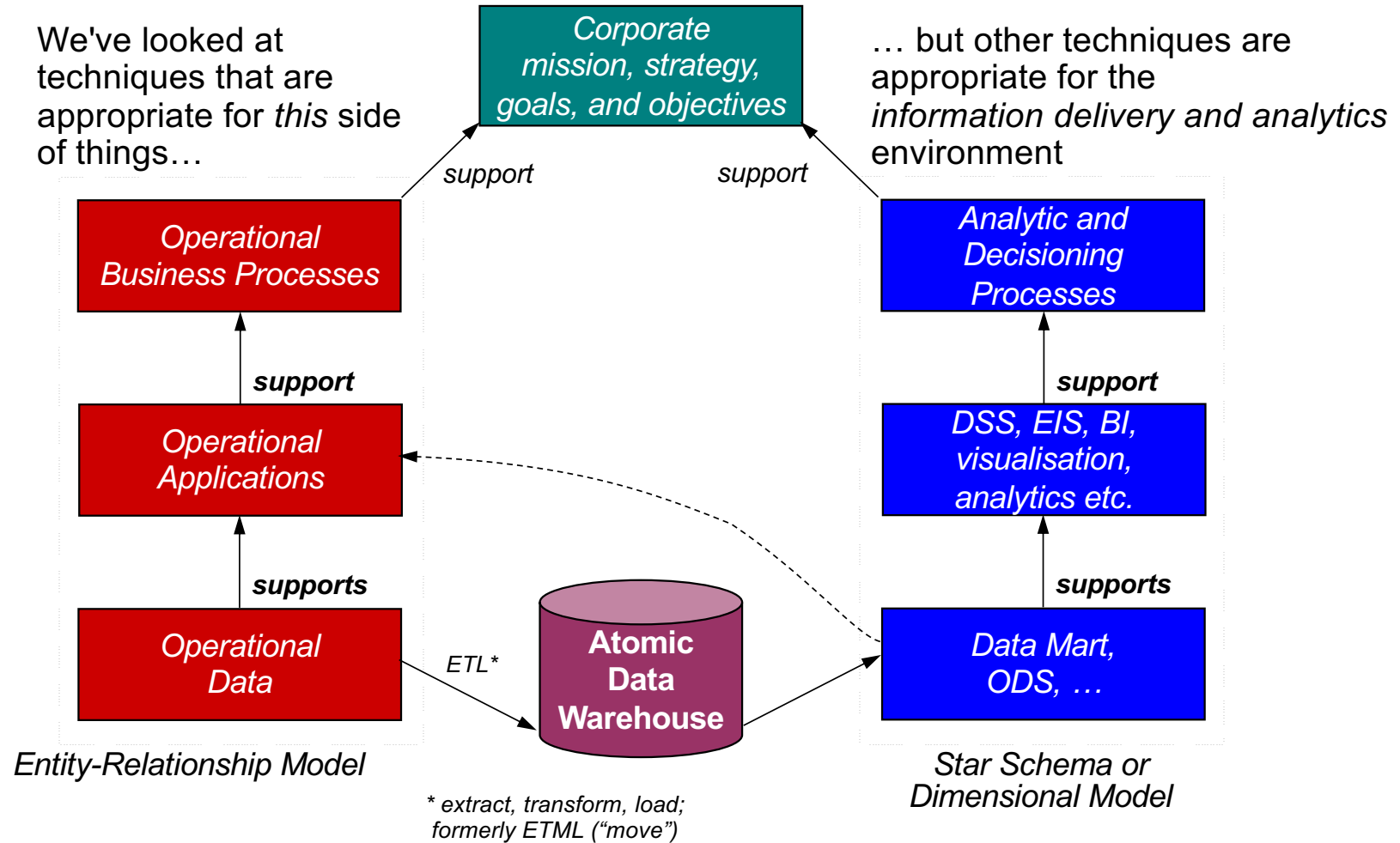| Technique	Why?	How?
1 Organise their minds to receive the presentation	· Otherwise, you're just "noise" · "Why is this person telling me these things?"	· "Here's the point I want to make." · "This is why you care, and how I know." (even if it's obvious) · "These are the caveats and limitations." · "This is how I'll make my point." (storyboard!)
2 Big picture first	· Provides context and perspective · Makes subsequent detail understandable	· Show contextual data model first, build up detailed models later · Process context first, process flow later · Describe 5 problem areas first, specifics of each area later
3 Do it live	· Focuses, demands that they watch · Involves them / you · It means 'attending has value'	· Use memory triggers, not a script · Build up content progressively on white board, flip chart, or screen · Add brainstorming, discussion, or questions · Have them physically "do stuff"
4 Present information in various forms	· Adds interest · Different forms have different strengths	· Supplement PowerPoint slides with flip charts, white boards, Post-Its, handouts, etc. · Use props – the thing itself, not a description · Use visual, auditory, and kinesthetic approaches
5 Show, then tell	· Point is more meaningful if experienced firsthand · Saves time, simplifies	· Scenario / example first, then concept / abstraction · Problem first, solution second · Thing first, description / discussion second

