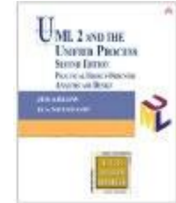
 

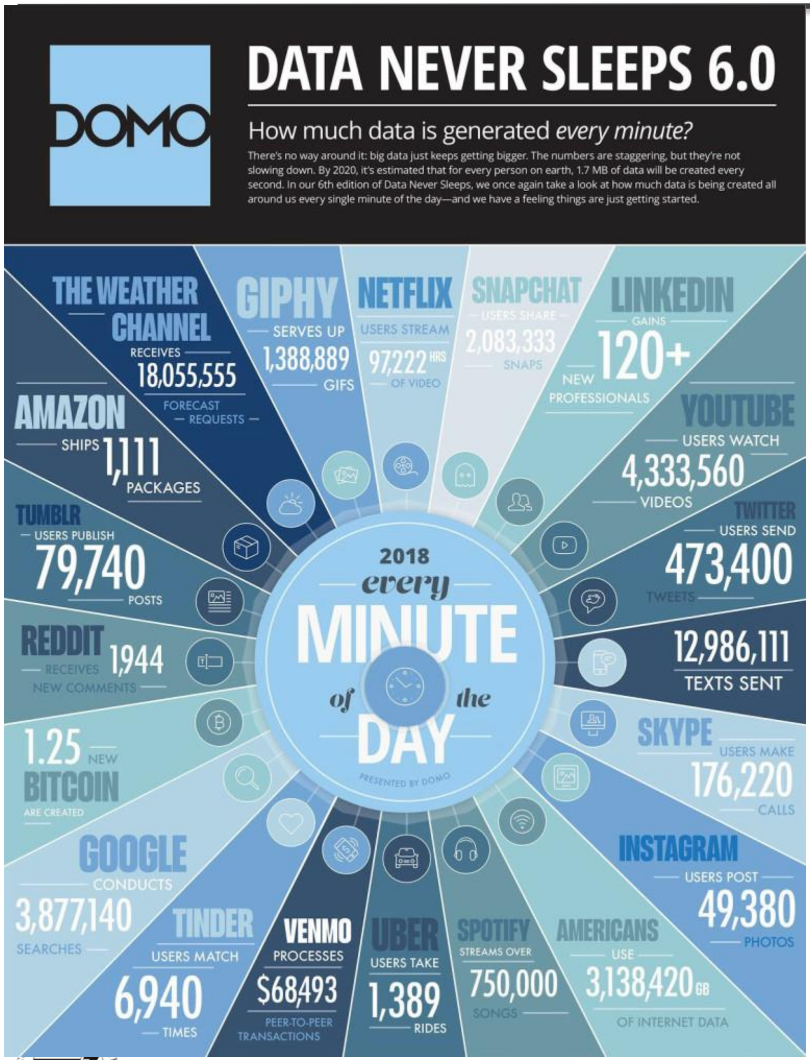
Lecture 5

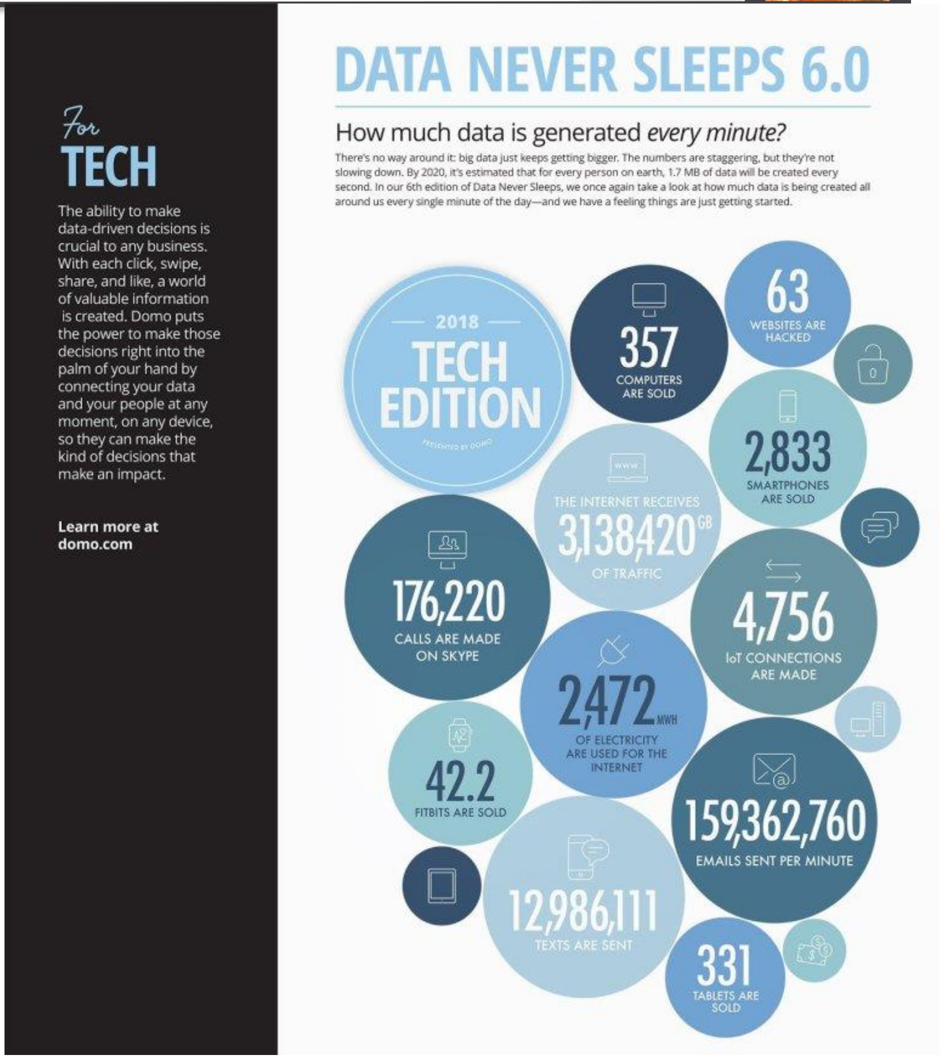
DATA MODELLING AND MANAGEMENT

PB007 Software Engineering I
Faculty of Informatics, Masaryk University
Fall 2019

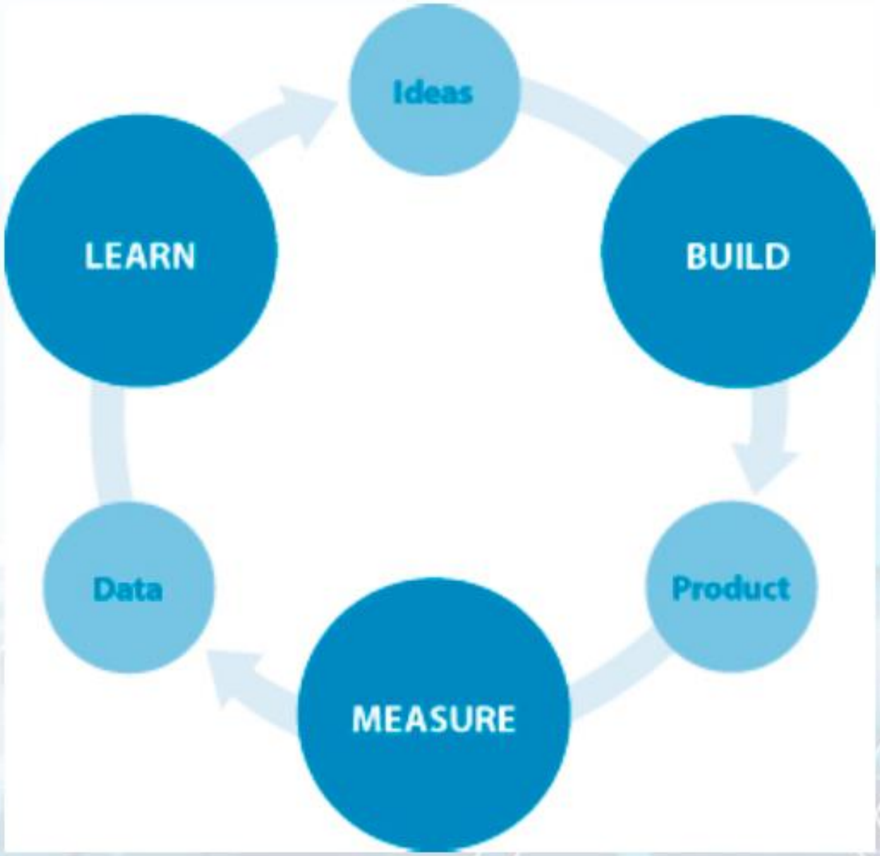
Motivation

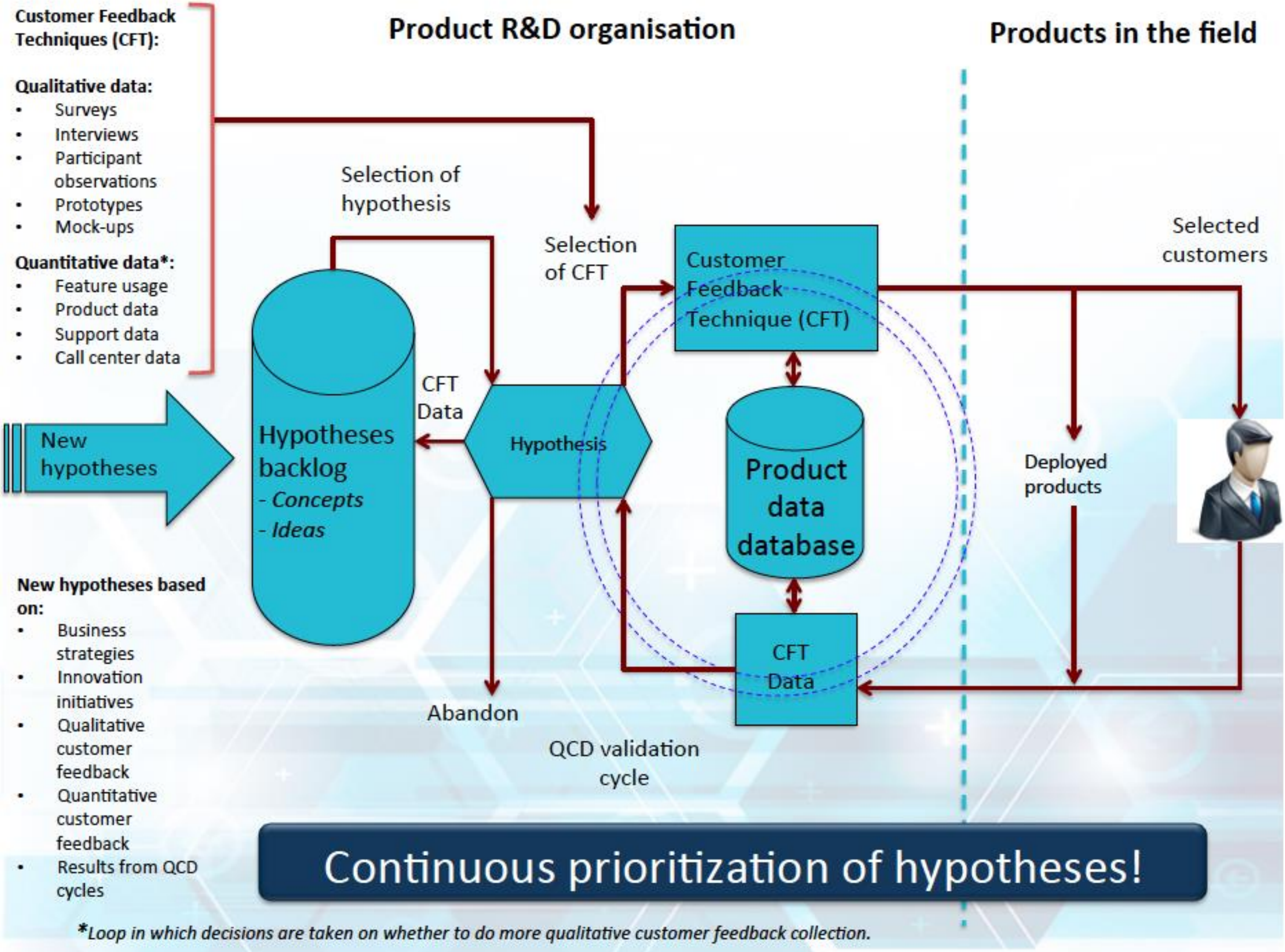

SOFTWARE ENGINEERING
9


DATA NEVER SLEEPS 6.0
How much data is generated every minute?
There's no way around it: big data just keeps getting bigger. The numbers are staggering, but they're not slowing down. By 2020, it's estimated that for every person on earth, 1.7 MB of data will be created every second. In our 6th edition of Data Never Sleeps, we once again take a look at how much data is being created all around us every single minute of the day—and we have a feeling things are just getting started.
THE WEATHER CHANNEL RECEIVES 18,055,555 FORECAST REQUESTS
GIPHY SERVES UP 1,388,889 GIFS
NETFLIX USERS STREAM 97,222 HRS OF VIDEO
SNAPCHAT USERS SHARE 2,083,333 SNAPS
LINKEDIN GAINS 120+ NEW PROFESSIONALS
AMAZON SHIPS 1,111 PACKAGES
YOUTUBE USERS WATCH 4,333,560 VIDEOS
TUMBLR USERS PUBLISH 79,740 POSTS
TWITTER USERS SEND 473,400 TWEETS
REDDIT RECEIVES 1,944 NEW COMMENTS
12,986,111 TEXTS SENT
1.25 NEW BITCOIN ARE CREATED
SKYPE USERS MAKE 176,220 CALLS
GOOGLE CONDUCTS 3,877,140 SEARCHES
INSTAGRAM USERS POST 49,380 PHOTOS
TINDER USERS MATCH 6,940 TIMES
VENMO PROCESSES $68,493 PEER-TO-PEER TRANSACTIONS
UBER USERS TAKE 1,389 RIDES
SPOTIFY STREAMS OVER 750,000 SONGS
AMERICANS USE 3,138,420 GB OF INTERNET DATA
2018 every MINUTE of the DAY

