



# Lecture 07: Data modelling

---

1. *Data modelling*
2. *Hierarchies*
3. *Specialization*
4. *Generalization*
5. *Grouping*
6. *Type- and instance level*
7. *Use the principle of an e-message to model changes*
8. *Homework: Modelling assignment*

Pär Douhan, [pdo@du.se](mailto:pdo@du.se)

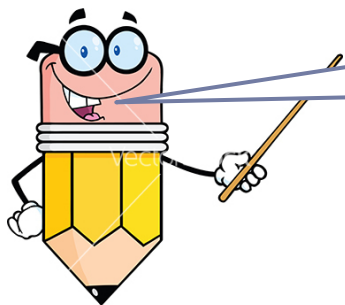


# Data modelling

---

## Business description

We have an e-service, on a website, where customers can log in, and shopping for goods by putting items in a shopping cart. One customer can, for a period, visit the site and shop repeatedly.

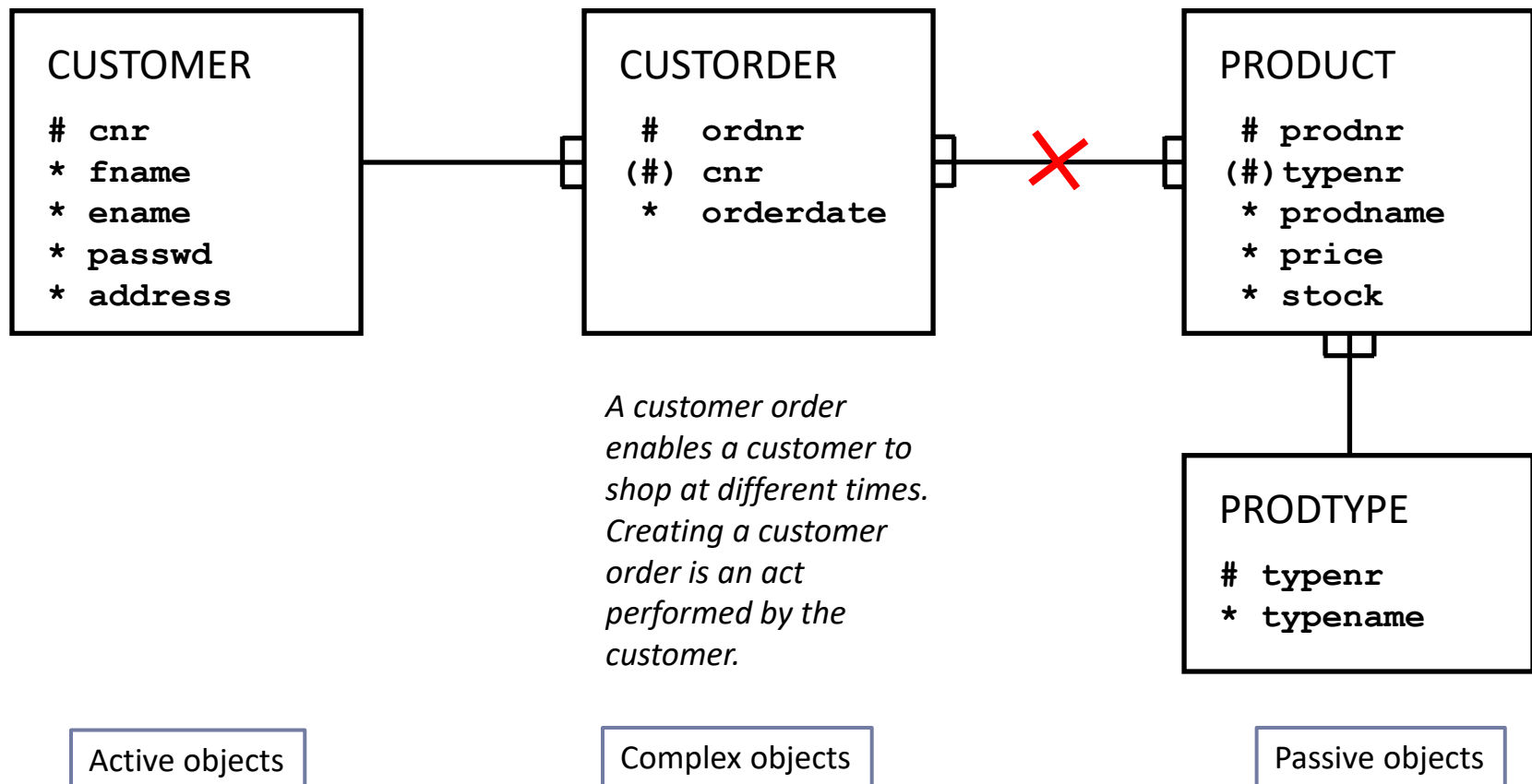


Sounds like we can use a basic model for trading!



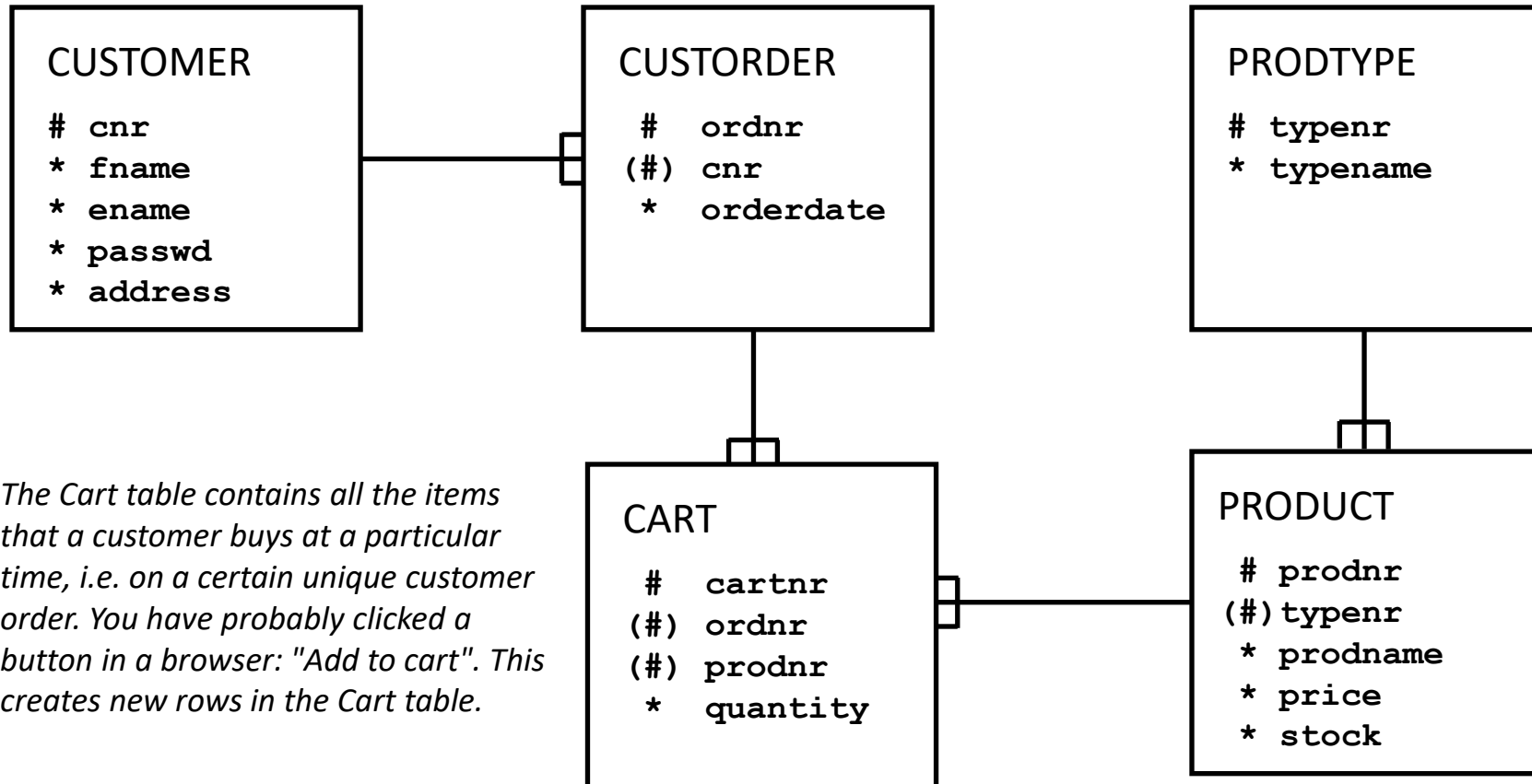
# Example model

We can quickly identify, at least, two objects:





