LOGICAL DESIGN

Feistritzer Weinberger

Exercise 1

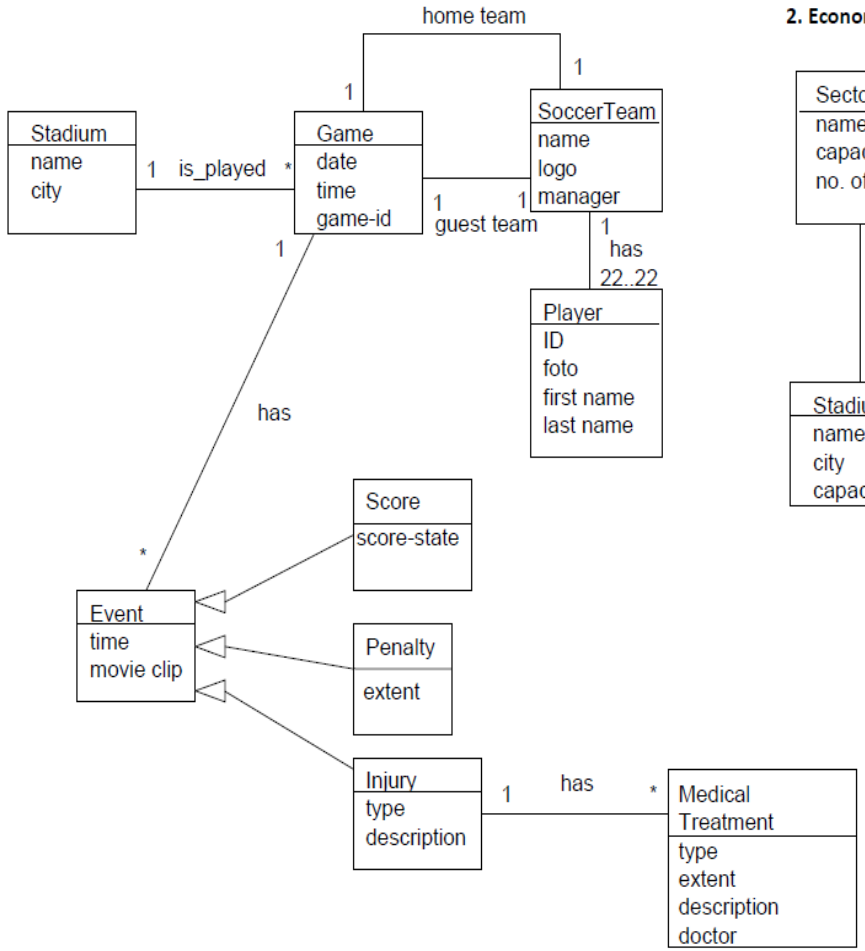
- Integrate these views:
  a) identify conflicts (and their conflict types)
  b) propose scenarios and interschema properties
  c) integrate the schemas into one schema.

Exercise 1 - Conflicts

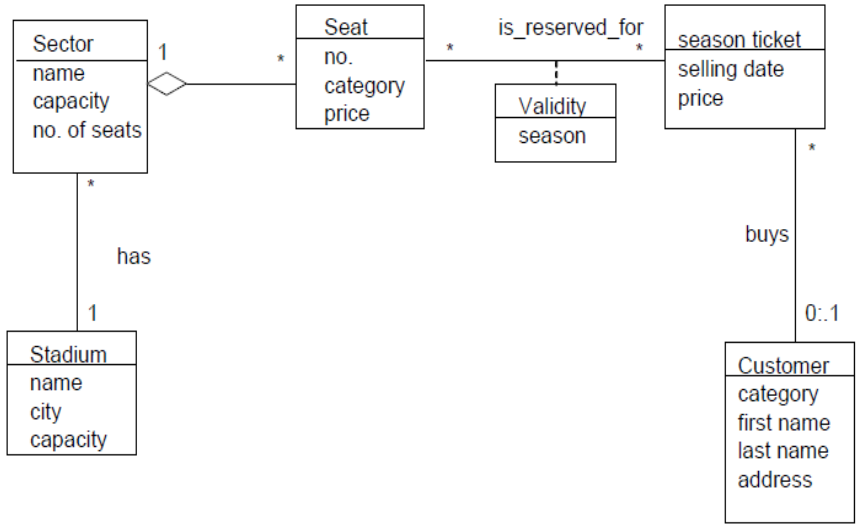
- Naming conflicts
  - homonyms: one word, different meaning
  - synonyms: different words, same meaning
  - Similarity: different names, same neighbours/constraints
  - Mismatch: same name, different neighbours /constraints
- Structural conflicts
  - Identical concepts: same structure and neighbours
  - Compatible concepts: different structure /neighbours, but no contradiction
  - Incompatible concepts: structural contradiction