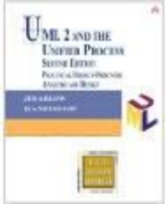

For TECH
The ability to make data-driven decisions is crucial to any business. With each click, swipe, share, and like, a world of valuable information is created. Domo puts the power to make those decisions right into the palm of your hand by connecting your data and your people at any moment, on any device, so they can make the kind of decisions that make an impact.
Learn more at domo.com
DATA NEVER SLEEPS 6.0
How much data is generated every minute?
There's no way around it: big data just keeps getting bigger. The numbers are staggering, but they're not slowing down. By 2020, it's estimated that for every person on earth, 1.7 MB of data will be created every second. In our 6th edition of Data Never Sleeps, we once again take a look at how much data is being created all around us every single minute of the day—and we have a feeling things are just getting started.
2018 TECH EDITION
357 COMPUTERS ARE SOLD
63 WEBSITES ARE HACKED
2,833 SMARTPHONES ARE SOLD
THE INTERNET RECEIVES 3,138,420 GB OF TRAFFIC
176,220 CALLS ARE MADE ON SKYPE
4,756 IoT CONNECTIONS ARE MADE
2,472 MWH OF ELECTRICITY ARE USED FOR THE INTERNET
42.2 FITBITS ARE SOLD
159,362,760 EMAILS SENT PER MINUTE
12,986,111 TEXTS ARE SENT
331 TABLETS ARE SOLD

The cycle of innovation

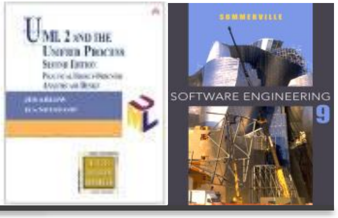



Customer Feedback Techniques (CFT):
Qualitative data:
• Surveys
• Interviews
• Participant observations
• Prototypes
• Mock-ups
Quantitative data*:
• Feature usage
• Product data
• Support data
• Call center data
Product R&D organisation
Products in the field
Selection of hypothesis
Selection of CFT
Customer Feedback Technique (CFT)
Selected customers
New hypotheses
Hypotheses backlog
- Concepts
- Ideas
CFT Data
Hypothesis
Product data database
Deployed products
New hypotheses based on:
• Business strategies
• Innovation initiatives
• Qualitative customer feedback
• Quantitative customer feedback
• Results from QCD cycles
CFT Data
Abandon
QCD validation cycle
Continuous prioritization of hypotheses!
*Loop in which decisions are taken on whether to do more qualitative customer feedback collection.

Outline



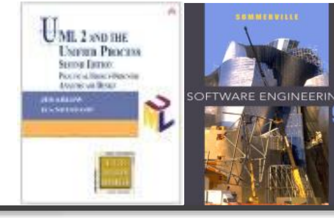
- ✧ Data management
- ✧ Data modelling
  - Entity relationship diagram (ERD)
- ✧ Relational database design
  - Normalization
- ✧ Other database concepts



Data management

Lecture 5/Part 1

Data



✧ **Information** converted into binary digital form

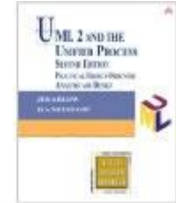
- Information that has been translated into a form that is efficient for movement, processing

✧ It can be created, processed, saved, and stored digitally

- This allows data to be transferred from one computer to another

✧ Digital information (i.e. data) in contrast to analog information does not deteriorate over time or lose quality after being used multiple times

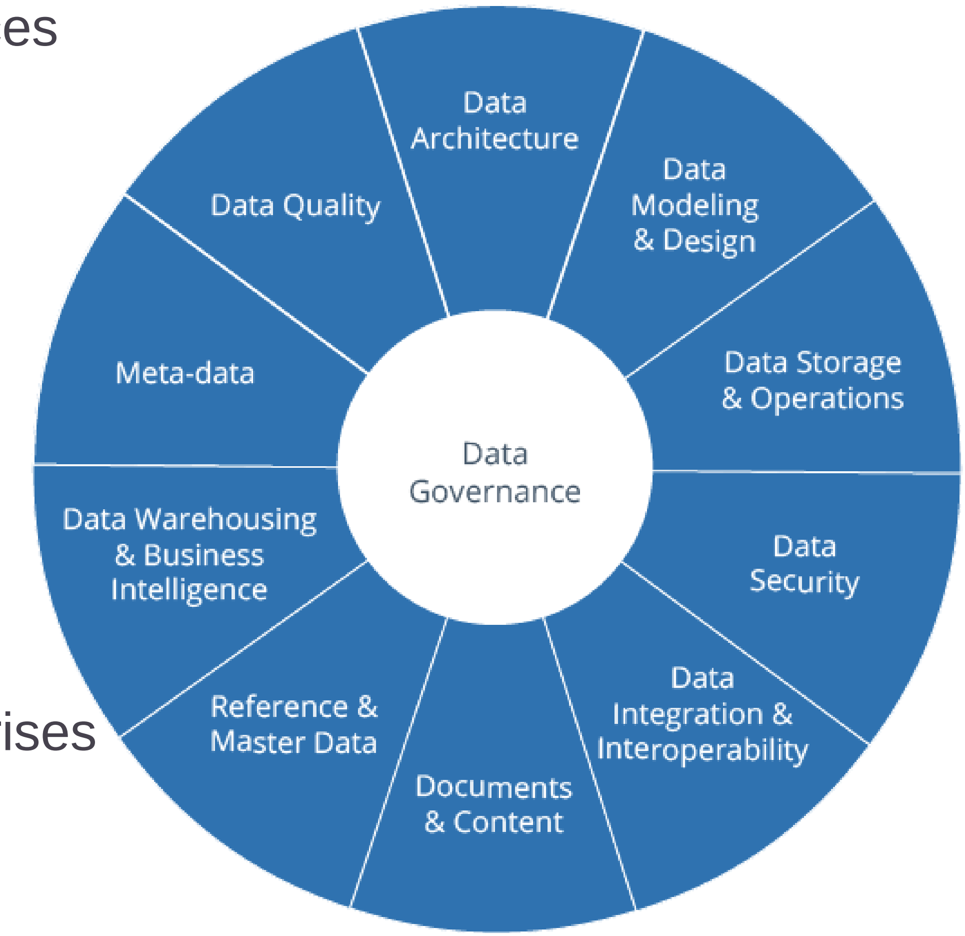
Data management



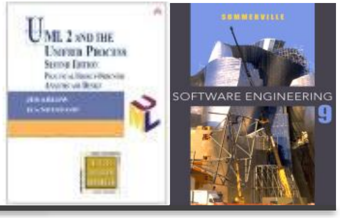
- Administrative process that includes **acquiring**, **validating**, **storing**, **protecting**, and **processing** required data to ensure the accessibility, reliability, and timeliness of the data for its users.
- Encompasses the **entire lifecycle** of a data asset, from the very initial creation of the data to the final retirement of the data.
- Some companies are good at collecting data, but they are not managing it well enough to **turn raw data into value**.

Data governance

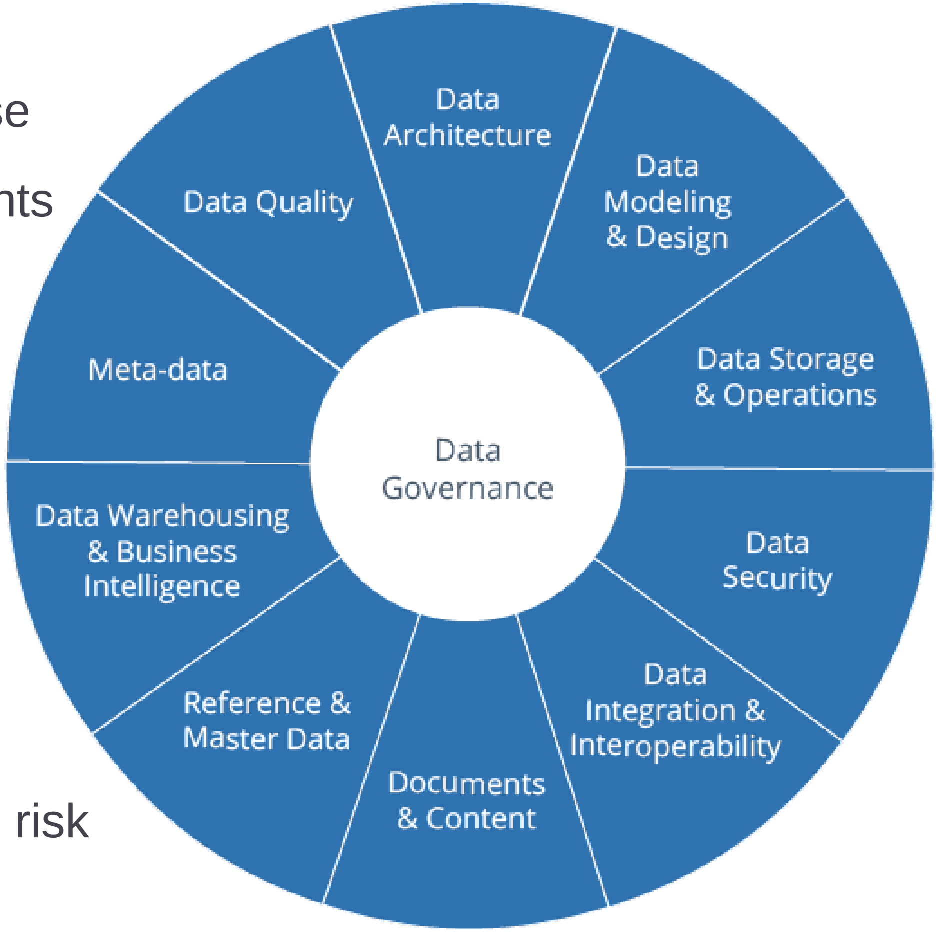
- A set of principles and practices that ensure high quality through the complete lifecycle of the data
  = a strategy
- Includes the people, processes and technologies needed to manage and protect the company's data assets
- Most relevant in large enterprises



Data governance – key goals



- ✧ Minimize risks
- ✧ Establish internal rules for data use
- ✧ Implement compliance requirements
- ✧ Improve internal and external communication
- ✧ Increase the value of data
- ✧ Facilitate the administration of the above
- ✧ Reduce costs
- ✧ Help to ensure the continued existence of the company through risk management and optimization



Data lifecycle

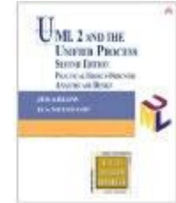
- Lifecycle may be a misleading term, since most lifecycles lead to reproduction or recycling, and data doesn't. But at least the data lifecycle has some distinct phases during which it needs to be managed.