190

Relating dimensional and ER modelling

➡	*Outline*

1. Interesting structures
2. Presentation techniques for data modellers
3. Modelling time & history
4. Rules on relationships and associations
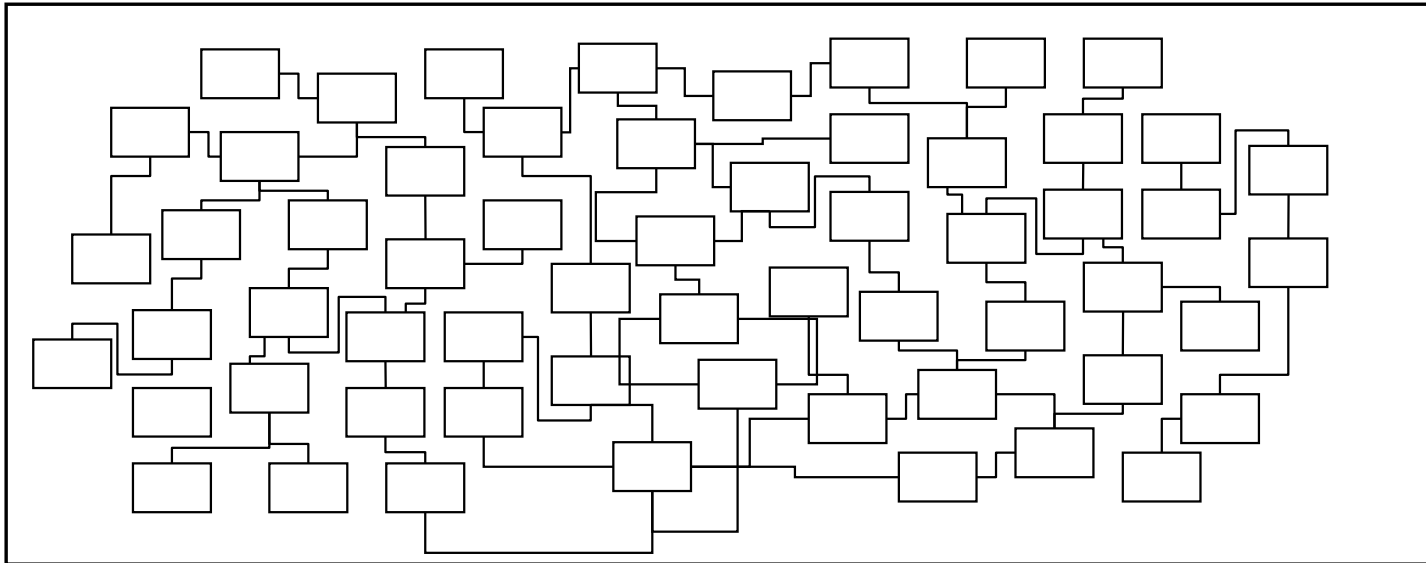5. Relating Dimensional and Entity-Relationship models

★	*Topics*

- Dimensional models – application and concerns
- The basics – facts, dimensions, measures, and attributes
- Relationship between ER and dimensional models
- Exercise – developing a basic dimensional model

!	*Themes*

- Way too much mystery and confusion
- The ERD is still essential
- Again – the "4 Ds" help

Two sides of the house

We've looked at techniques that are appropriate for *this* side of things…

Corporate mission, strategy, goals, and objectives

… but other techniques are appropriate for the *information delivery and analytics* environment

support   *support*

Operational Business Processes

support

Operational Applications

supports

Operational Data

Analytic and Decisioning Processes

support

DSS, EIS, BI, visualisation, analytics etc.

supports

Data Mart, ODS, …

ETL*   **Atomic Data Warehouse**

Entity-Relationship Model

* extract, transform, load;
formerly ETML ("move")

Star Schema or Dimensional Model

192

Oh-oh…

A detailed data model might be too complex to present to business folks for query, OLAP, BI, etc.

Dimensional models

- ✓ Used to model and implement data structures for various types of business intelligence tools.

- ✓ One or more dimensional models per warehouse model

- ✓ We'll use the terms dimensional model and star schema interchangeably

- ✓ Any combination of dimensions can be used in a query

  - the same dimension will appear in many dimensional models
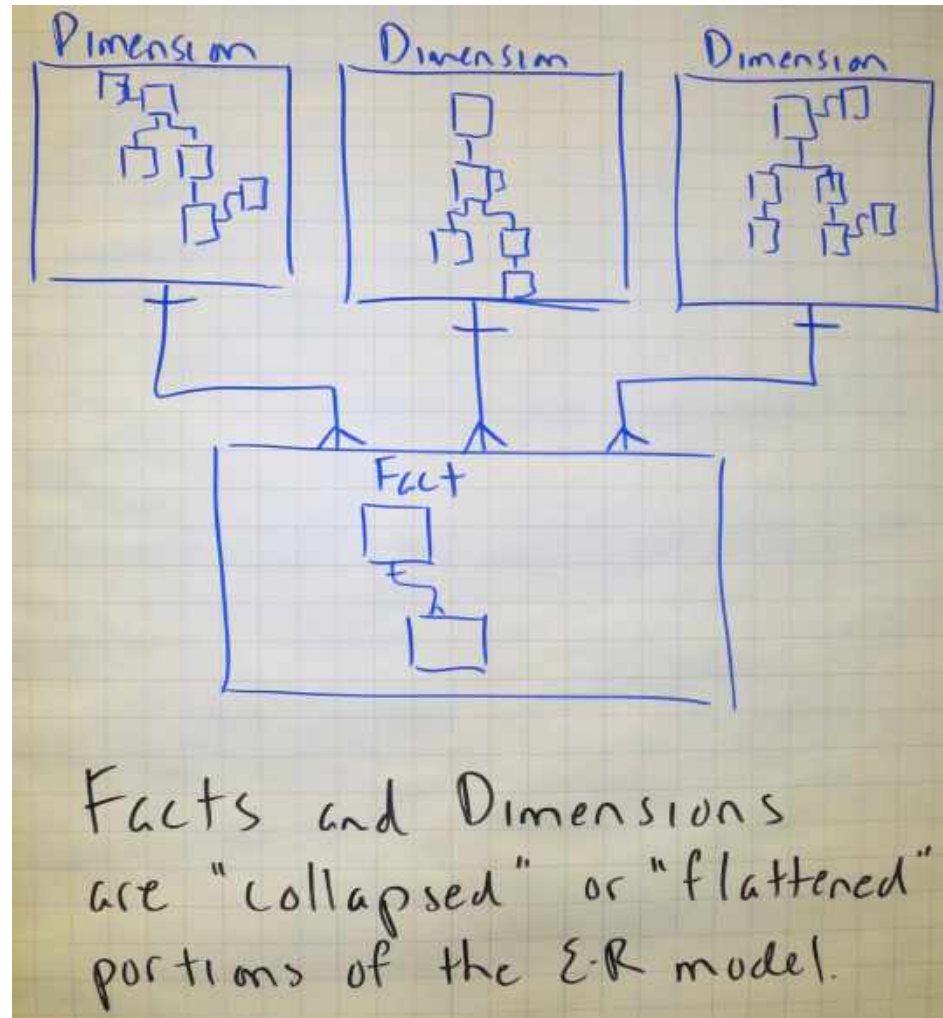
  - should be managed as "shared dimensions"
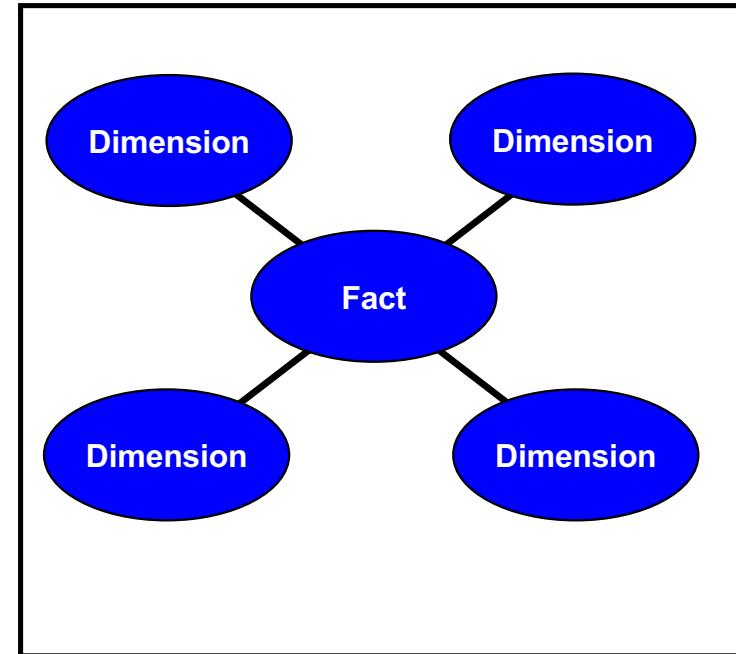
Dimensional model concepts