# Example model



*The Cart table contains all the items that a customer buys at a particular time, i.e. on a certain unique customer order. You have probably clicked a button in a browser: "Add to cart". This creates new rows in the Cart table.*

Active objects

Complex objects

Passive objects



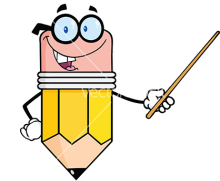
# Tables with sample data

**CUSTORDER**

ordnr	kundnr	datum
2000	2556	20150126
2001	2556	20150203
2002	3056	20150315

**PRODTYPE**

typenr	typename
23	sport
33	kitchen
43	car



**CART**

cartnr	ordnr	prodnr	quantity
702564	2000	400	2
702565	2000	800	3
702567	2001	800	5
702569	2002	450	3
702572	2002	800	15

**PRODUCT**

prodnr	typenr	artname	priceEUR	stock
400	23	football	60	500
450	33	strainer	3.9	126
800	43	glycol	10	50
850	43	washer fluid	4.3	200
520	33	drainer	14.9	0



# Hierarchies

KVINNA **MAN** BARN

News & Style Skor Kläder Sport & träning Väskor & accessoarer Premium Märken Rea Hitta

Startsida > Man > Kläder > Jackor

**Kategori**


- > Kläder
  - > T-shirts & piké
  - > Skjortor
  - > Tröjor & kofter
  - > Jeans
  - > Byxor & shorts
  - > Jackor**
    - Tunn jackor
    - Skinnjackor
    - Jeansjackor
    - Kavajer
    - Outdoorjackor
    - Träningsjackor
    - Fleecejackor
    - Västar
    - Vinterjackor
    - Dunjackor
  - > Rockar
  - > Kostymer
  - > Träningskläder
  - > Badmode
  - > Underkläder
  - > Strumpor

**Jackor** (2254 artiklar)


Varumärke Färg Pris Storlek

Mönster Foder Kollektion Nya varor

Sortera efter: Popularitet



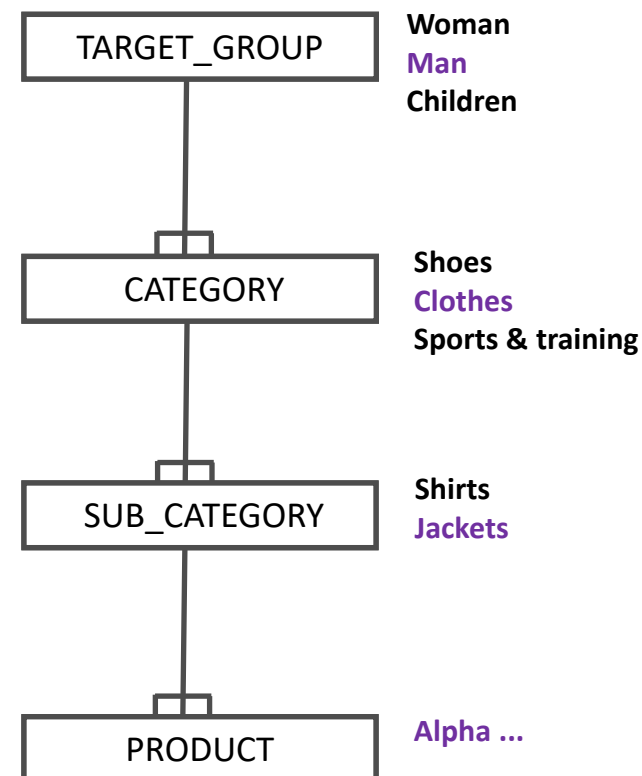
**Alpha Industries**  
MA1 TT - Tunn jacka - replica blue  
1 395 kr



**Selected Homme**  
Skinnjacka - black  
1 995 kr

Jackor

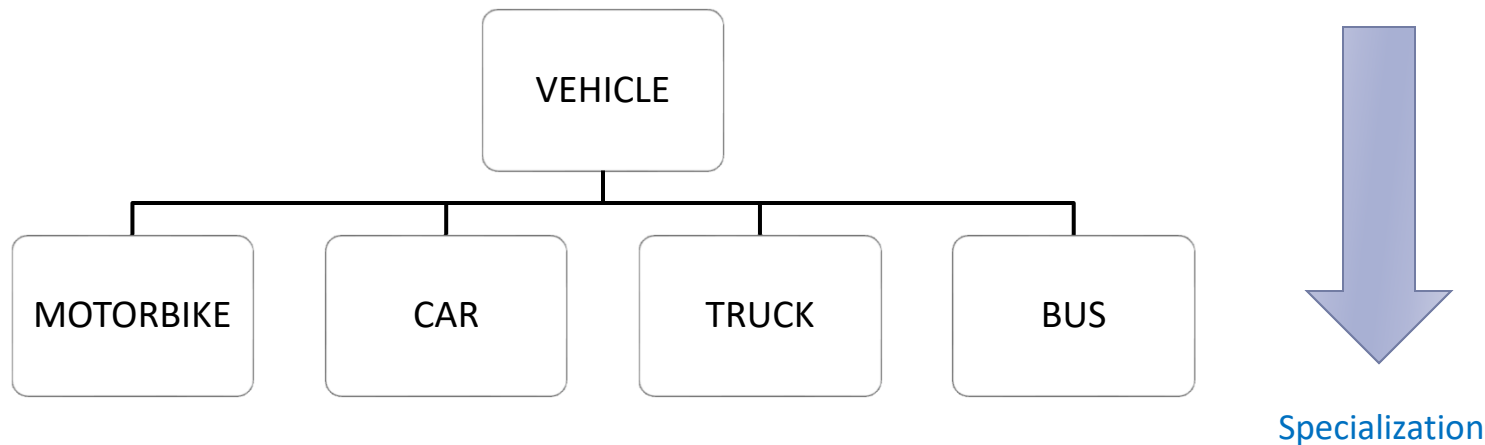
Alla plagg är viktiga men vissa är helt enkelt viktigare än



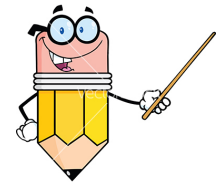


# Specialization, Top Down

---



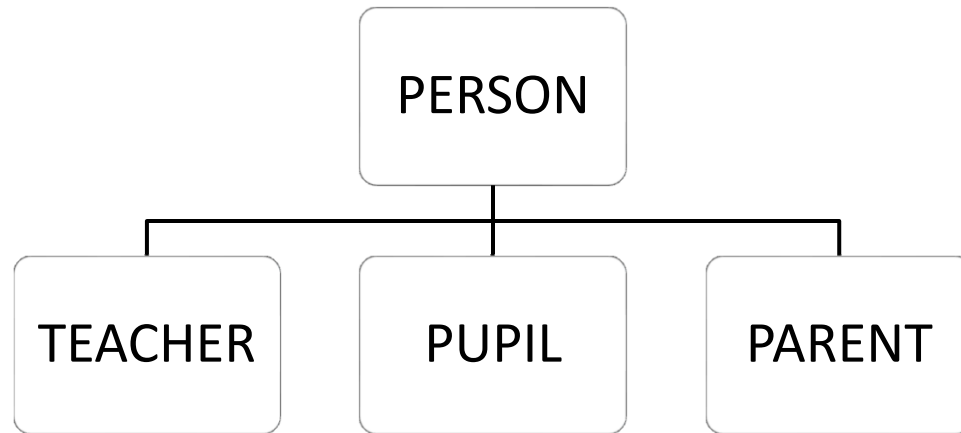
- Motorcycle, car, truck and bus are specializations of the type Vehicle.
- Not all object types have exactly the same properties.
- A motorcycle has different properties than a bus.
- However, some characteristics are common and are found in the object type of vehicle.