- **Data Capture / Create**
- **Data Maintain / Store**
- **Data Use**
- **Data Publish / Share**
- **Data Archive**
- **Data Purg / Destroy**

Data modelling

Lecture 5/Part 2

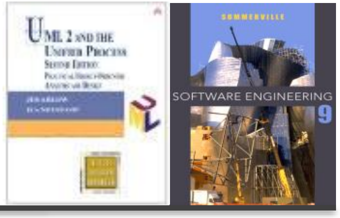
Data modeling



- ✧ Defines static data structure, relationships and attributes
- ✧ Complementary to the behavior model in structured analysis; models information not covered by DFDs
- ✧ More stable and essential information comparing to DFD

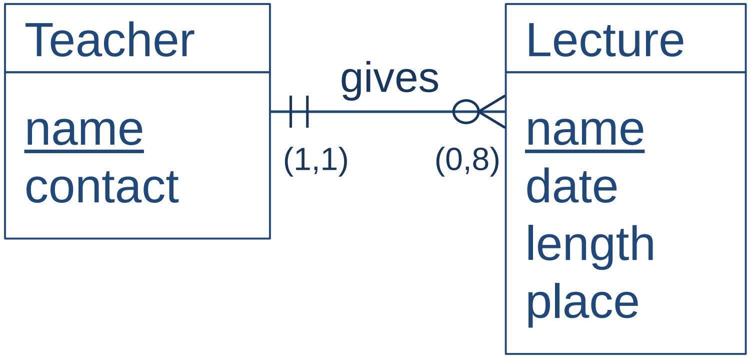
- ✧ **Entity-Relationship modeling**
  - Identify system entities – both abstract (lecture) and concrete (student)
  - For each entity examine – the purpose of the entity, its constituents (attributes) and relationships among entities
  - Check model consistency and include data details

Entity Relationship Diagram (ERD)

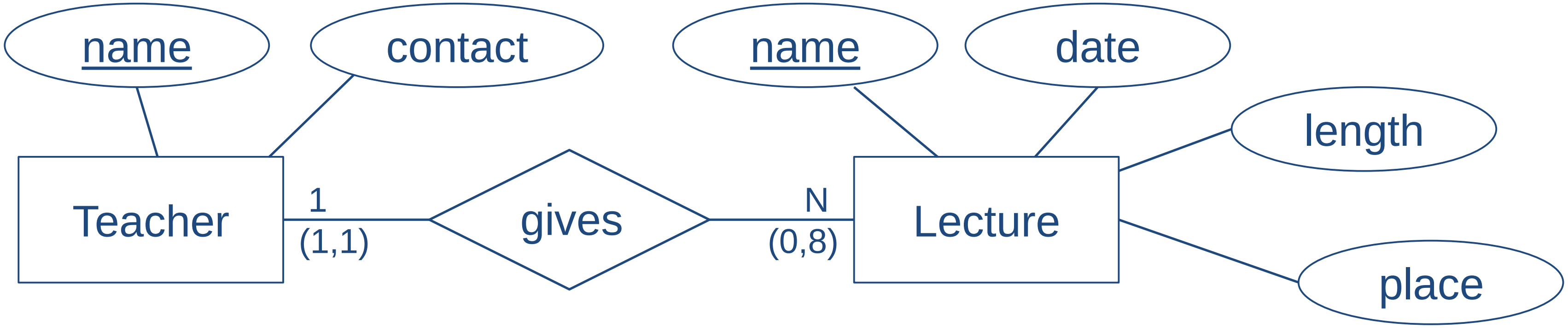


- **Entities** and their types
- **Relationships** and their types
- **Attributes** and their domains

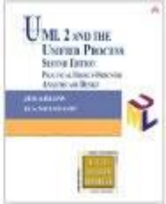
Crow's Foot notation
(implementation level descript.)



Chen's notation
(concept level description)



Entities and Entity types



✧ An **Entity** is anything about which we want to store data

- Identifiable – entities can be distinguished by their identity
- Needed – has significant role in the designed system
- Described by attributes shared by all entities of the same type

✧ An **Entity set** is a set of entities of the same **Entity type**.

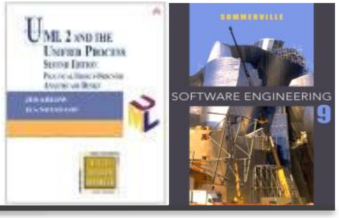
Entity	Entity type
You	Student
Your neighbor	Student
Me	Teacher
This PB007 lecture	Lecture

Student

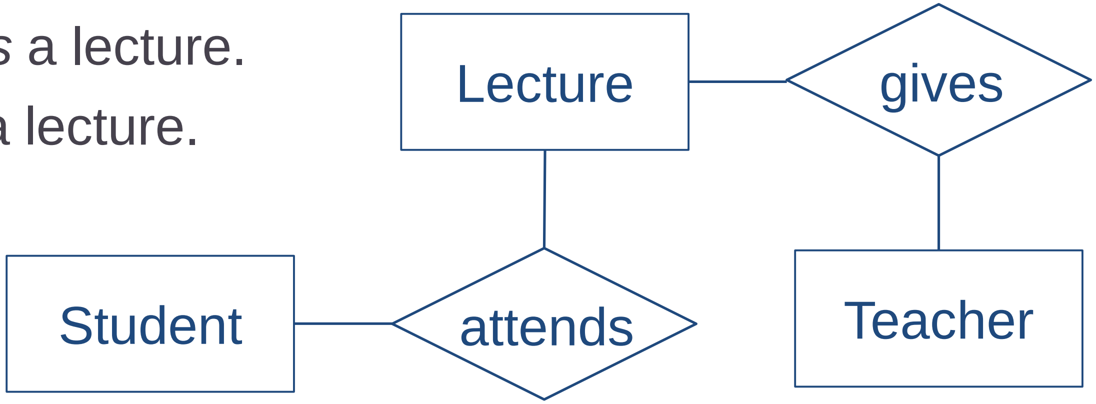
Teacher

Lecture

Relationships and Relationship types

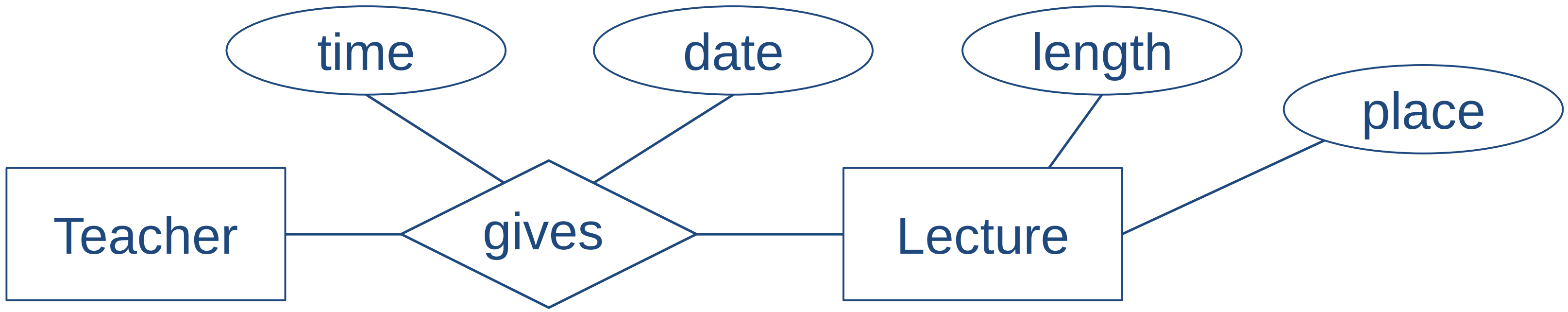


- ✧ Entities take part in **Relationships** (among possibly more than two entities), that can often be identified from verbs or verb phrases.
  - You are *attending* this PB007 lecture.
  - I am *giving* this PB007 lecture.
- ✧ A **Relationship set** is a set of relationships of the same **Relationship type**.
  - A student *attends* a lecture.
  - A teacher *gives* a lecture.



Attributes and Attribute domains

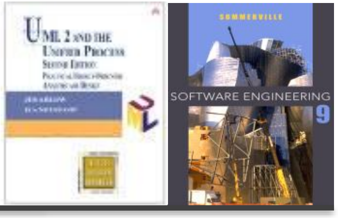
- An **Attribute** is a fact, aspect, property, or detail about either an entity type or a relationship type.
  - E.g. a lecture might have attributes: time, date, length, place.
- An **Attribute type** is a type domain of the attribute. If the domain is complex (domain of an attribute *address*), the attribute may be an entity type instead.



Attributes or entities?

- To decide whether a concept be modeled as an attribute or an entity type:
  - Do we wish to store any information about this concept (other than an identifying name)?
  - Is it single-valued?
  - E.g. *objectives* of a *course* – are they more than one? If just one, how complex information do we want to store about it?
- General guidelines:
  - Entities can have attributes but attributes have no smaller parts.
  - Entities can have relationships between them, but an attribute belongs to a single entity.

Relationship-type degree



Every manager leads exactly one department.
Every department is led by exactly one manager.

Every edition plan contains one or more book titles.
Every book title is part of exactly one edition plan.

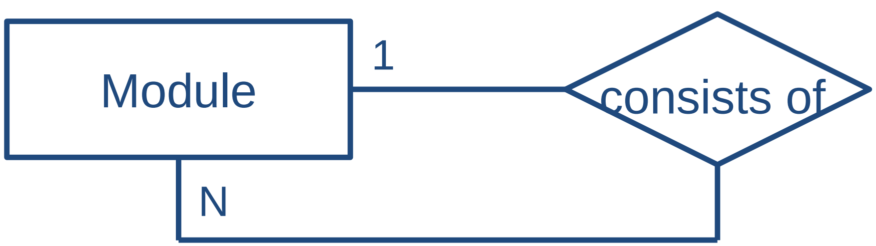
Every producer produces one or more products.
Every product is produced by one or more producers.

Relationship-type degree

Mandatory relationship

Optional relationship

Recursive relationship



Cardinality ratio

✧ **Cardinality ratio** of a relationship type describes the number of entities that can participate in the relationship.

✧ One to one 1:1

- Each lecturer has a unique office.

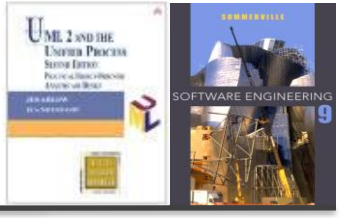
✧ One to many 1:N

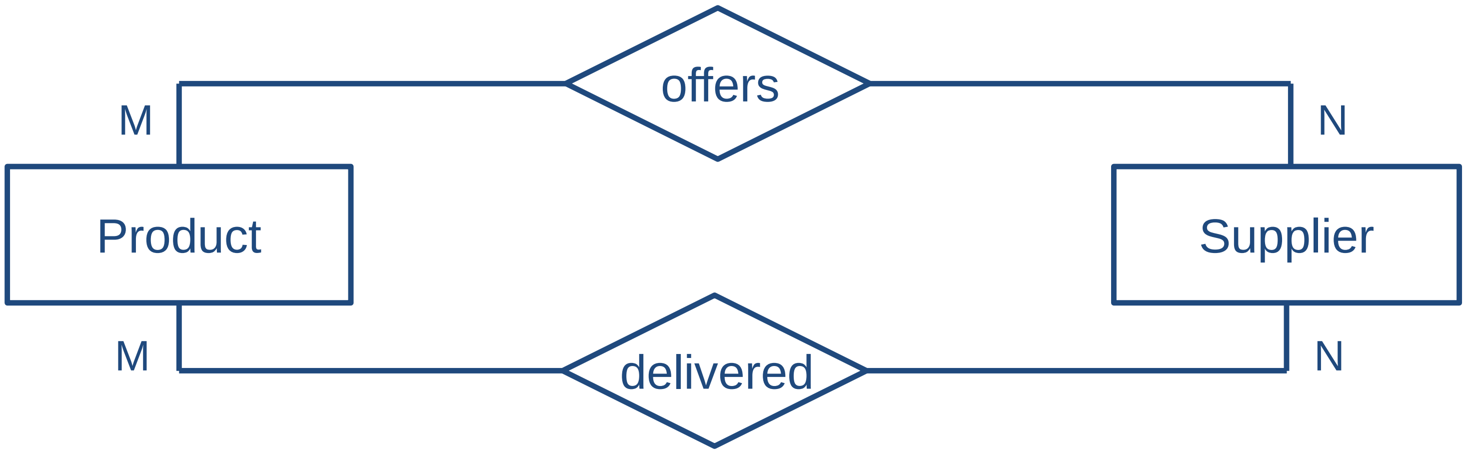
- A lecturer may tutor many students, but each student has just one tutor.

✧ Many to many M:N

- Each student takes several modules, and each module is taken by several students.