"Facts"	*"Dimensions"*

"Facts"

- ✓ the central thing you want to count or measure

- ✓ has a count, usually "1"

- ✓ often details of a transaction or other core Associative entity (e.g., Sale, Shipment, Crime, Claim, …)

- ✓ can have attributes, but when they apply to a Fact they are called *measures* (e.g., Sale has Total Amount, Time, Payment Method)

"Dimensions"

- ✓ how you want to organise or summarise the facts

- ✓ often a Type or Kernel entity (e.g., Region, Time Period, Product, Customer, …)

- ✓ can have *attributes* (e.g., Product has Category, Price, and Colour)

195

Dimensional model – example

✓ The *Fact* is usually an associative or characteristic entity from somewhere quite "low" in the ERD

✓ The Fact will usually include a "count" of something, even if the value is implicitly "1"

- E.g., "dollars" or "hours" or "units"

✓ The *Dimensions* are "clusters" of the Fact's parents, grandparents, etc. entities

✓ Any combination of Dimensions can be used in a query

- the same Dimension will appear in many dimensional models

- should be managed as "shared Dimensions"

196
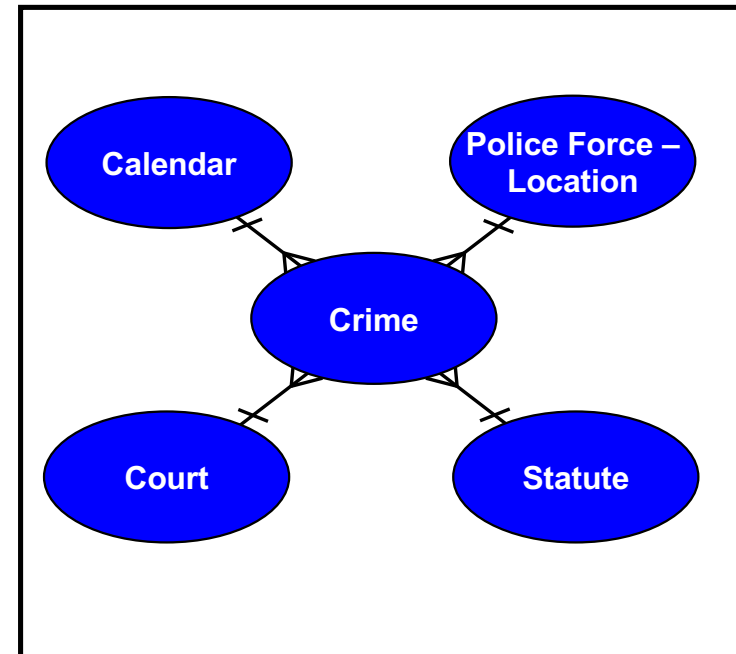
The classic methodology

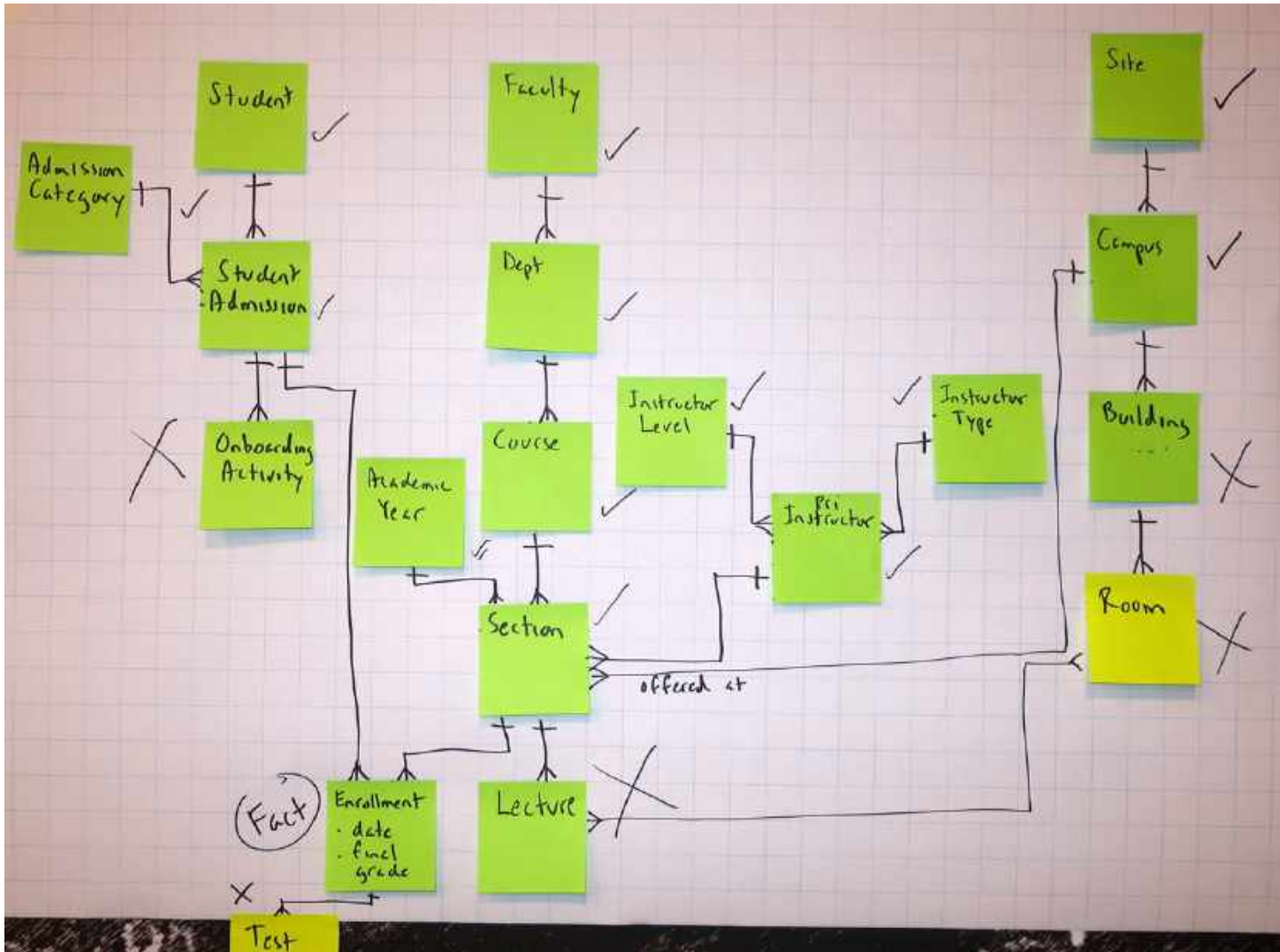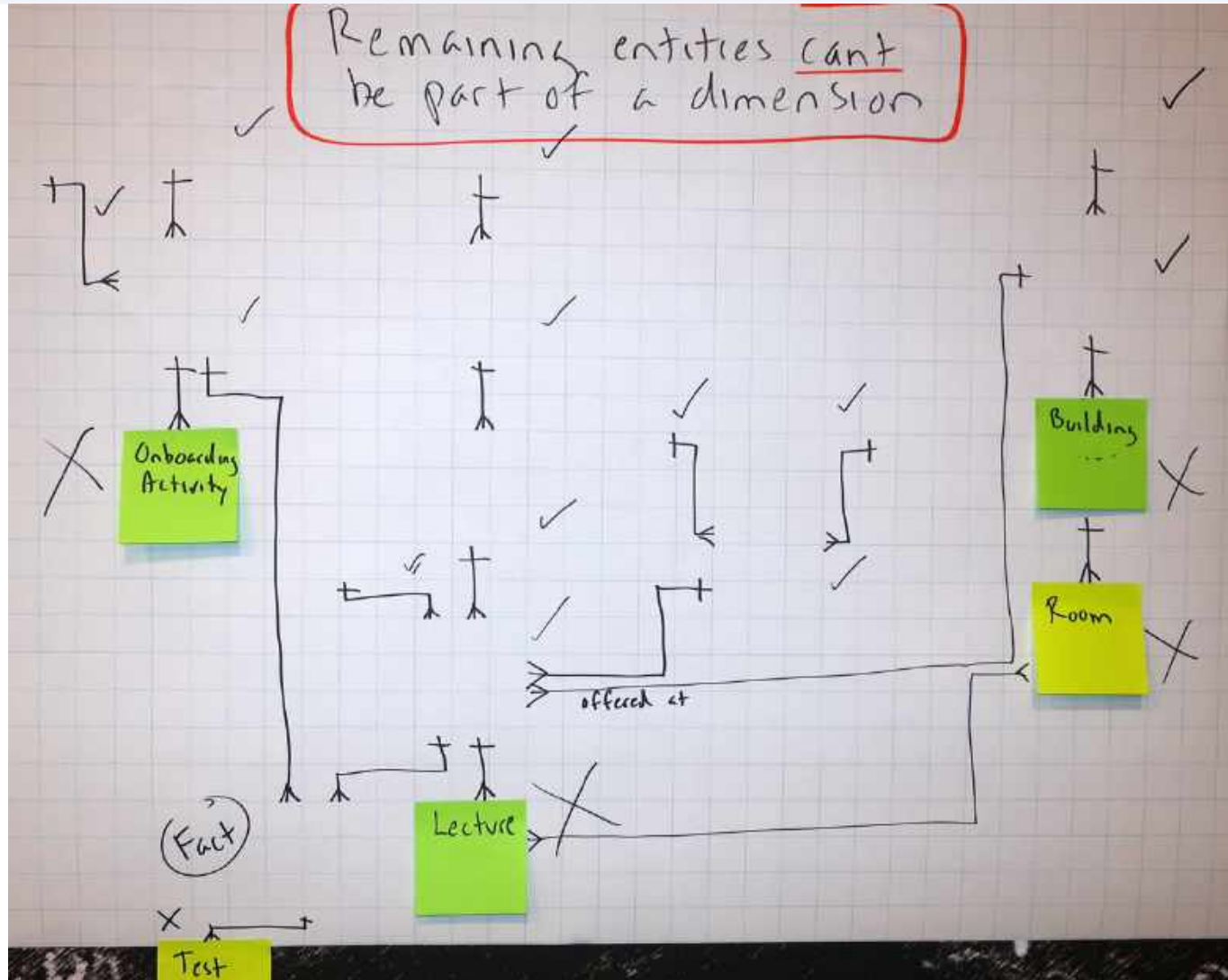| Step	Notes
1 Identify questions	What sorts of relationships among the data are of interest? E.g., want to study sales by product colour and customer, or by region and employee seniority.