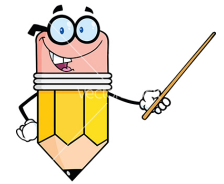
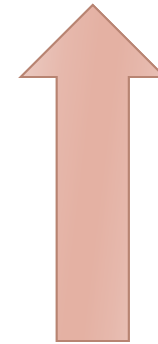


# Generalization, Bottom Up

---



Generalization

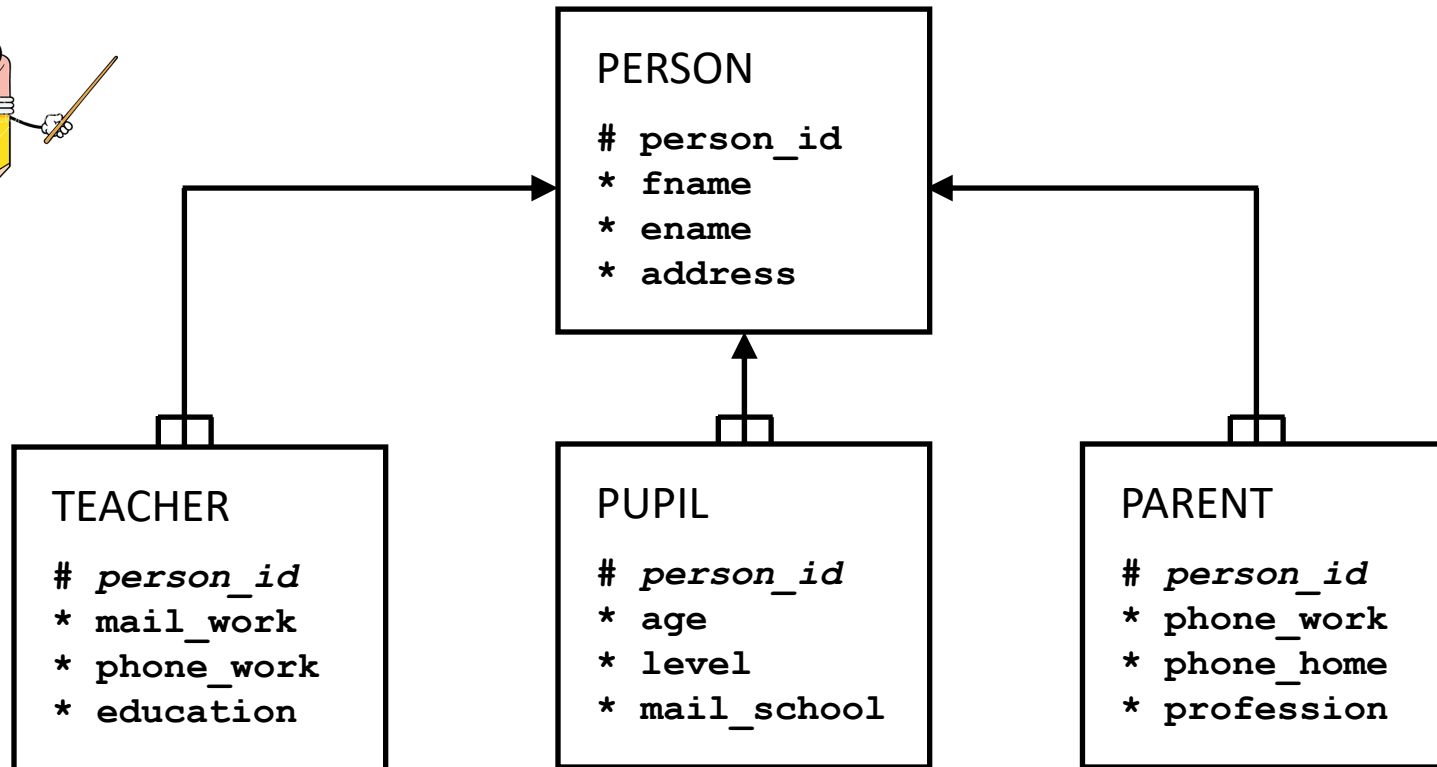
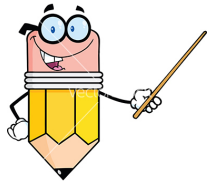


- We can use generalization if two or more object types have partially common variable sets. If you then find that the object types can in some reasonable sense be sub types of a common super type, then you can define and name this super type to assign the common variables to it.
- By generalizing object types you make it possible to provide more transparent object graphs.
- At an overview level, subdivisions of object types can be omitted, which can instead be reported in supplementary detail graphs for different parts of the object system.





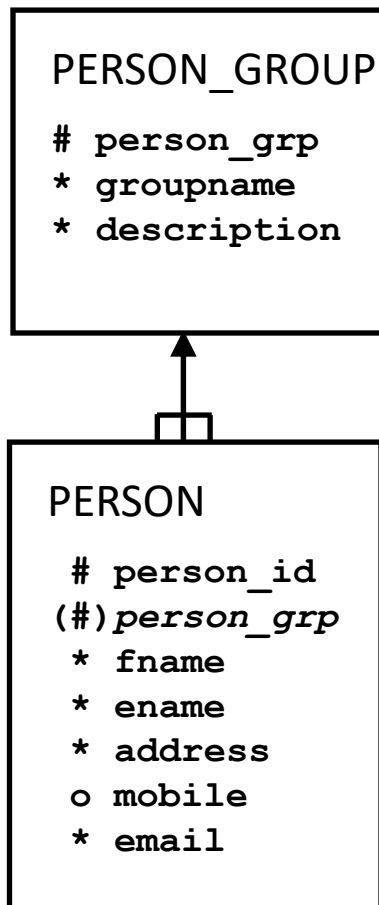
# Keys?



- The parent table Person's PK becomes PK and FK in the child tables.
- The column Person\_id is both PK and FK in the tables Teacher, Student and Parent.

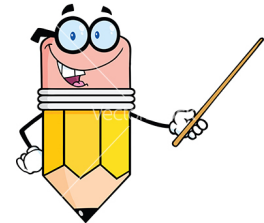


# Grouping - a simpler solution



Examples of person types:

- **Teacher**
- **Pupil**
- **Parent**

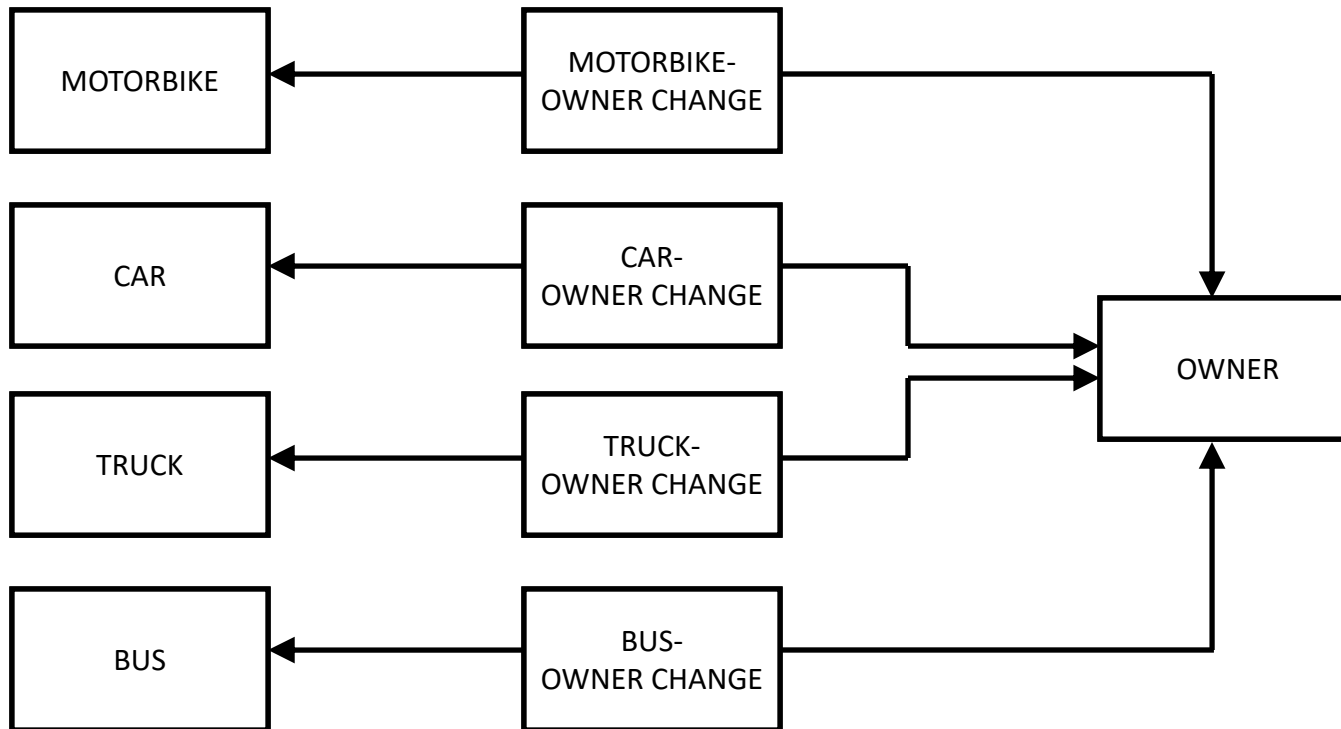


- A simpler and more flexible solution is to let all people have the same set of attributes.
- In the previous example, all subgroups of Person had partially unique attributes.
- If this is not required, we can instead assign all persons to a certain group of persons: teacher, pupil or parent.
- Here all people have the same attributes. This way does not allow certain types of persons, e.g. teachers have their own unique characteristics.