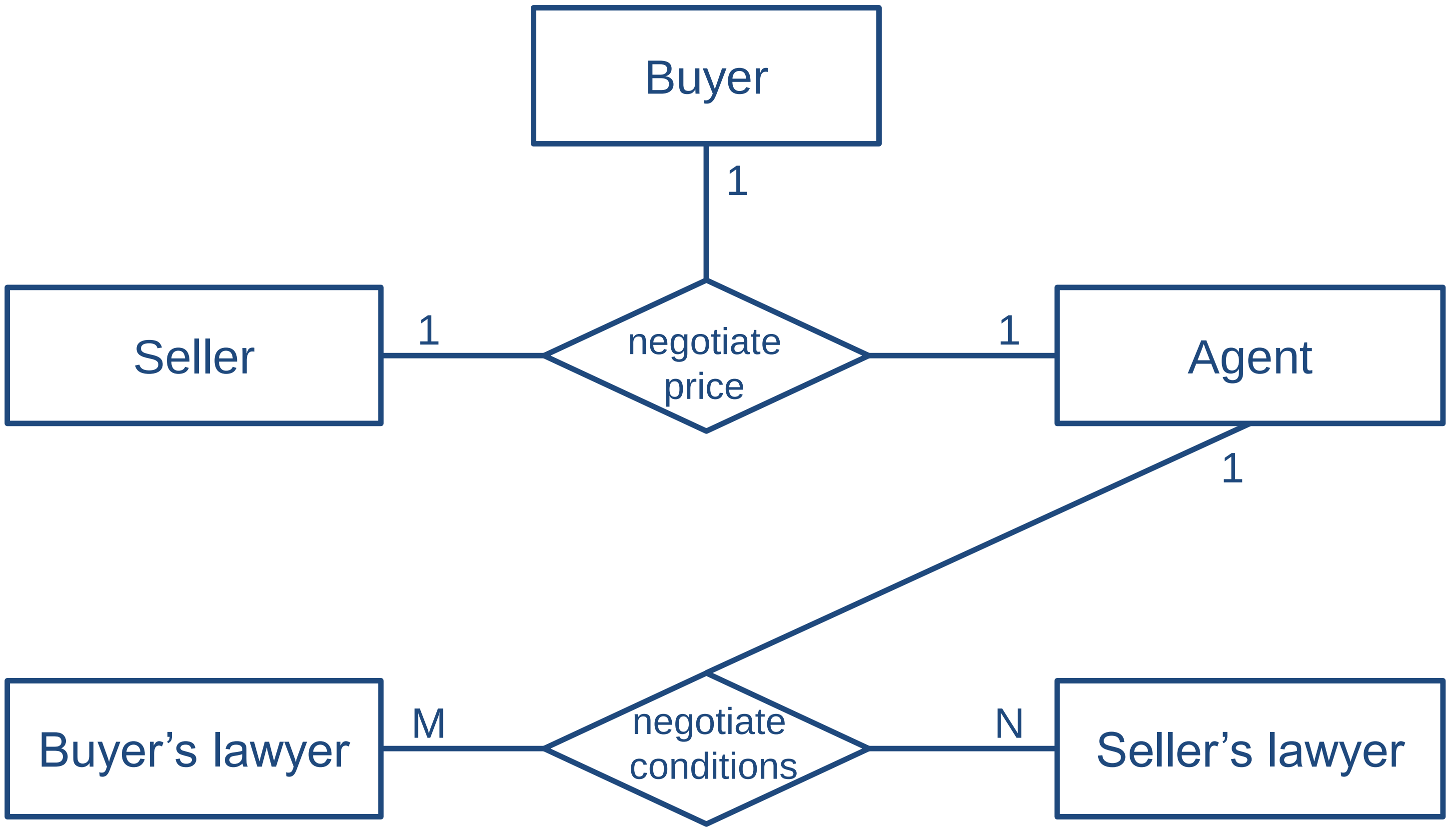
More relationships between two entities



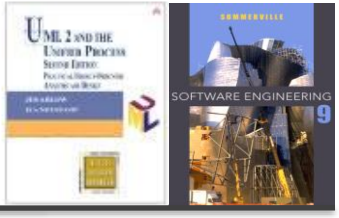


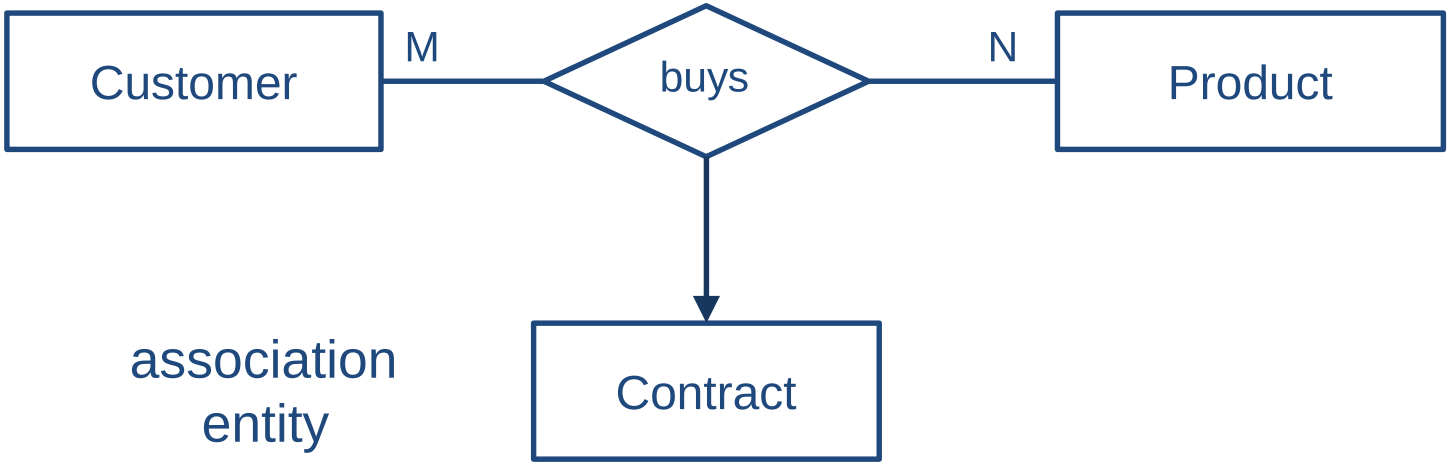
- Relationship *offers* has attributes:
  - *payment conditions, due date*.
- Relationship *delivered* has attributes:
  - *delivery note details*.

Relationships among more than two entities



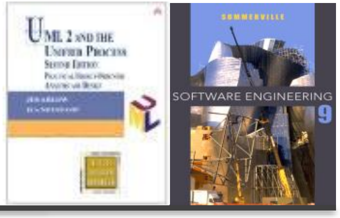
Association entity

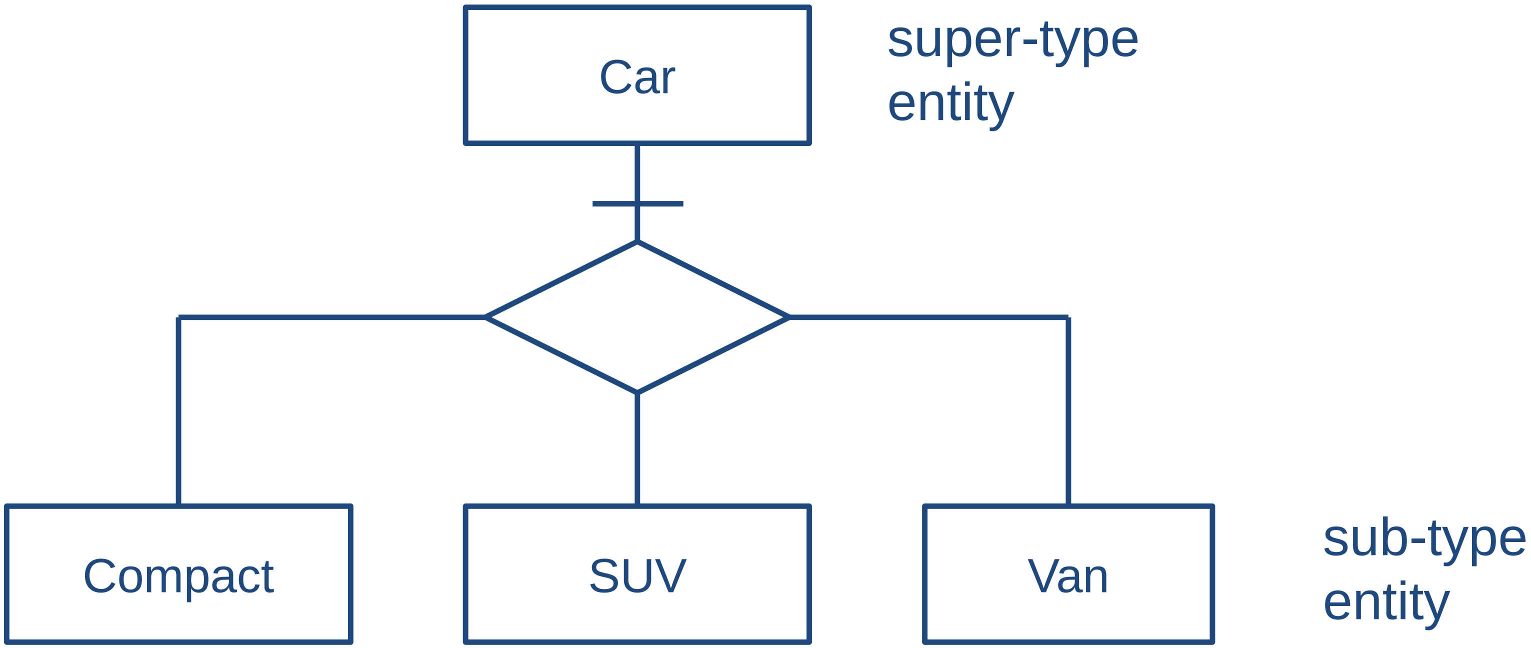




✧ The *Contract* exists just as a result of the relationship between the *Customer* and *Product* entity.

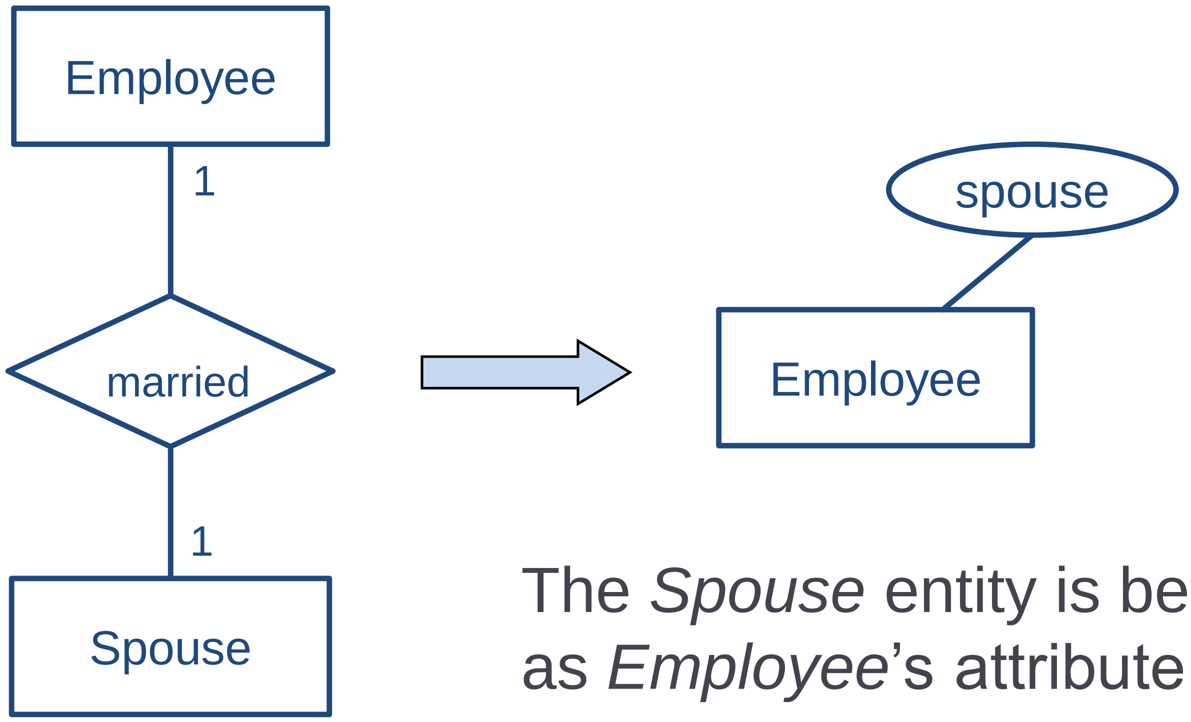
Super-type and sub-type entities





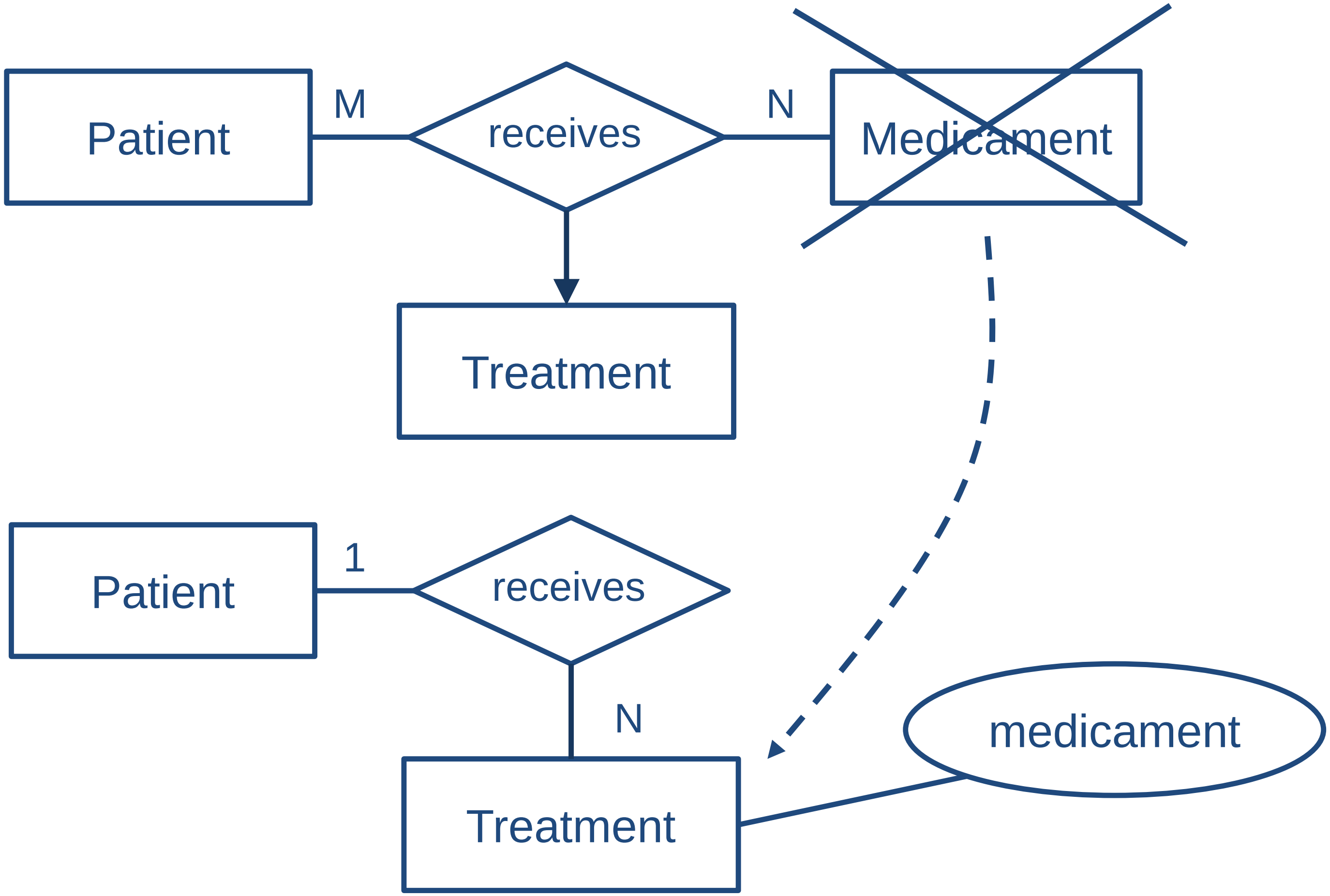
✧ Extended ERDs model also inheritance, i.e. the relationship of specialization–generalization