2 Identify facts	What is the central thing (or things) of interest? Often a transaction or event entity with multiple parents and classifications. E.g., a Sale
3 Identify dimensions	How will facts be organised? Usually an entity related to the fact entity (a foreign key.) E.g., Employee, Customer, … May be hierarchic, e.g. Country, Region, "State", …
4 Add attributes	What additional detail is needed? Facts have "measures" and dimensions have "attributes". E.g., Sale units, total price, time of day, …
5 Add calculations	Identify calculations such as totals, average, or projection that should be pre-defined. E.g., average sale price, total sales per month,

But it's easier with an ERD

Not a dimension

Publisher
Publisher ID
Name

Dimension

Title
Title ID
Name
Author

available from

Cardholder

Cardholder ID
Name
Number
Member Since Date

Dimension

is an instance of

Dimension

Format Type
Format Type Code
Name

Copy
Title ID
Copy SID
Purchase Price Amt
Acquisition Date
Status Code
Format Type Code (fk)

is classified by

Dimension

is taken by

Loan
Loan ID
Date
Cardholder ID (fk,nn)

Dimension

takes   is part of

Fact

Loan Item
Loan ID
Title ID
Copy SID
Due Date
Return Date
Status Code

198

"Flattening" the ERD

Facts and Dimensions are "collapsed" or "flattened" portions of the E-R model.