Exercise 1 – Identify Conflicts 1

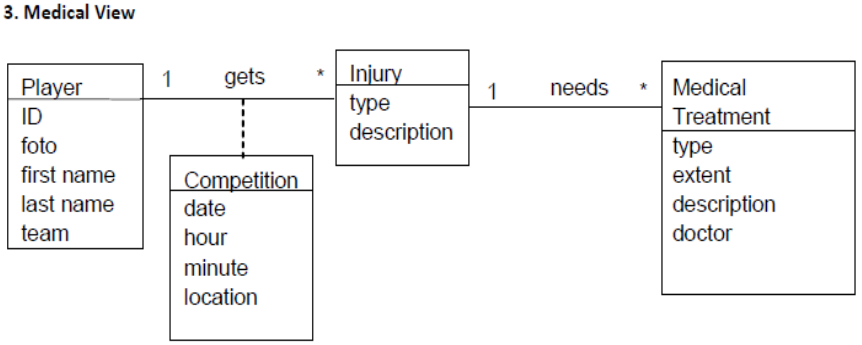
3. Game View



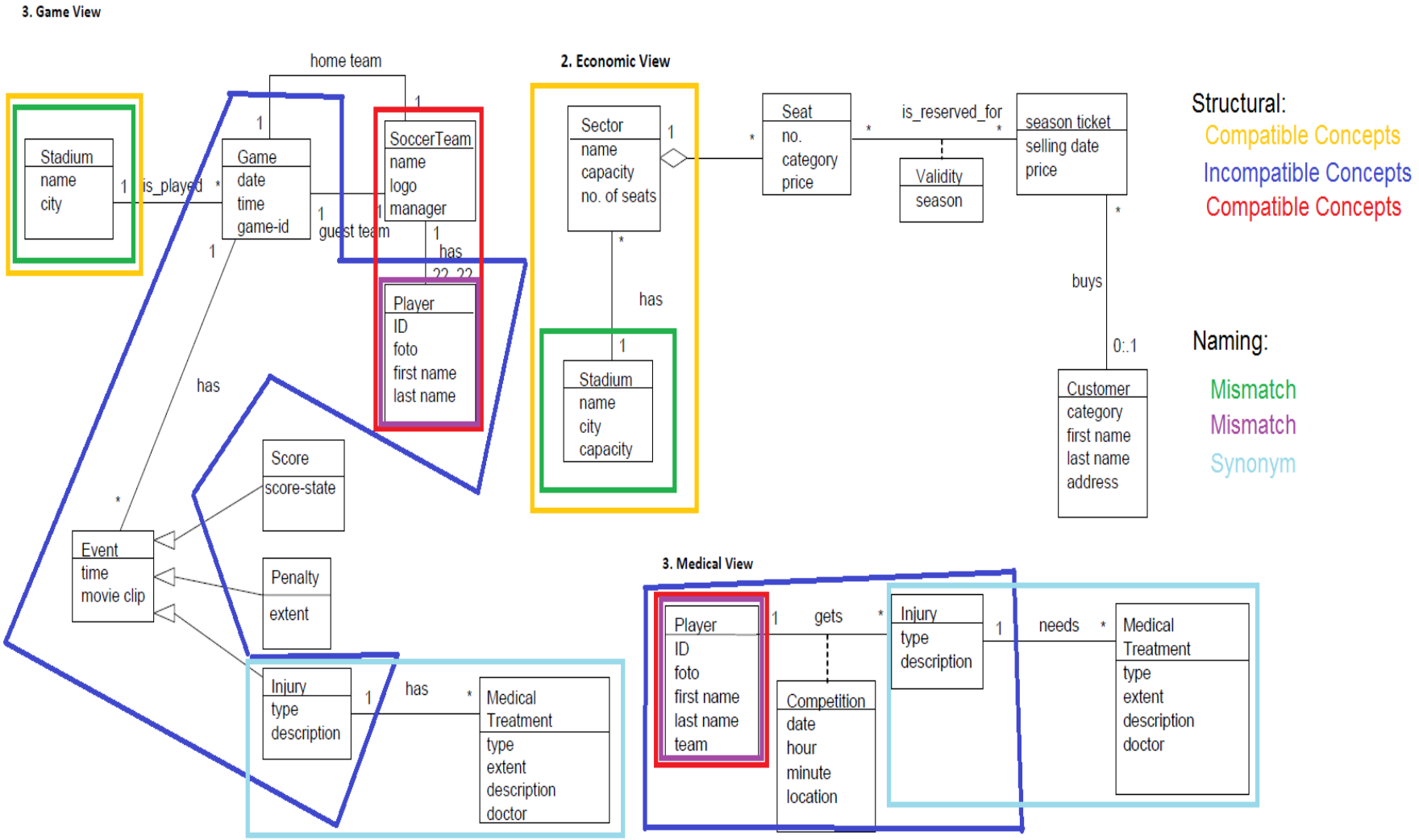
2. Economic View