Removal of unneeded (redundant) entities

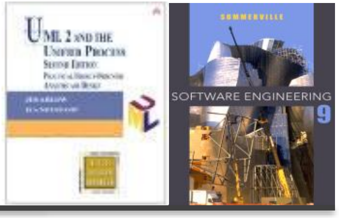


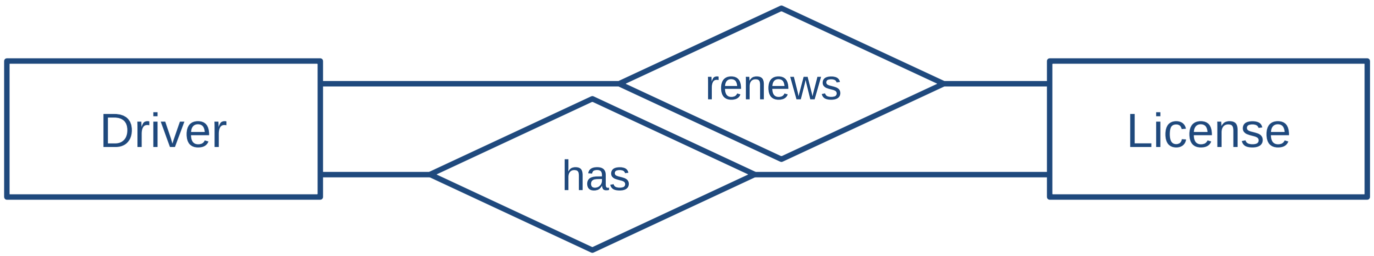
The *Spouse* entity is better suited as *Employee*'s attribute.

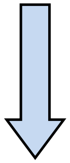
Removal of unneeded (redundant) entities



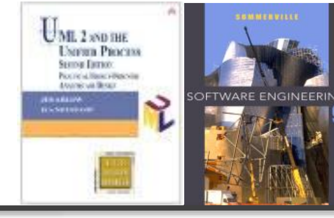
Removal of unneeded relationships







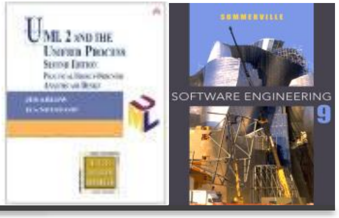
The duty to *renew* the license can be derived from the entities