# Some entities can't form part of facts or dimensions

200

"Flattened" ERD becomes the Dimensional Model
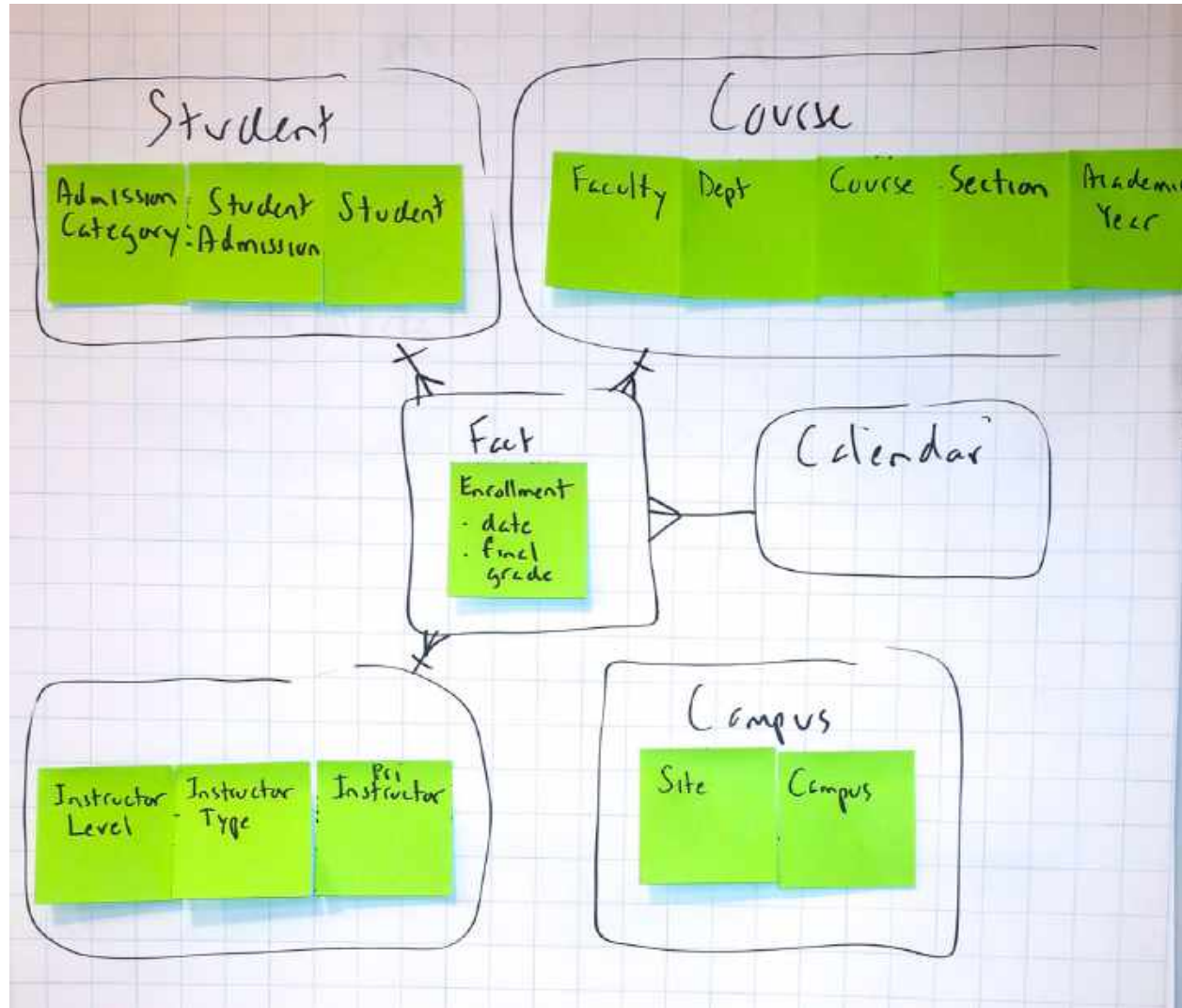
From E-R to dimensional

✓ Any parent (or grandparent or…) entities that are encountered following M:1 relationships from the fact are *possible* dimensions

✓ Any entities that are 1:M or M:M from the fact cannot be dimensions without "faking" the data

✓ Additional dimensions not in the original structure (e.g., Time Period) can be added

✓ Essentially, a basic dimensional model (no snowflakes) collapses an ER model to a two-level structure with a 1:M relationship between each dimension and the fact
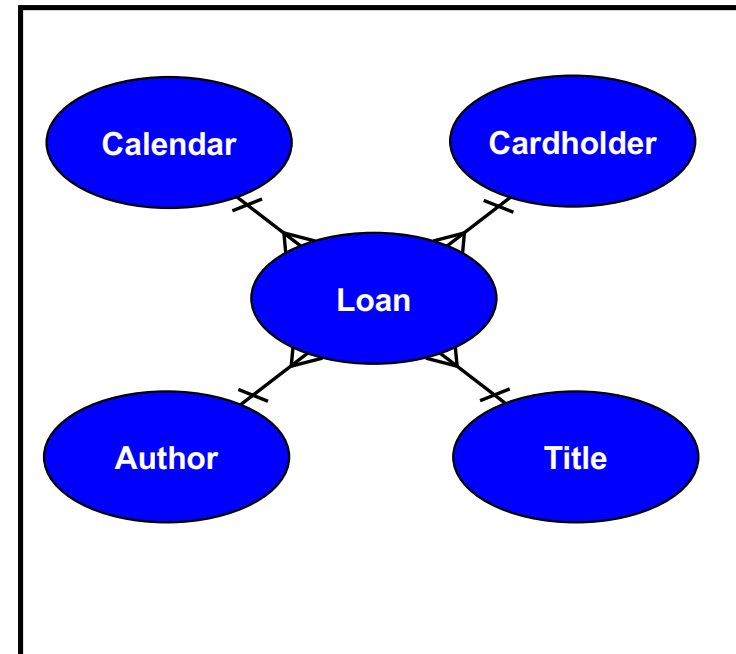
Exercise: dimensional modelling

Jim's sister-in-law June has just returned from a BI conference, and she has Jim all wound up about building a query database so he can analyse sales (purchases by customers.)