# Incorrect generalization level

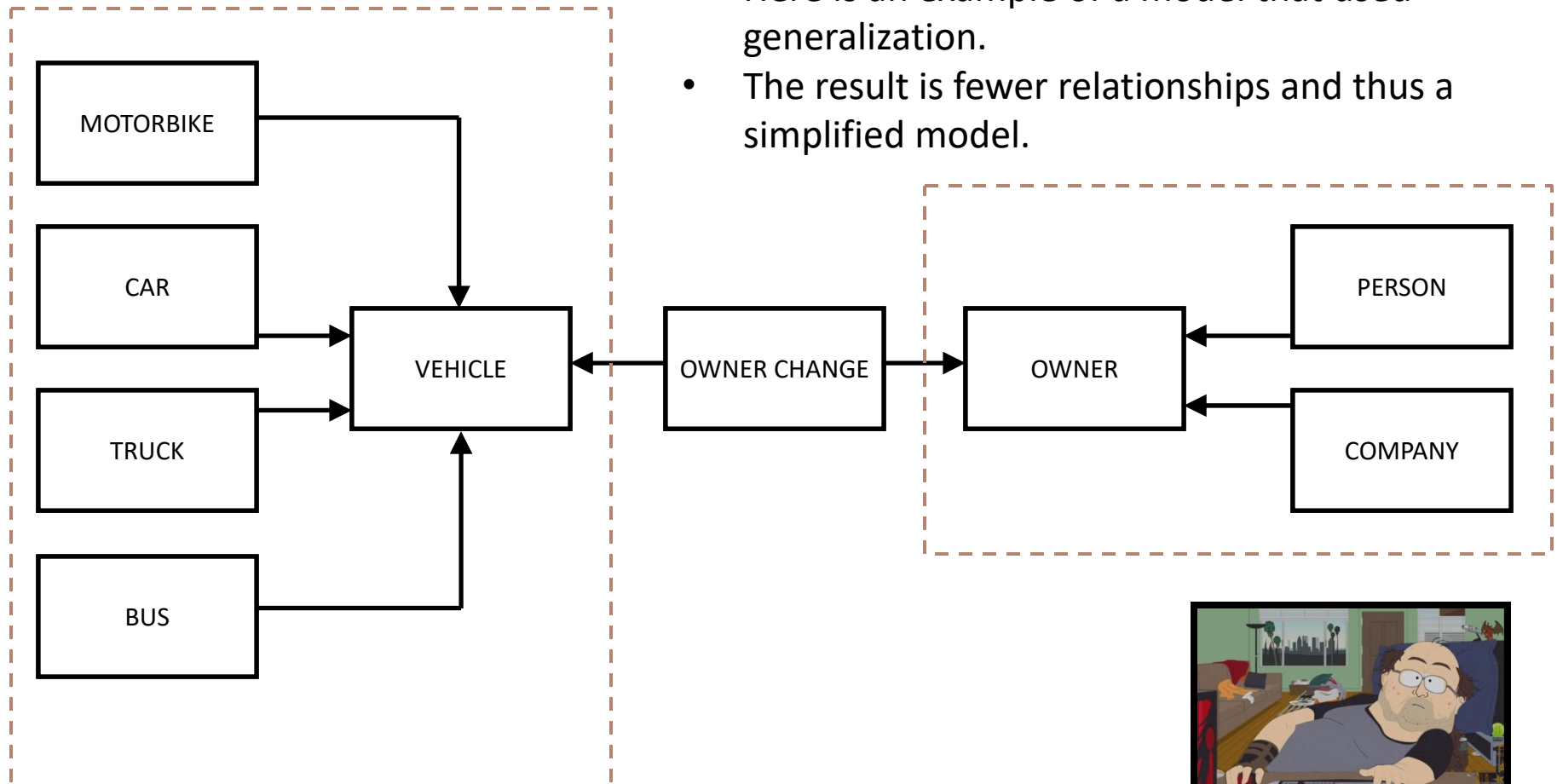
Incorrect generalization level can give very complex models:





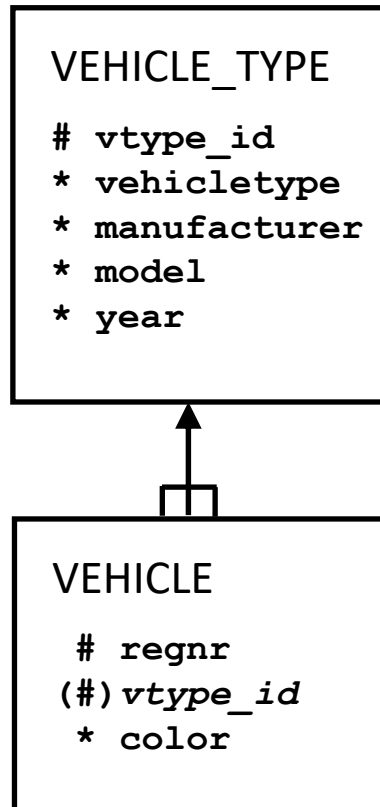
# Example of generalization

- Here is an example of a model that used generalization.
- The result is fewer relationships and thus a simplified model.

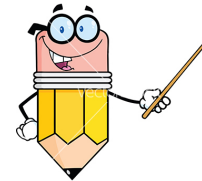




# An even simpler solution



- Examples of vehicle types:
- **Motorbike**
- **Car**
- **Truck**
- **Bus**

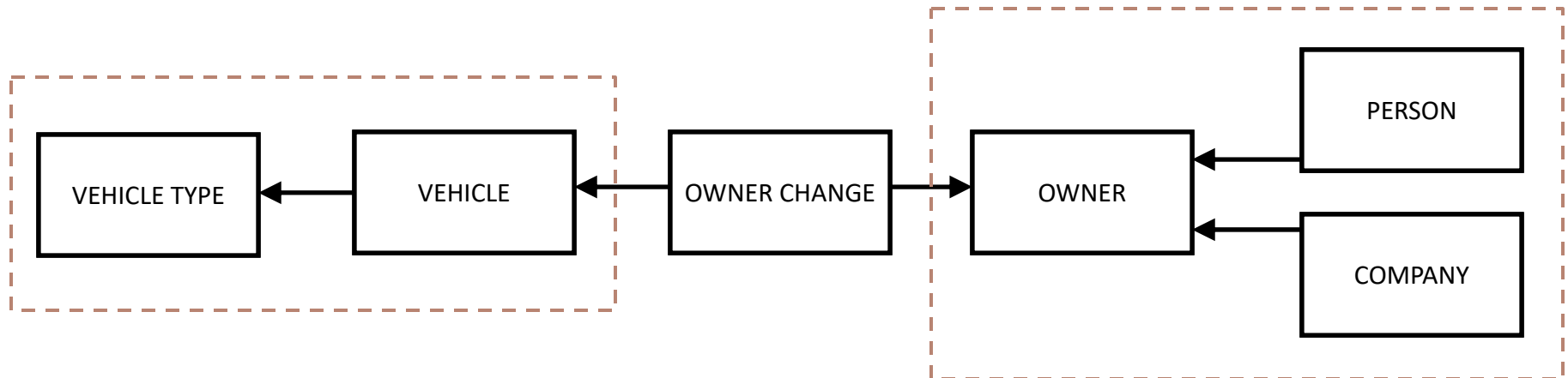


- Here are all characteristics that are unique to each vehicle copies.
- The vtype\_id will tell what kind of vehicle it is.