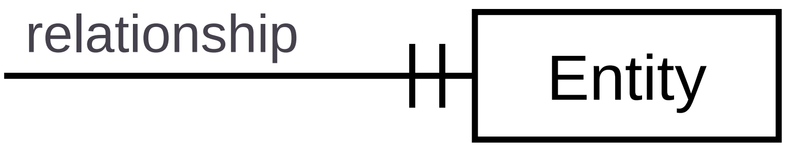


Relational Database Design

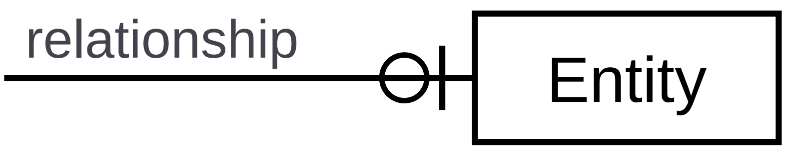
Lecture 5/Part 3

Crow's Foot notation

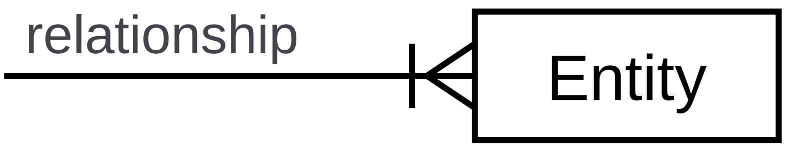




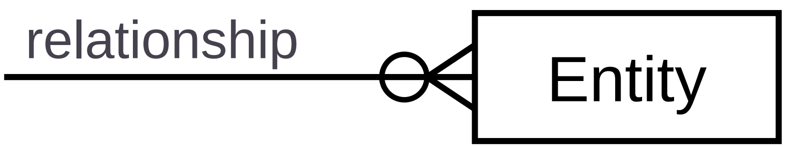
Exactly one occurrence



None or one occurrence

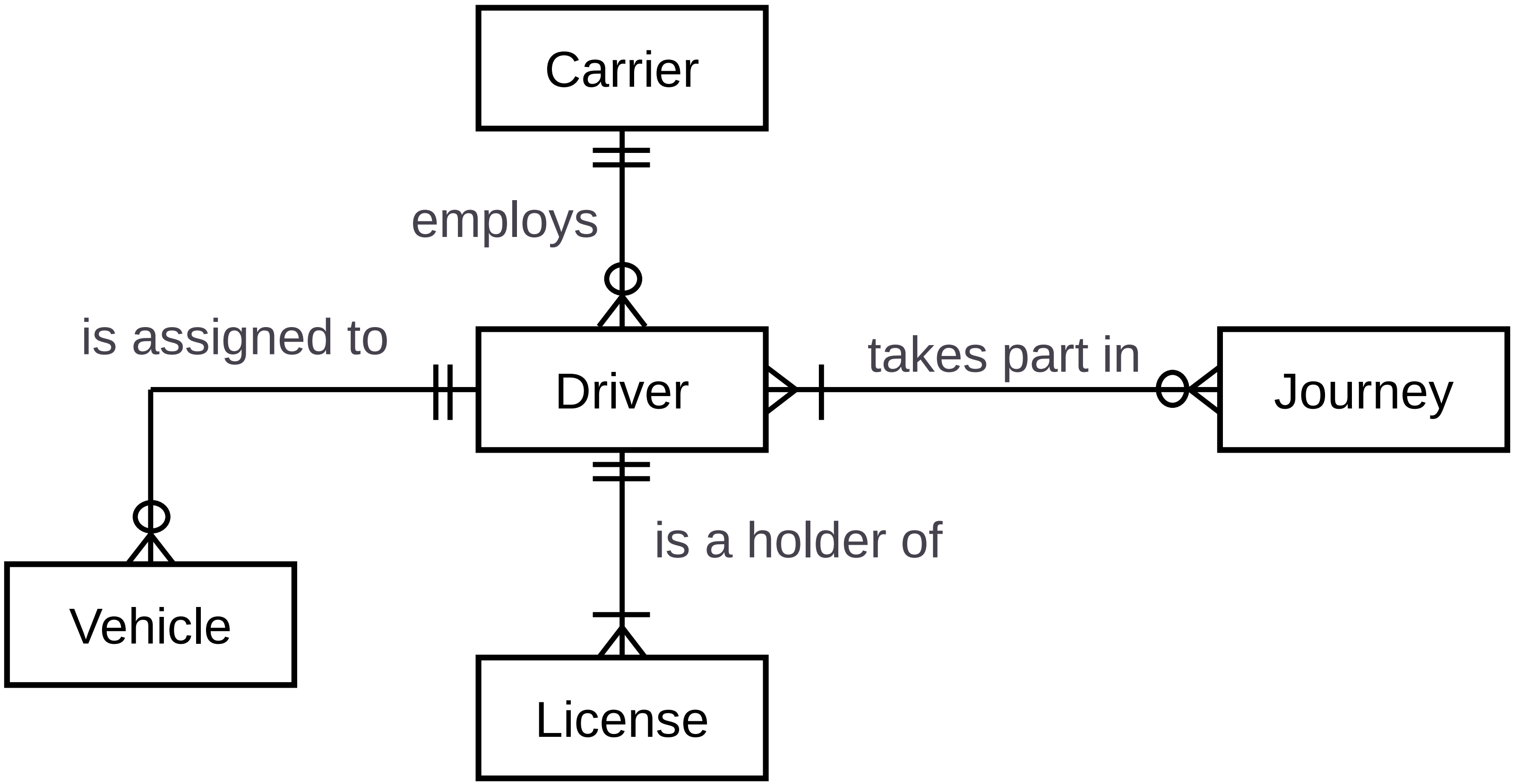


One or more occurrence

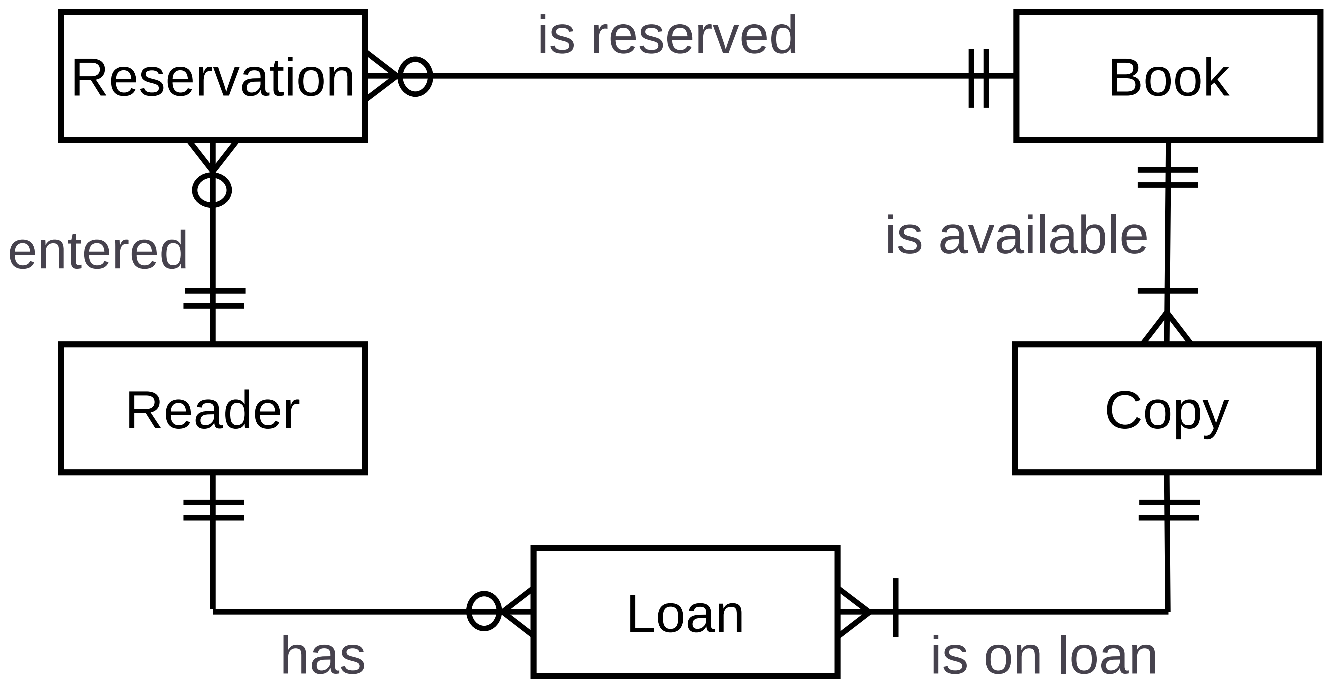


None or more occurrences

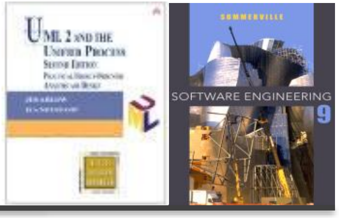
ERD example – Transport

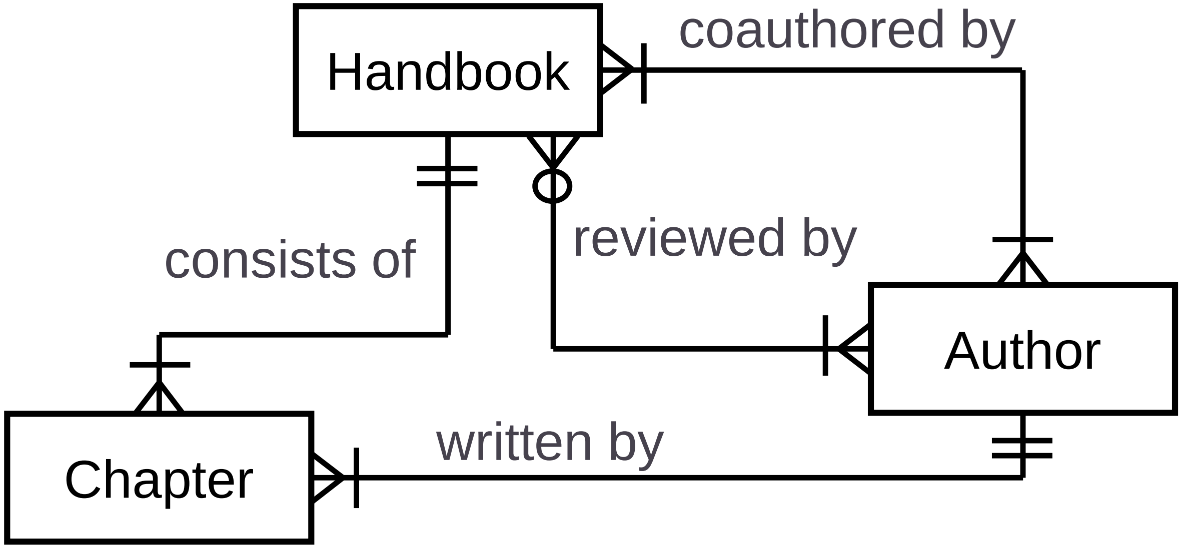


ERD example – Library

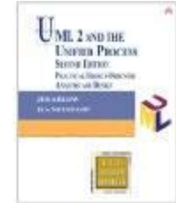


ERD example – Book editing





Relational database design based on ERDs

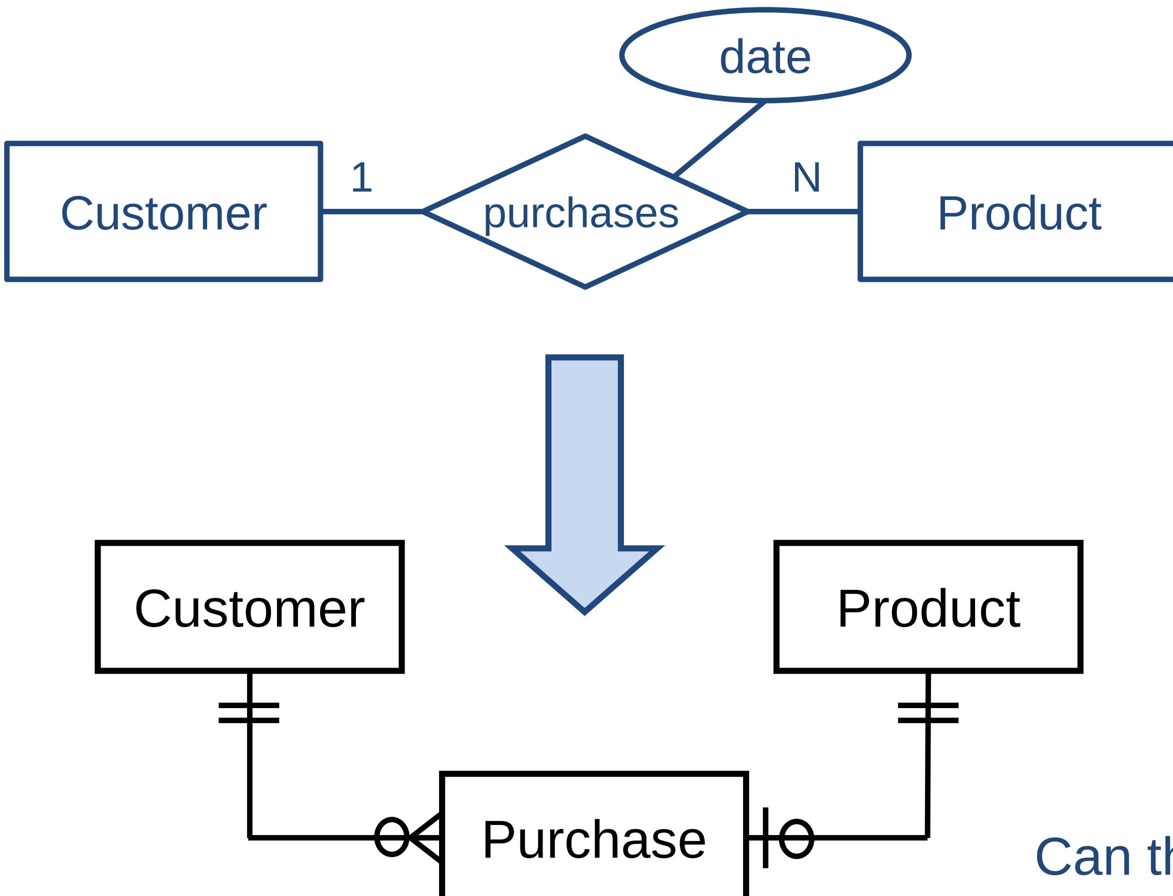


✧ Entity-relationship modeling is a first step towards database design.

Database design process:

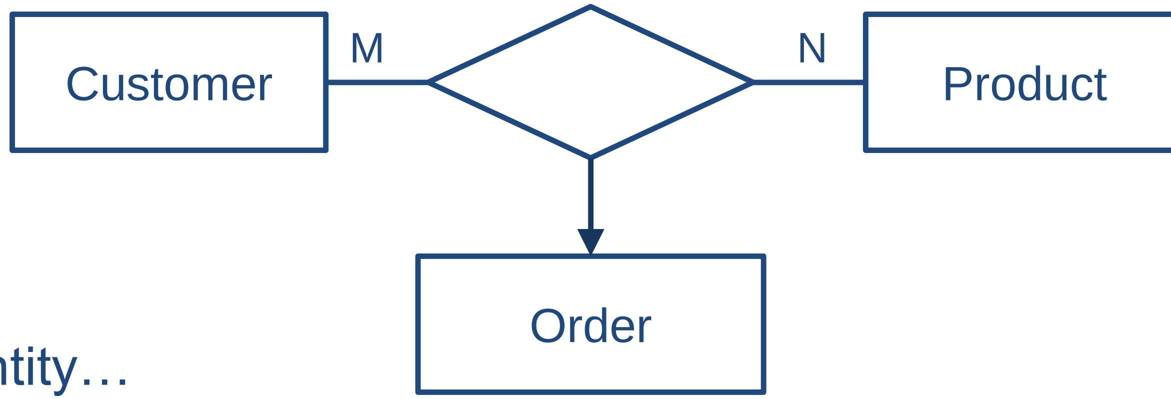
1. **Determine the purpose of the database.**
2. **Find and organize the information required** - Create ERD model of the system. Each entity type becomes a table, attribute becomes a column, entity becomes a row in the table. Handle relationships with attributes, association entities and M:N relationships.

Relationships to entities



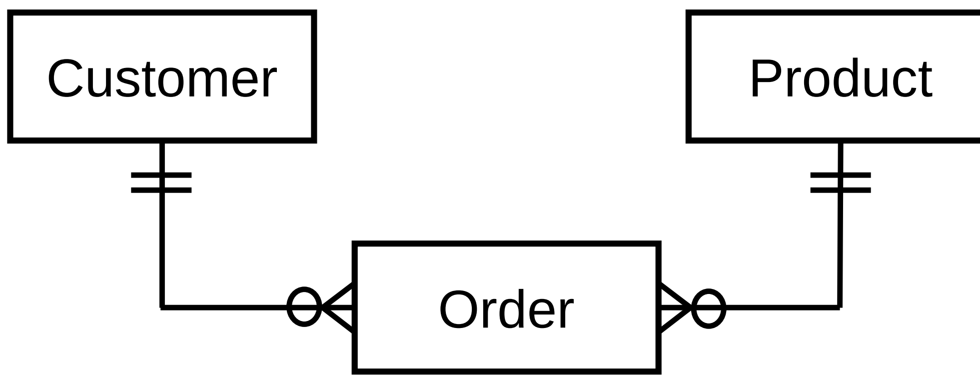
Can the purchase entity be omitted?

Association entities

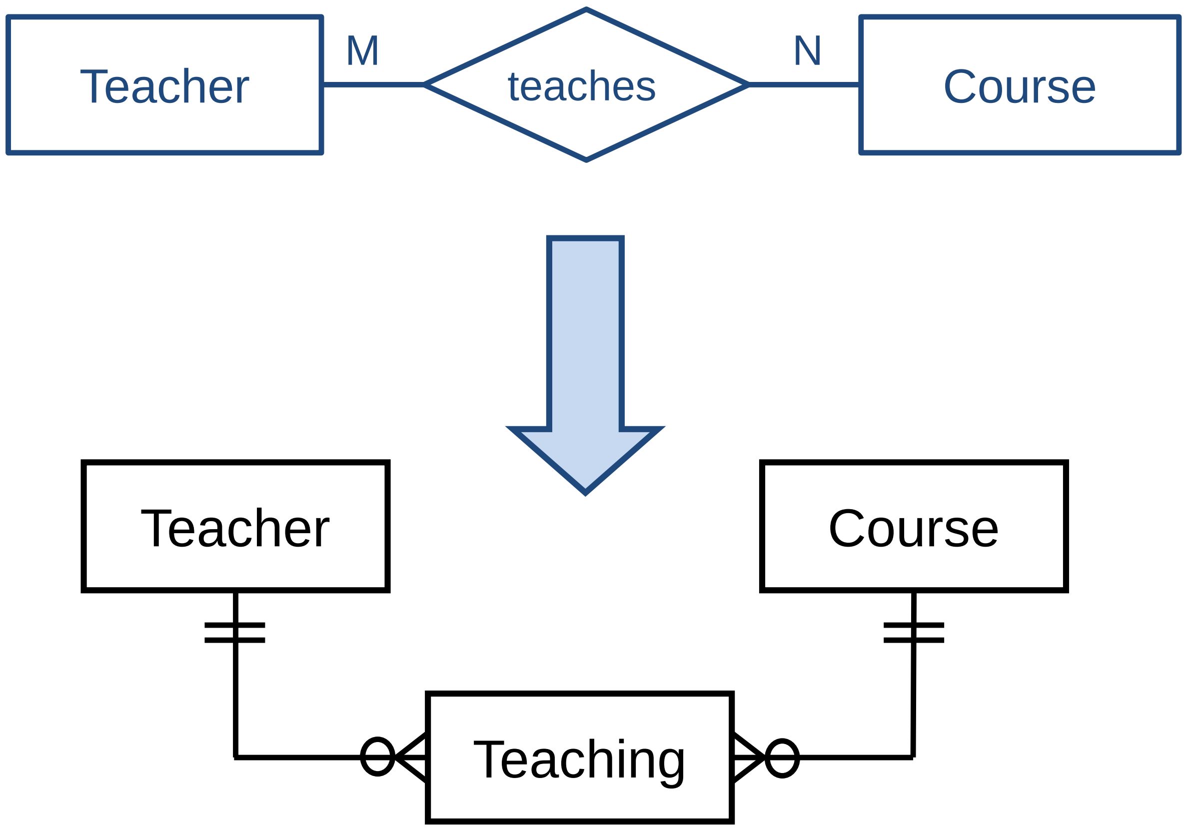


Association entity…

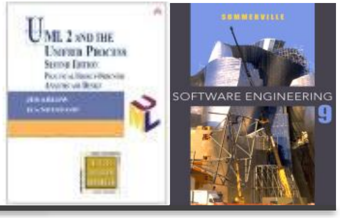
… can become
an entity on its own

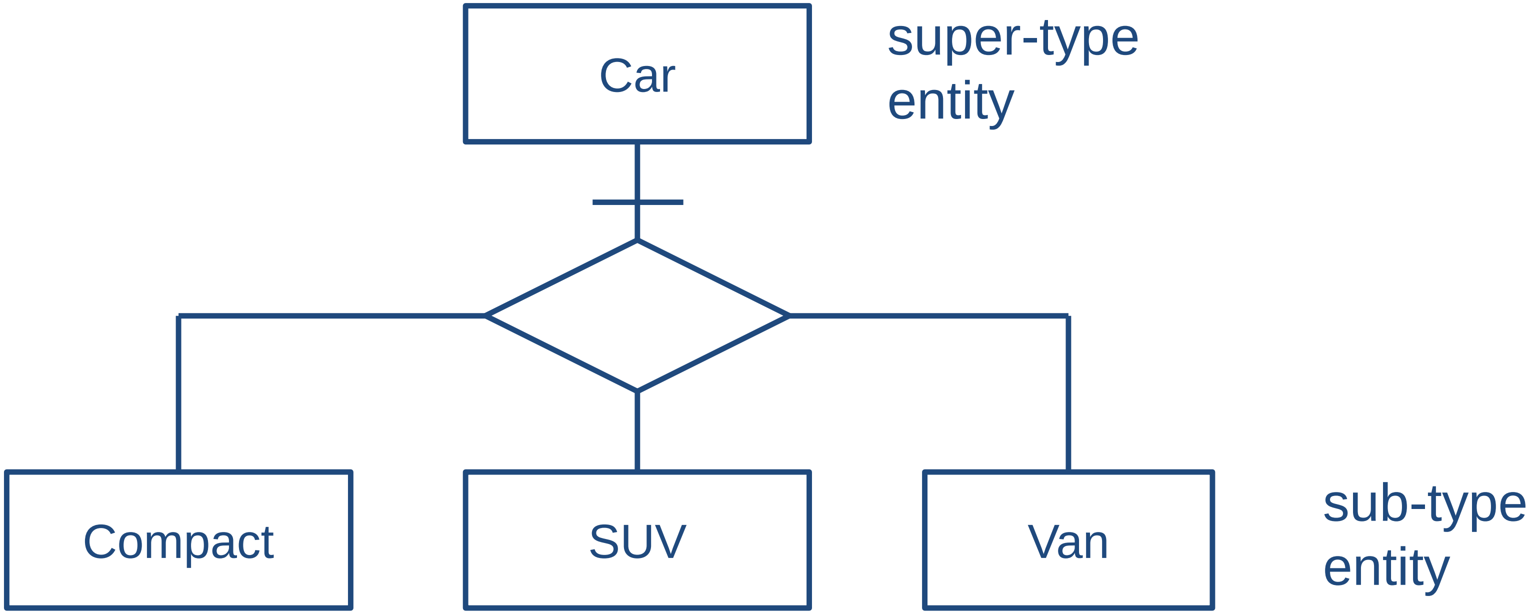


M:N relationships



Sub-types and super-types

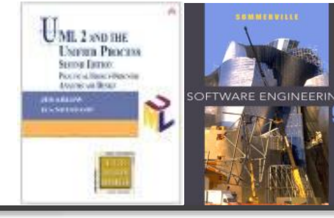




✧ Three options:

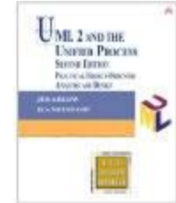
- One big Car entity with all attributes
- Three smaller Compact, SUV and Van entities
- Four entities with relationship between sub-type and super-type entity

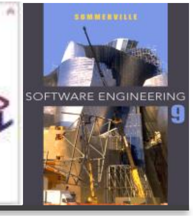
Database design process (continued)



3. **Specify primary keys** - Choose each table's primary key. The primary key is a column that is used to uniquely identify each row. An example might be Product ID or Order ID.
4. **Apply the normalization rules** - Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the tables.
5. **Refine the design** - Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.