Construct a dimensional model for Jim, using the following E-R model as a starting point. At this point, don't worry about individual attributes – just which entities would collapse into which fact or dimension. A few notes:
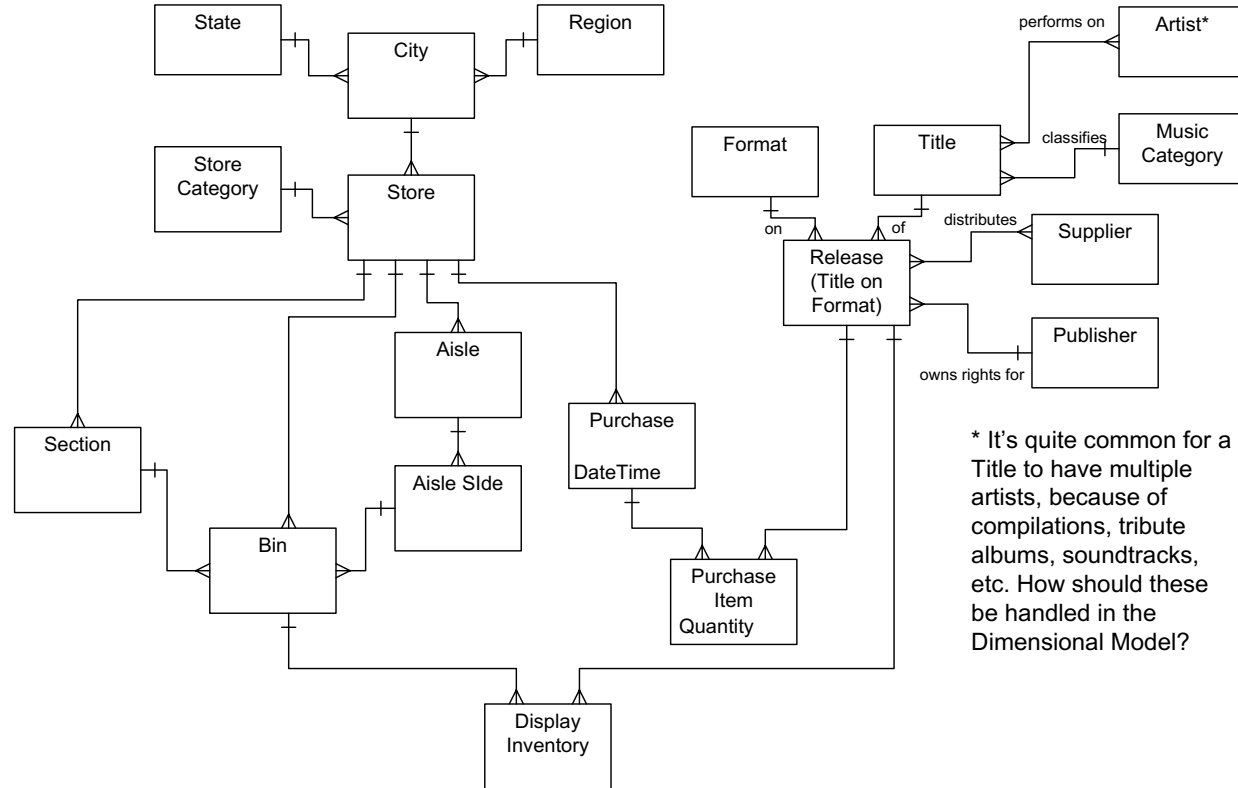
- Jim's has grown to a nationwide chain, with stores in many regions. Most regions cover one or more states, although some regions only cover part of a state (e.g., Northern California and Southern California). Each store is in a single city, though, and each city is in only one region.

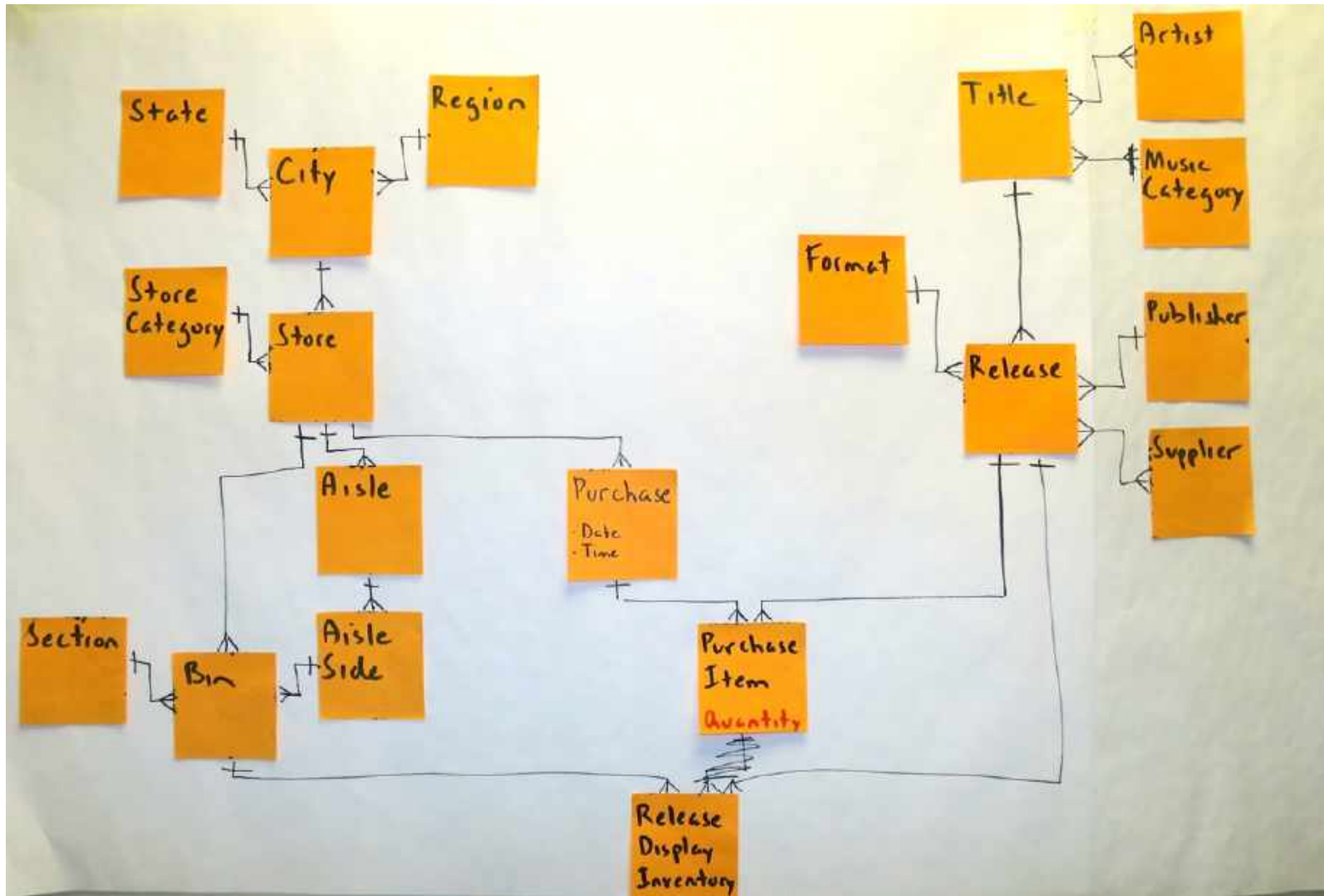- The layout of stores (Sections, Aisles, Store Categories, etc.) varies widely across the stores.