# Example

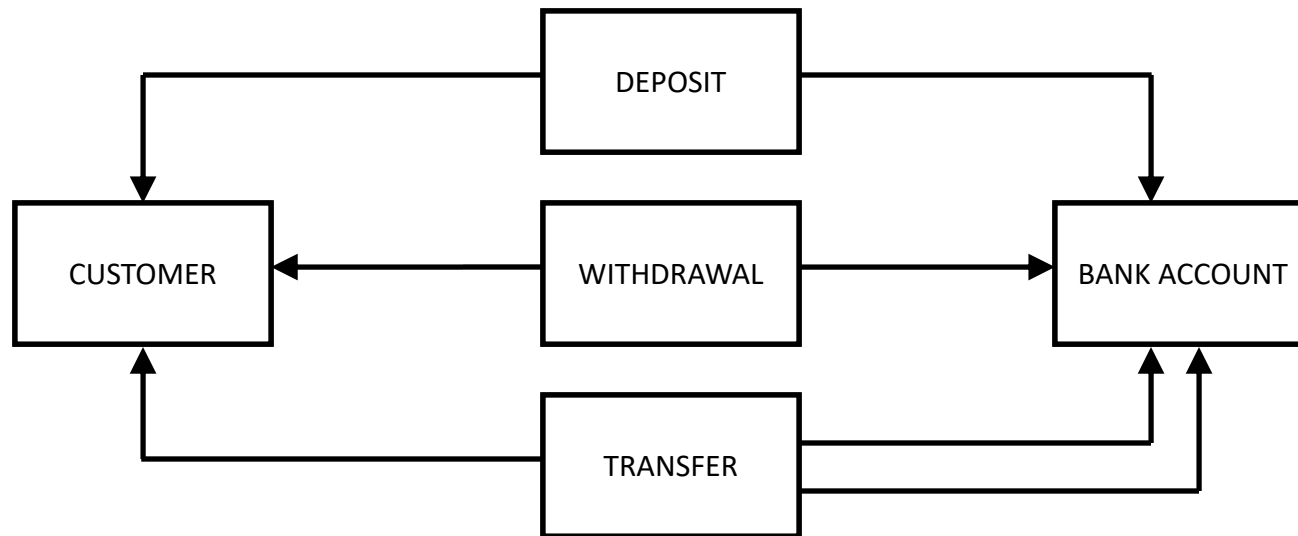
- By connecting all the variables for the different vehicles to a common type, we get an even simpler model.
- Individuals and companies differ so much that it is not possible to group these into an owner type.





# Clear models

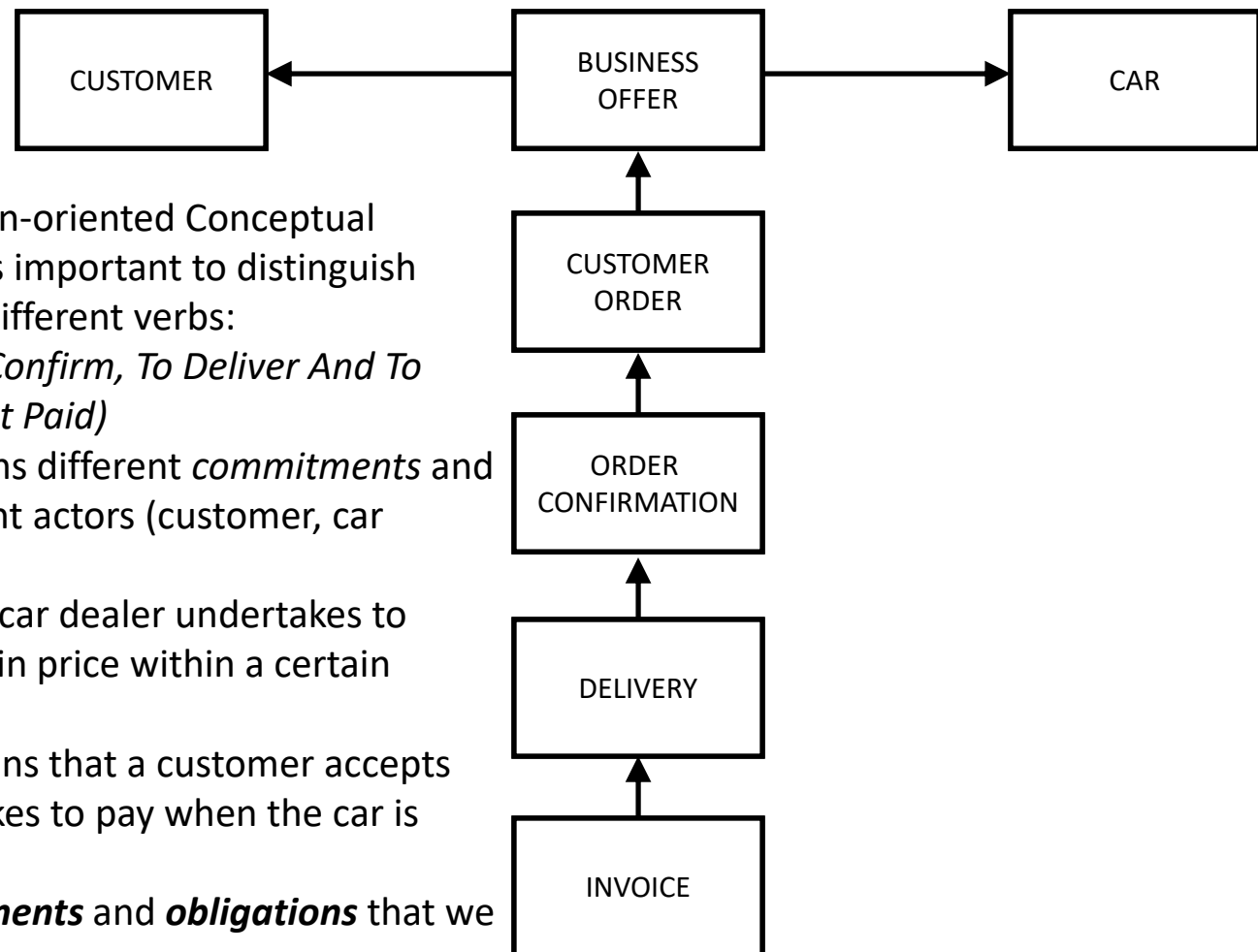
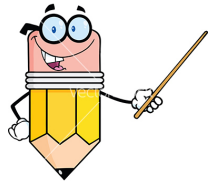
---



- Here the data model becomes clear.
- We see what actions customers can perform on their bank accounts.



# Actions

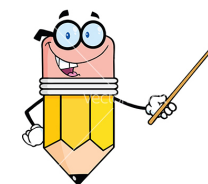
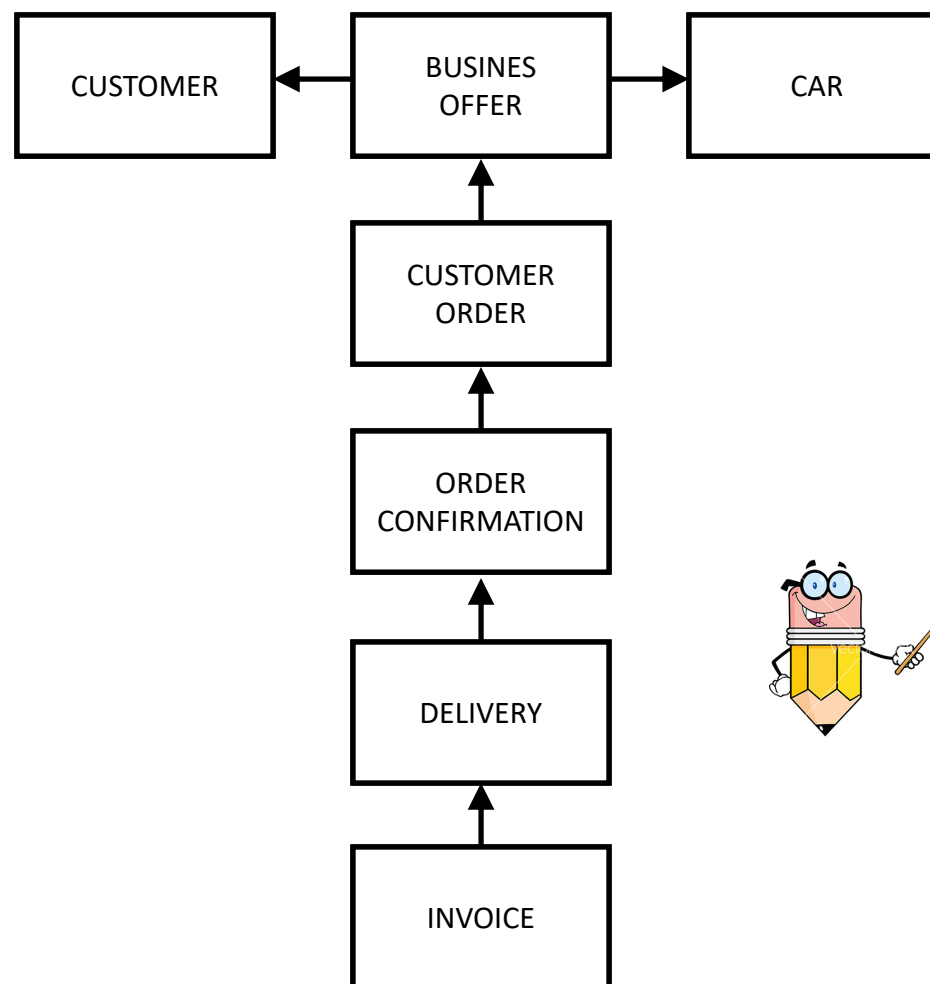
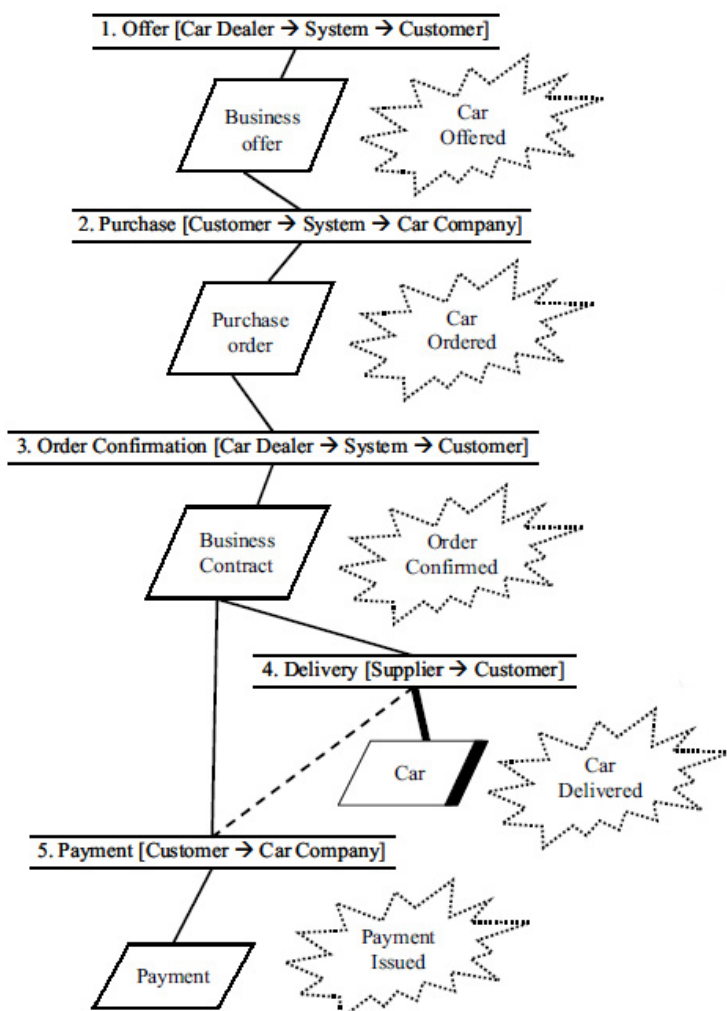