Entities and keys





✧ Superkey

- A set of attributes that **uniquely identifies** each entity.

✧ Candidate key

- A **non-redundant** superkey, i.e. all items of a candidate key are necessary to identify an entity, no key attribute can be removed.
- There can be more combinations of entity attributes that can be used as candidate keys.

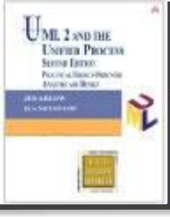
✧ Primary key

- The **selected candidate key**, marked with # symbol.

✧ Foreign key

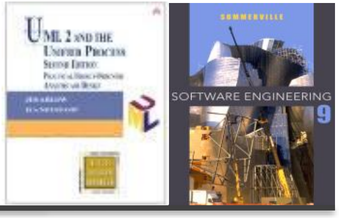
- A set of attributes in one entity that **uniquely identifies** (i.e. is a primary key in) **another entity**.

Data normalization goals by E.F. Codd

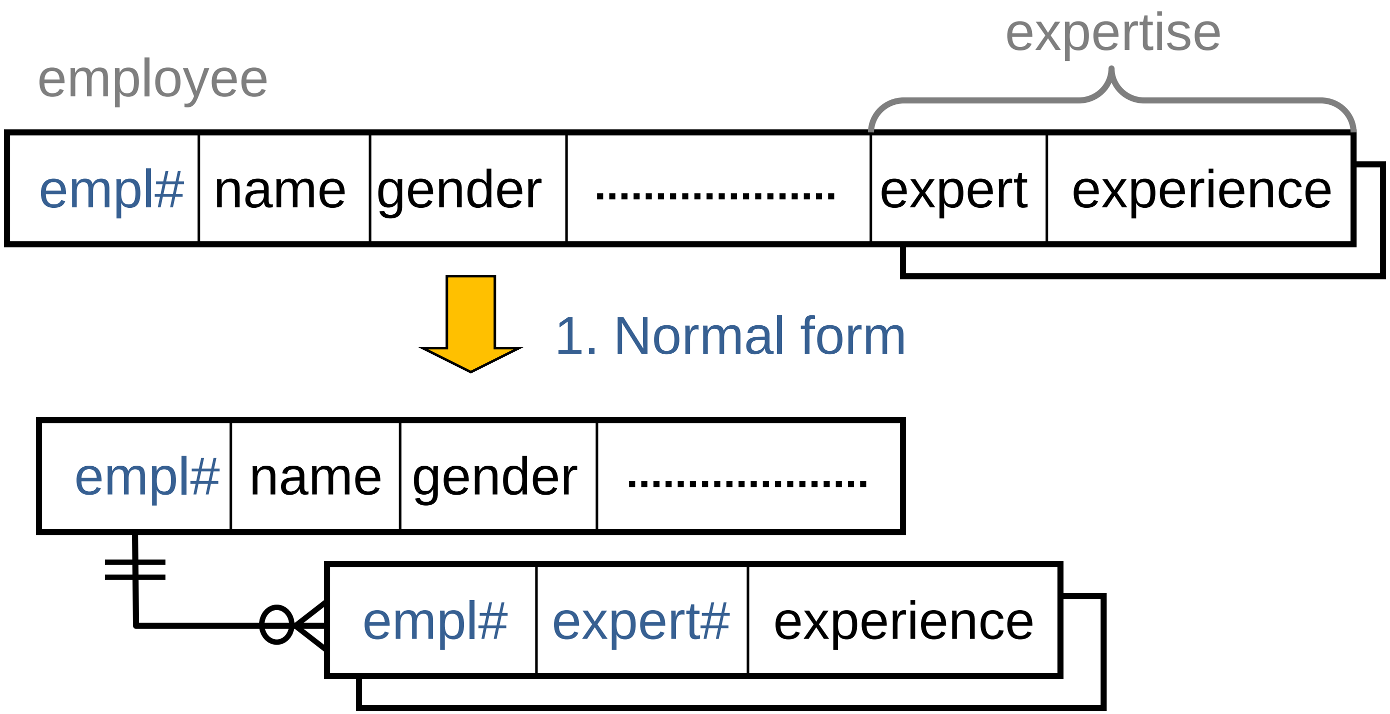


- Minimize **redundancy** and **dependency**
  - Minimize redesign when extending database structure
  - Make the data model more informative to users
- Free the database of modification anomalies
  - **Update anomaly** – the same information expressed on multiple rows → update resulting in logical inconsistencies.
  - **Insertion anomaly** – certain facts cannot be recorded, because of their binding with another information into one record.
  - **Deletion anomaly** – deletion of data representing certain facts necessitating deletion of unrelated data.
- Avoid bias towards any particular **pattern of querying**

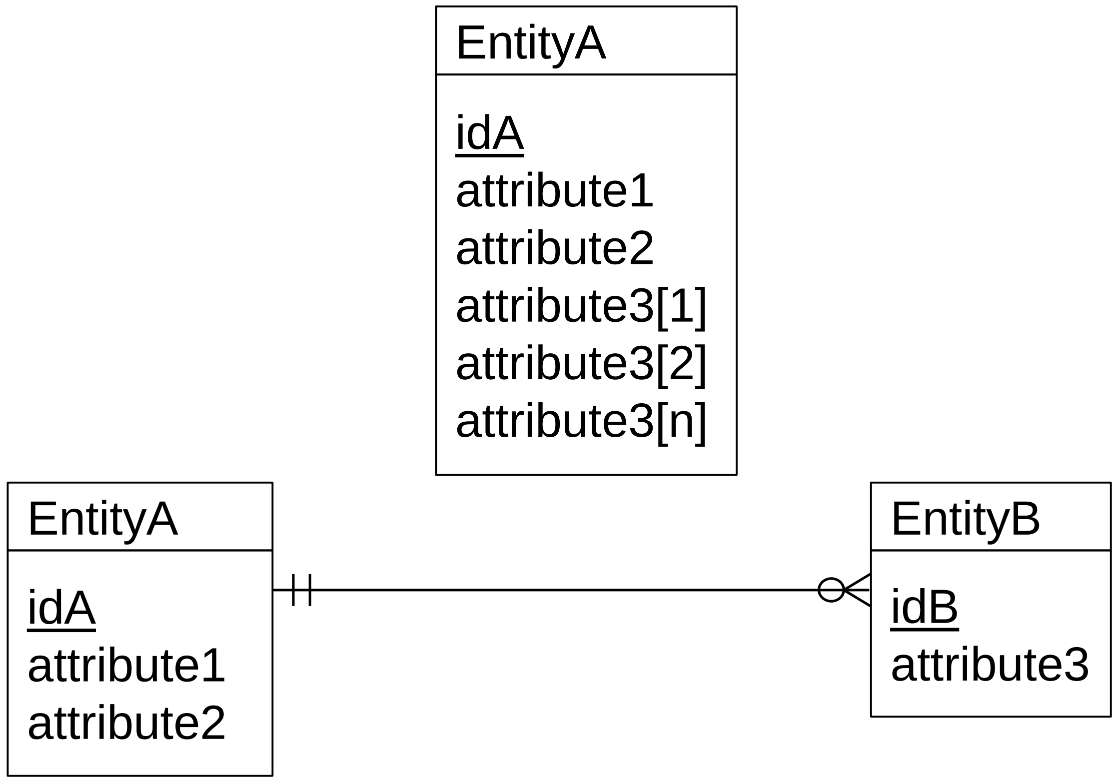
1. Normal form – no repeating groups



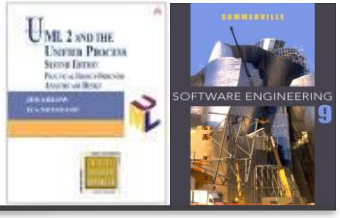
Def.1NF: A relation is in 1NF if the domain of each attribute contains only **atomic values**, and the value of each attribute contains only a **single value** from that domain.



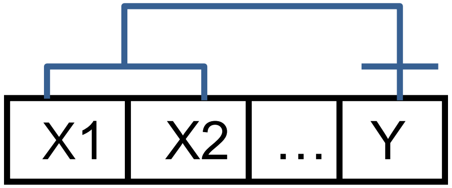
1. Normal form – normalization example



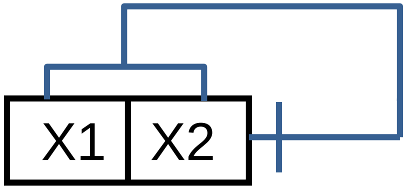
Functional dependency



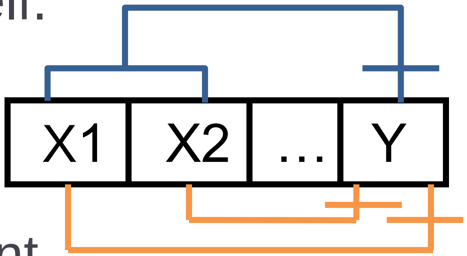
- Functional dependency
  - In a given table, an attribute Y is said to have a functional dependency on a set of attributes X if and only if each X value is associated with precisely one Y value.



- Trivial functional dependency
  - A trivial functional dependency is a functional dependency of an attribute on a superset of itself.



- Full functional dependency
  - An attribute is fully functionally dependent on a set of attributes X if it is: functionally dependent on X, and not functionally dependent on any proper subset of X.



2. Normal form – no partial dependency

Def. 2NF: In 1NF and no non-prime attribute in the table is functionally dependent on a proper subset of any candidate key.

Example EMPLOYEE

Is in 2NF

empl#	name	salary	project	deadline

Example PROGRAMING

Not in 2NF

developer#	package#	developer name	package name	hours

What anomalies can you identify in this example?

2. Normal form – no partial dependency

not part of any candidate key

Def. 2NF: In 1NF and no non-prime attribute in the table is functionally dependent on a proper subset of any candidate key.

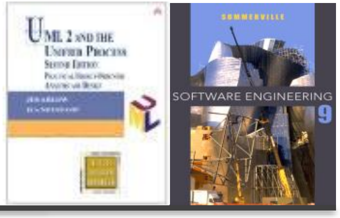
- Does the "candidate key" part of the definition make difference?
- When there is only one-item primary key, is 2NF guaranteed?