- The "Store Category" indicates if the store is a mall location, streetfront, "captive" (contained within another retail outlet,) etc. Web sales are not a factor.

Jim is especially interested in how the same Title sells depending on where in the Store it is displayed, because the same Title might end up in different Sections. He also wants to look at Sales by Store, Region, Artist, Publisher, Supplier, Category, … well, just about everything! You'll have to decide what's possible, and then be prepared to explain it to Jim!
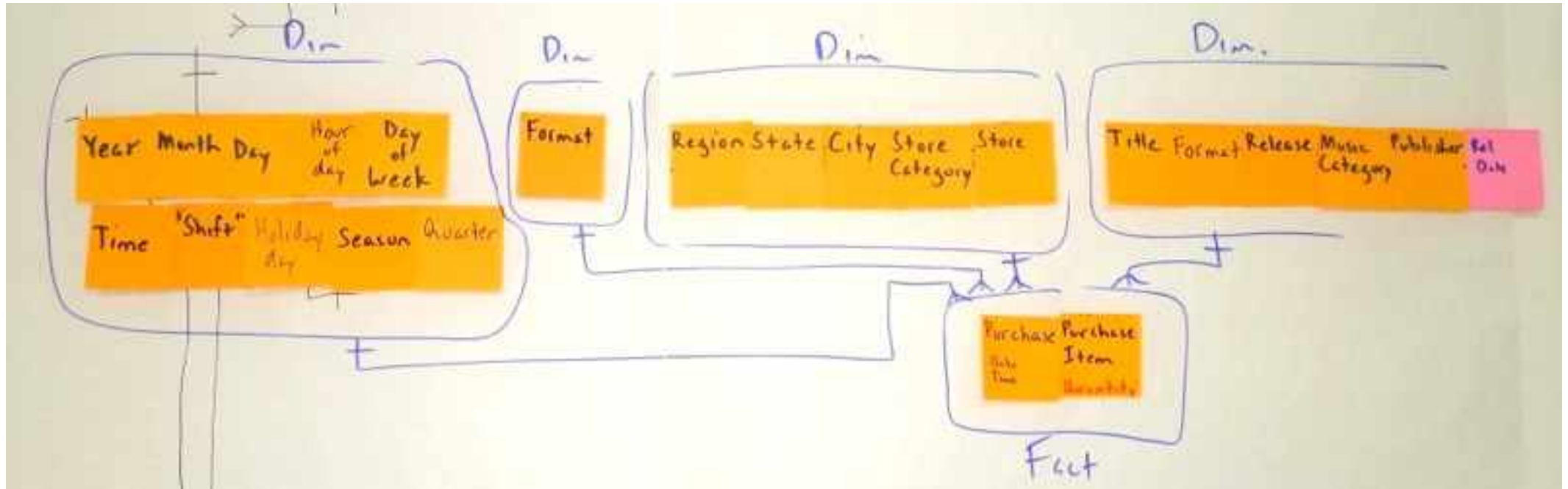
Dimensional modelling exercise

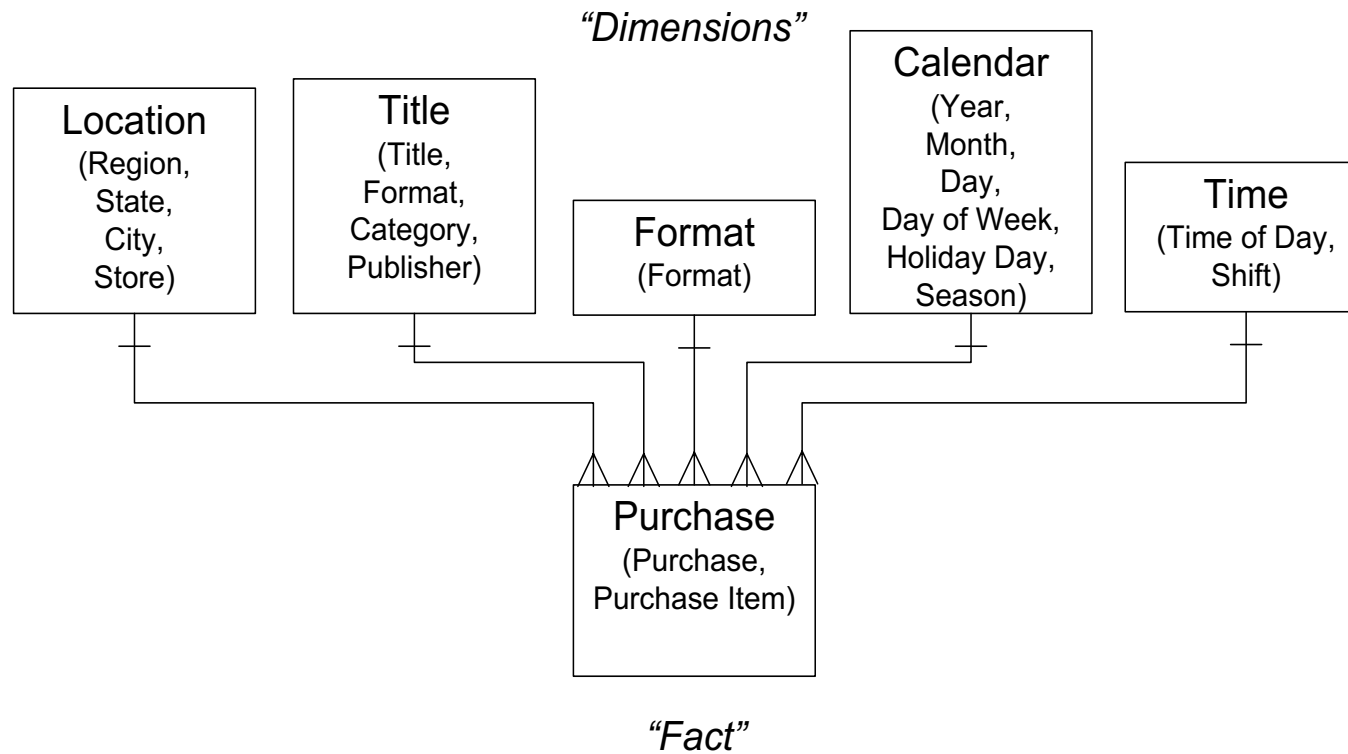* It's quite common for a Title to have multiple artists, because of compilations, tribute albums, soundtracks, etc. How should these be handled in the Dimensional Model?