- According to the Action-oriented Conceptual Modeling method, it is important to distinguish objects representing different verbs:
- *To Offer, To Order, To Confirm, To Deliver And To Invoice (Request To Get Paid)*
- This is because it means different *commitments* and *obligations* for different actors (customer, car dealer).
- An offer means that a car dealer undertakes to deliver a car at a certain price within a certain period of time.
- A customer order means that a customer accepts the offer and undertakes to pay when the car is delivered.
- It is different *commitments* and *obligations* that we model.



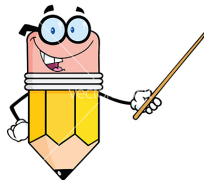
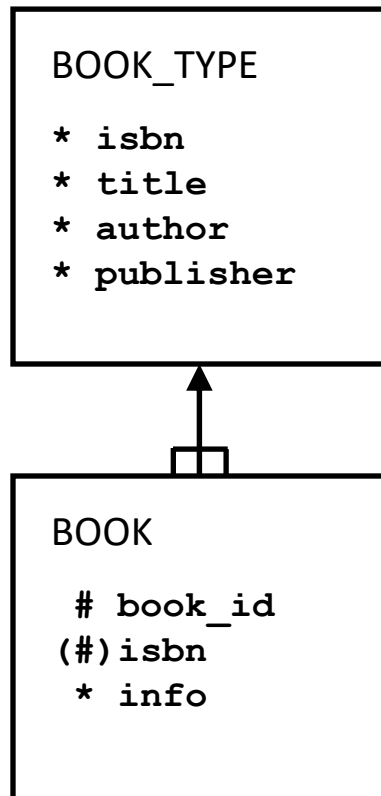


# Clear connection to the process





# Type and instance level



- In this example, there are **book types** and **book copies** in a library.
- It would just as well be movie types and movie copies in a movie rental store.
- Or car types and cars in a car rental company.
- At the type level, all information that is common to all copies is stored. A book type can exist without any instances (book copies) belonging to the book type.
- If we were to store information about the title, author and publisher for all book copies, we would get lots of physical redundancy.
- Imagine if we have 200 book copies of the book type with ISBN 186100690X, "Beginning Oracle Programming", "Thomas Kyte", "Wrox".
- We would then need to store the same information in 200 rows.

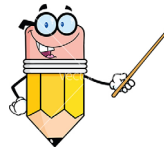


# Physical redundancy

## BOOK

book_id	isbn	title	author	publisher
22235556	1-861006-90-X	Beginning Oracle Programming	Thomas Kyte	Wrox
22235557	1-861006-90-X	Beginning Oracle Programming	Thomas Kyte	Wrox
22235558	1-861006-90-X	Beginning Oracle Programming	Thomas Kyte	Wrox
22235559	91-44-35991-8	Database systems	Bo Sundgren	Studentlitteratur
22235560	91-44-35991-8	Database systems	Bo Sundgren	Studentlitteratur

- Here we see a clear example of **physical redundancy**.
- This is a consequence of the fact that we have not made a difference between the type and the instance level.
- The table is in **2NF**, this as there is a partial dependency between the column isbn and (title, author and publisher).
- This causes physical redundancy.

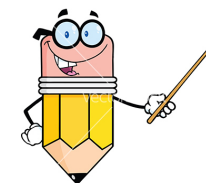




# Type and instance level

## BOOK\_TYPE

isbn	title	author	publisher
186100690X	Beginning Oracle Programming	Thomas Kyte	Wrox
9144359918	Database systems	Bo Sundgren	Studentlitteratur
1861004826	Expert one-on-one	Thomas Kyte	Wrox



## BOOK

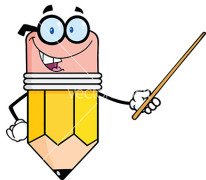
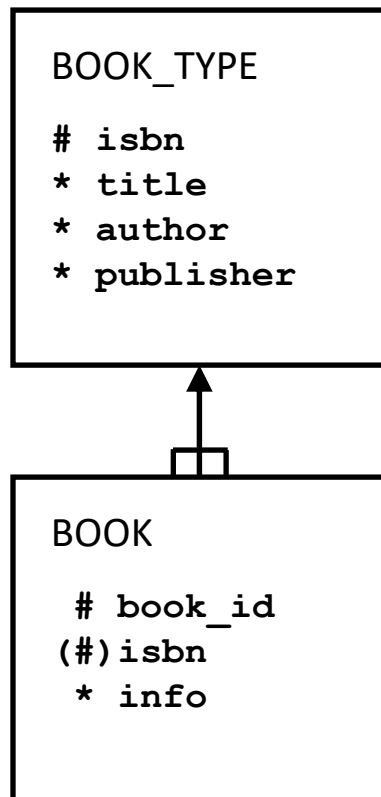
book_id	isbn	info
22235556	186100690X	ok
22235557	186100690X	Some pages have "dog ears"
22235558	186100690X	ok
22235896	9144359918	ok
22235896	9144359918	Very worn



- Now all physical redundancy is gone.
- Common information for the book copies is stored in the Book Type table.
- The model is in **3NF**.