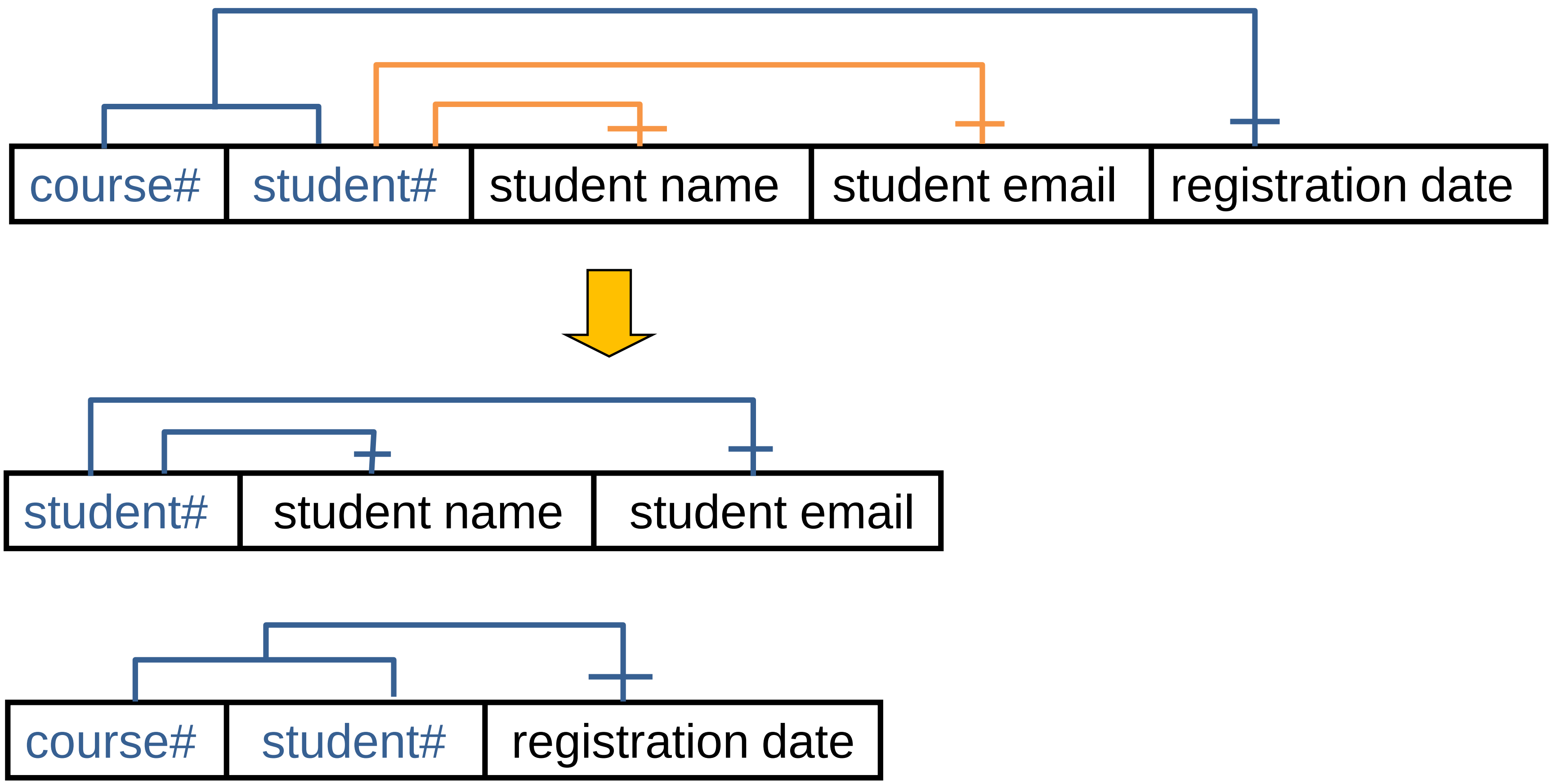
Example DISHWASHER MODELS

Not in 2NF

manufacturer	model	model full name#	manufacturer country

2. Normal form – normalization example

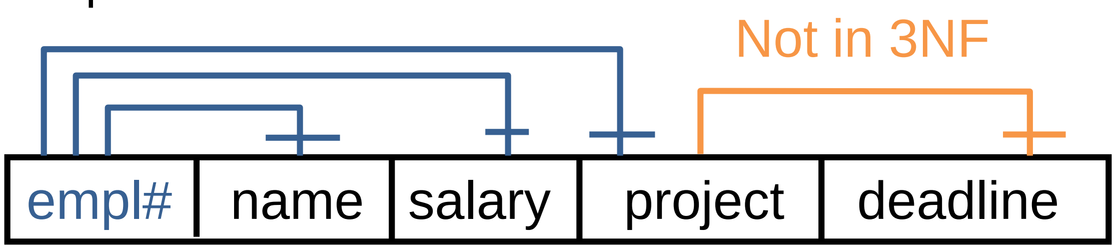




3. Normal form – no transitive dependency

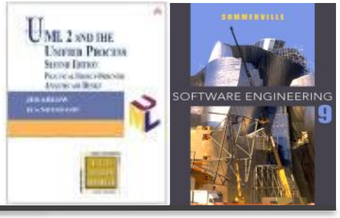
Def. 3NF: In 2NF and every non-prime attribute is non-transitively (i.e. only directly) dependent on every candidate key.

Example EMPLOYEE

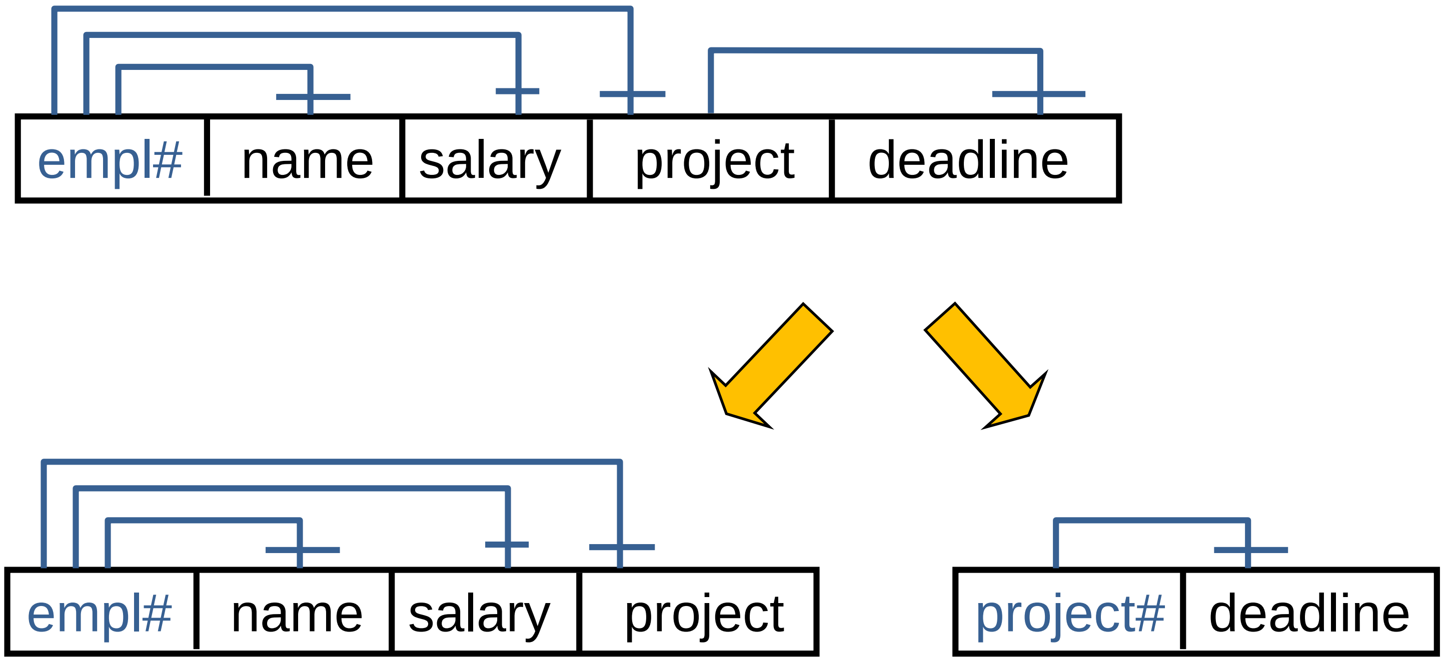


What anomalies can you identify in this example?

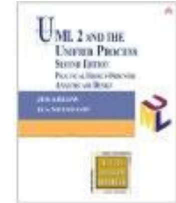
3. Normal form – normalization example



deadline is transitively dependent on empl#



ERD vs. UML Class Diagram



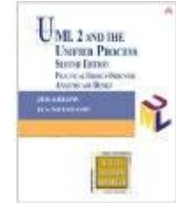
✧ Class diagrams

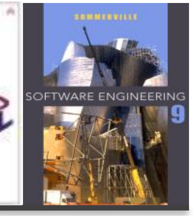
- model both **structural and behavior features** of a system (attribute and operations),
- contain **many different types of relationships** (association, aggregation, composition, dependency, generalization), and
- are more likely to **map into real-world objects**.

✧ Entity relationship models

- model only **structural data view** with a low variety of relationships (simple relations and rarely generalization), and
- are more likely to **map into database tables** (repetitive records).
- They allow us to design **primary and foreign entity keys**, and used to be normalized to simplify data manipulation.

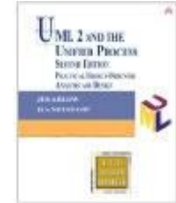
ERD vs. UML Class Diagram





- Although there can be one to one mapping between ERD and Class diagram, it is very common that
  - one class is mapped to more than one entity, or
  - more classes are mapped to a single entity.
- Furthermore, not all classes need to be persistent and hence reflected in the ERD model, which uses to be driven by the database design.
- **Summary:**
  - ERD is **data-oriented** and **persistence-specific**
  - Class diagram targets also operations and is persistence independent

Key points

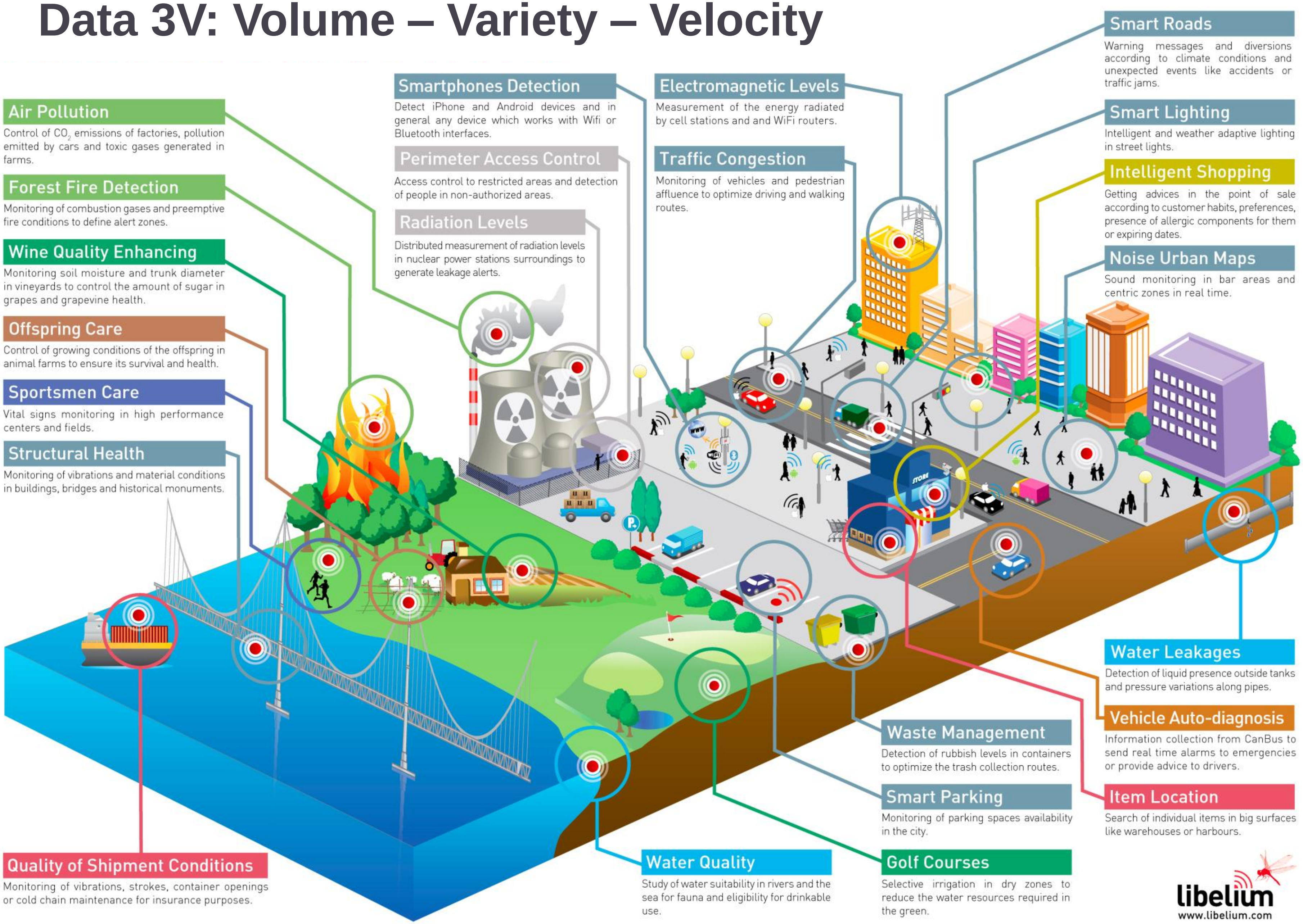


- Data modeling, and ERD in particular, focuses on modeling data **entities, relationships and attributes**.
- Data normalization focuses on reducing **redundancy and dependency** in database design, and on avoiding bias towards a particular **pattern of querying**.
  - 1NF: no repeating groups
  - 2NF: no partial dependency
  - 3NF: no transitive dependency

Other database concepts

Lecture 5/Part 4

Data 3V: Volume – Variety – Velocity



Other database concepts

Storage strategies:

- ✧ Relational vs. NoSQL databases
- ✧ Key/value stores
- ✧ Document databases
- ✧ Graph databases

Related concepts:

- ✧ Cloud computing
- ✧ Object Relationship Mapping (ORM)