205

Workshop example

206

Workshop example

Workshop example

208

Solution: dimensional model

"Dimensions"

| Location
(Region,
State,
City,
Store) | Title
(Title,
Format,
Category,
Publisher) | Format
(Format) | Calendar
(Year,
Month,
Day,
Day of Week,
Holiday Day,
Season) | Time
(Time of Day,
Shift) |

Purchase
(Purchase,
Purchase Item)

"Fact"

Other courses for analysts by Alec Sharp

Working With Business Processes – *Process Change in Agile Timeframes*	2 days

Business processes matter, because business processes are how value is delivered. Understanding how to work with business processes is now a core skill for business analysts, process and application architects, functional area managers, and even corporate executives. But too often, material on the topic either floats around in generalities and familiar case studies, or descends rapidly into technical details and incomprehensible models. This workshop is different – in a practical way, it shows how to discover and scope a business process, clarify its context, model its workflow with progressive detail, assess it, and and transition to the design of a new process by determining, verifying, and documenting its essential characteristics. Everything is backed up with real-world examples, and clear, repeatable guidelines. *Our most popular workshop!*

Business-Oriented Data Modelling – *Useful Models in Agile Timeframes*	2 days

Data modelling was often seen as a technical exercise, but is now known to be essential to other initiatives such as business process change, requirements specification, Agile development, and even big data, analytics, and data lake implementation. Why? – because it ensures a common understanding of the things – the entities or business objects – that processes, applications, and analytics deal with. This workshop introduces concept modelling from a non-technical perspective, provides tips and guidelines for the analyst, and explores entity-relationship modelling at contextual, conceptual, and logical levels using techniques that maximise client involvement.

Advanced Business Process Techniques – *Aligning Process Work with Strategic, Organisational, and Cultural Factors*	2 days

We regularly hear about the importance of "alignment" in achieving success when working with business processes, but alignment with what? This workshop provides specific, repeatable techniques to help your business process initiatives align with human factors, organisational culture, and enterprise strategy and goals. Rather than save these concerns for a single "think about people, culture, and strategy" phase, it shows how to incorporate them at every stage, from process identification, scoping, and initial assessment through to modelling, analysis, and resign. Regularly receives rave reviews.

Business-Oriented Data Modelling Masterclass – *Balancing Engagement, Agility, and Complexity*	3 days

This highly interactive workshop combines the core content from two popular data modelling offerings by Alec Sharp – "Business Oriented Data Modelling" and "Advanced Data Modelling." The first day of the workshop gets both new and experienced modellers to the same baseline on terminology, conventions, and unique, business-engaging approaches. The next two days provide intense, hands-on practice with more advanced situations, such as the enforcement of complex business rules, handling recurring patterns, satisfying regulatory requirements to model time and history, capturing complex changes and corrections, and integrating with dimensional modelling. In all cases, the underlying philosophy is that a data model is a description of a business, not of a database.

Three main themes are explored in a very practical way:
1. The foundations of data modelling – what a data model *really* is, and maximising its relevance
2. The human side of data modelling - improving communication skills and engaging the business
3. The complex side of data modelling - getting better at modelling difficult situations

Model-Driven Business Analysis Techniques – *Proven Techniques for Processes, Applications, and Data*	3 days

Simple, list-based techniques are fine as a starting point, but only with more rigorous techniques will a complete set of requirements emerge, and those requirements must then be synthesised into a cohesive view of the desired to-be state. This three-day workshop shows how to accomplish that with an integrated, model-driven framework comprising process workflow models, a unique form of use cases, service specifications, and business-friendly data models. This distinctive approach has succeeded on projects of all types because it is "do-able" by analysts, relevant to business subject matter experts, and useful to developers. It distills the material from Clariteq's three, two-day workshops on process, data, and use cases & services.
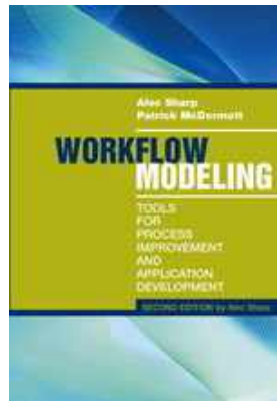
*** *Note: two-day in-person workshops are delivered virtually as three half-day sessions via Zoom.*
   Three-day in-person workshops are delivered virtually as five half-day sessions via Zoom.

Thank you – stay in touch!

Alec Sharp
Clariteq Systems Consulting Ltd.
West Vancouver, BC, Canada

- asharp@clariteq.com

- t: @alecsharp

- ig: @alecsharp01

- www.clariteq.com

- http://amzn.to/dHun1o

And most of all, if you have questions or comments…
don't be shy – send me a note!