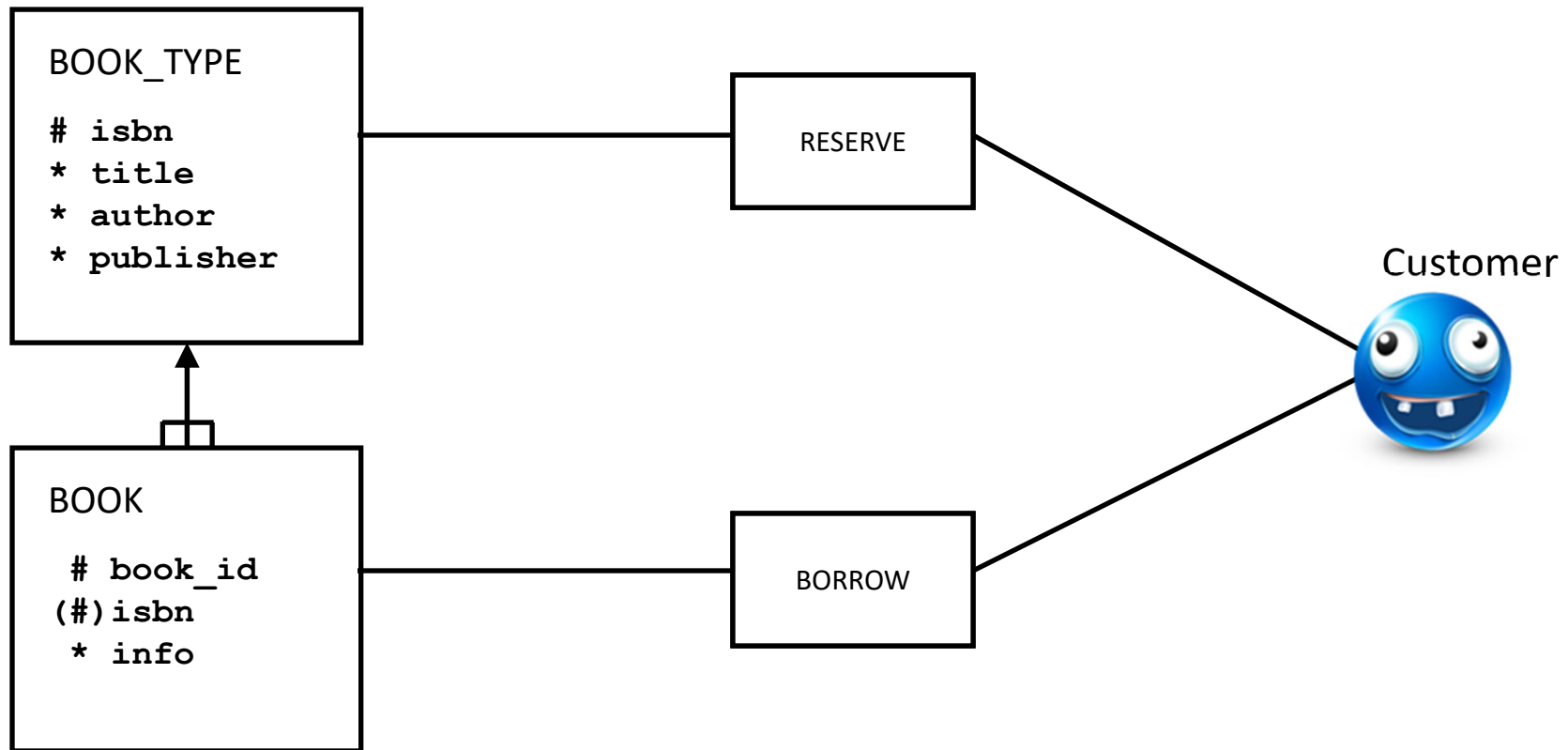
# Relate to the right level



- When creating data models, it is very important to relate certain objects to the right level.
- A customer who borrows a book at a library will leave the library with one or more physical book copies.
- In the data model, **a book loan** should be linked to the book copy, i.e. to **the instance level**.
- However, if a customer wishes to reserve one or more books, the reservation must be linked to the book type, i.e. to the type level.
- It does not matter to the customer what book copy he or she gets, mainly that it is the right book type.
- If a customer should reserve a book copy and this copy is never returned, the system will not be able to notify the customer that the reserved book has been available again.
- However, if a book type is reserved, the system will notify the customer, perhaps with an SMS, when a book copy of the reserved type has been returned.



# Relate to the right level



Keep this in mind in the data model!



# Business description

---

**Rental Car Ltd.** rents out passenger cars to companies. It is common for a company to rent several cars at the same time. To solve that problem, a rental order is always created. This consists of a unique rental order number, date, a reference to who in the staff who rented the car and which company the rental order refers to.

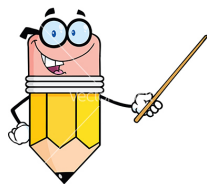
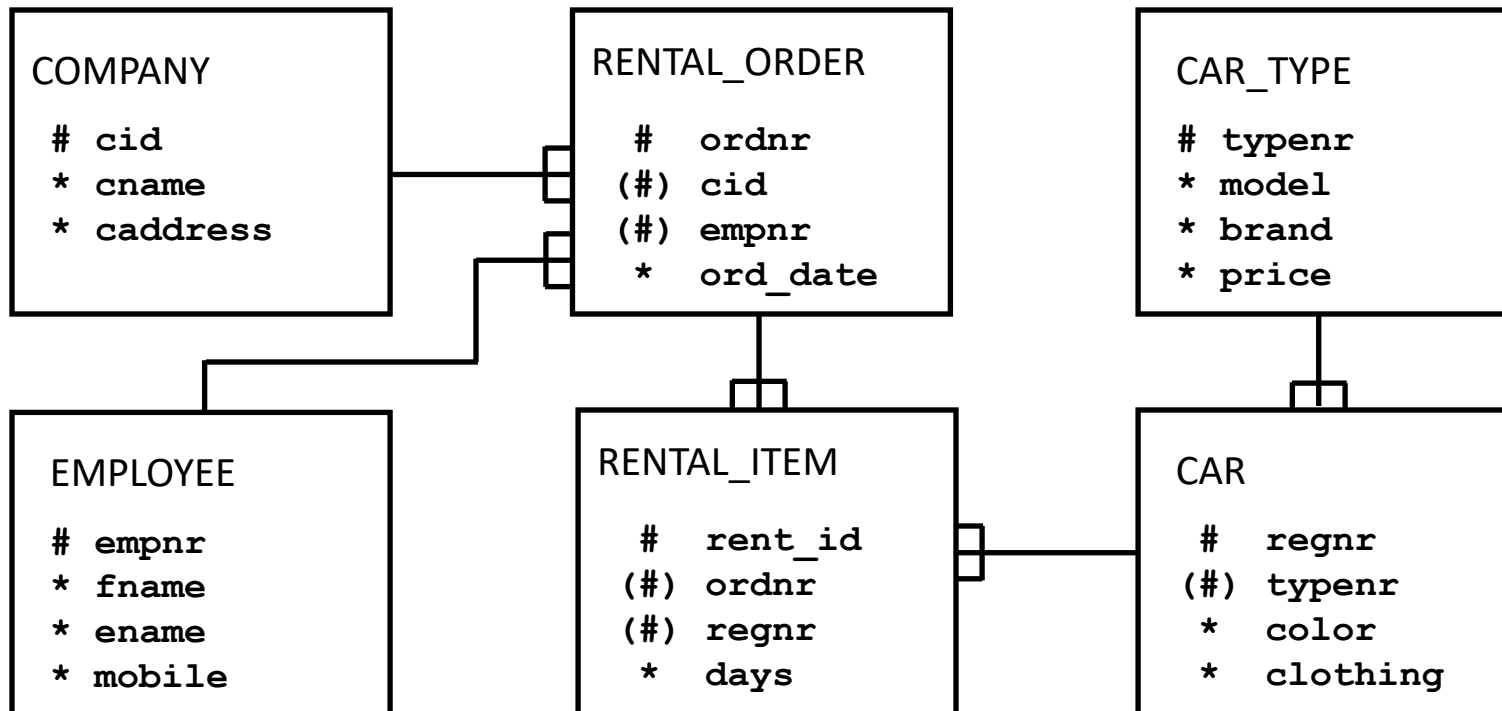
Rental Car Ltd. has a limited number of car types for hire. They only have models of Audi and Volvo. Audi has models A3, A4, A5 and A8. VOLVO models are S60, V60 and S90. Of one car type, there may be several different car models for rent. There are 12 car copies of the Audi A8. The only thing that sets the car apart is reg. number, color and type of clothing, leather or fabric. It is the same price for all car models of the same car type. This is whether the color is red, blue, black or white, or if the car's clothing is in leather or in fabric.

A rental order may include several different car copies. Each copy that is rented determines its own rental time. Information about what has been rented out is stored for at least 10 years.

Draw **PK = #**, **FK = (#)**, and **attributes** which is mentioned in the business description and the relationships between tables according to the model from the lectures. The data model must be in **3NF**.



# Data model of a car rental business

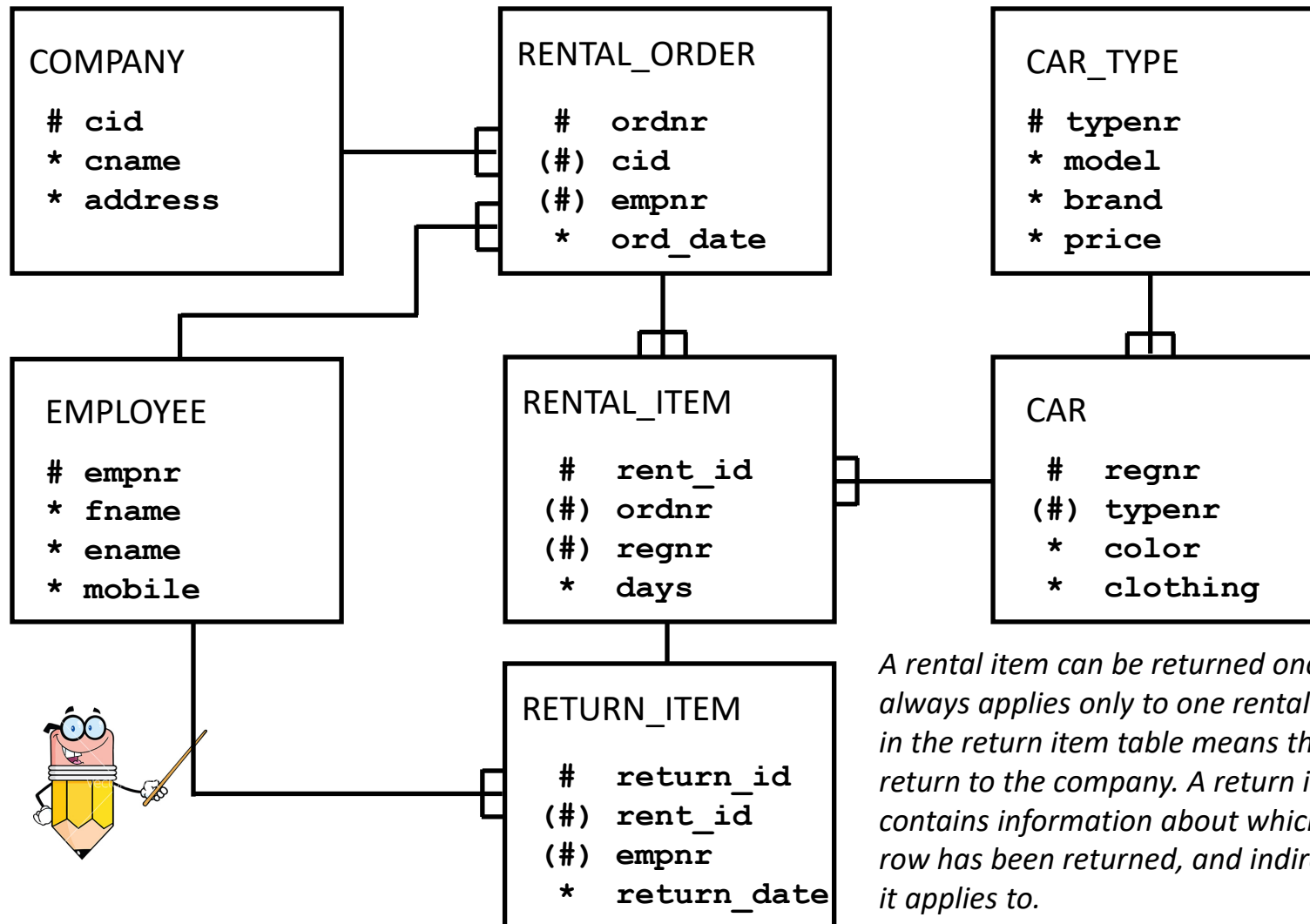


*An insert in the rental item table means that a car leaves the company for x days. A rental order item i.e. a row in the rental item table always contains information about which rental order it belongs to and which car has been rented out and how long it may be out. This model lacks support for car return.*





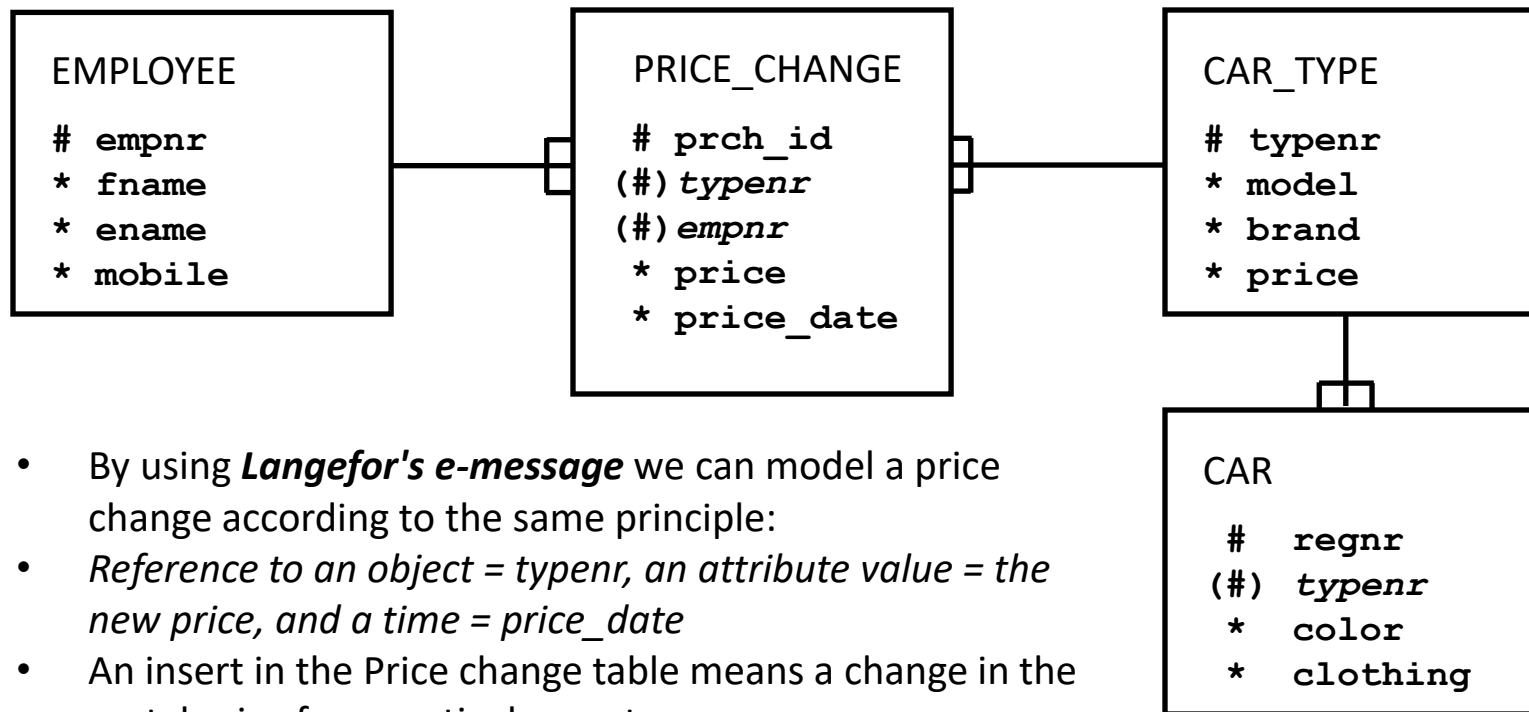
# Car rental – return car



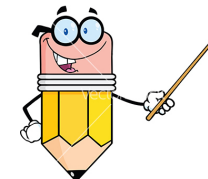
*A rental item can be returned once. A return always applies only to one rental item. An insert in the return item table means that a car will return to the company. A return item always contains information about which rental item row has been returned, and indirectly which car it applies to.*



# Price change



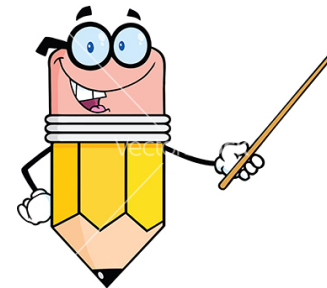
- By using **Langefor's e-message** we can model a price change according to the same principle:
- *Reference to an object = typenr, an attribute value = the new price, and a time = price\_date*
- An insert in the Price change table means a change in the rental price for a particular car type
- When a price changes, we update the price found in the Car type table with the new price (latest price).
- The Price Change table will contain all price changes, i.e. the whole price history. It is then used to send invoices with the right price.





## Try this at home 😊

- Sit down and try to make a data model of a movie business:
- The cinema is divided into different movie theaters.
- In the movie theaters there are rows of seats.
- Cinema show (activity performed by staff at the cinema)
- Ticket (Tickets are purchased for cinema shows and are linked to a certain chair)
- ***Where does the movie appear? How many tickets can you sell for a particular show?***



# The End

---