3. Medical View



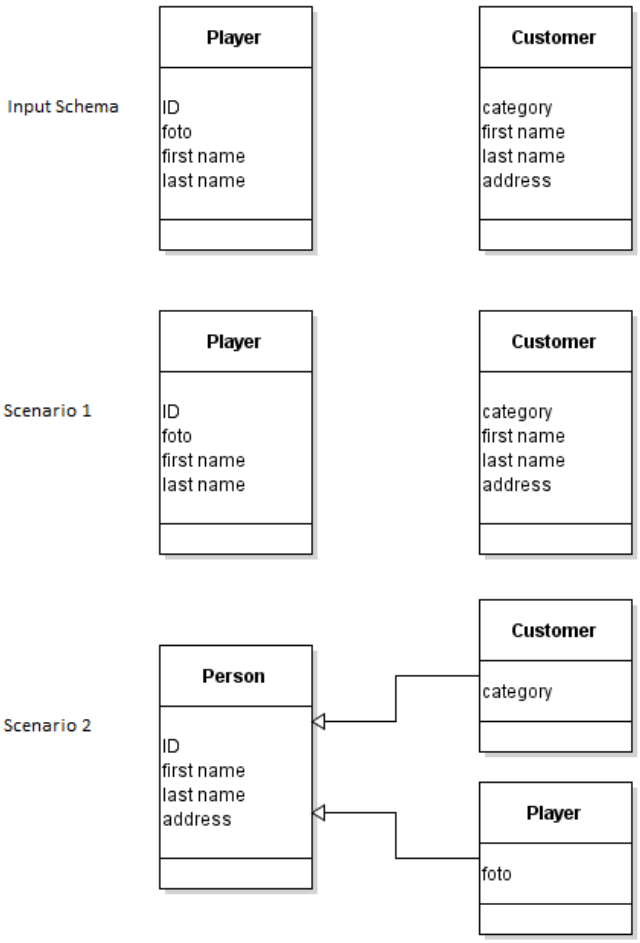
Exercise 1 – Identify Conflicts 2

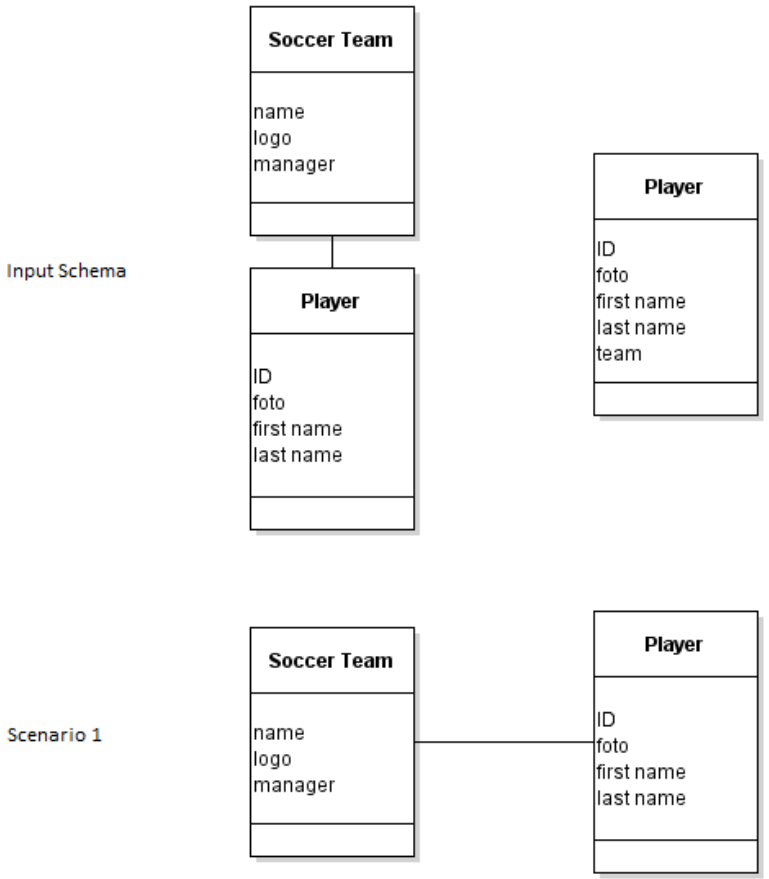


Example 1 – Identify Conflicts 3

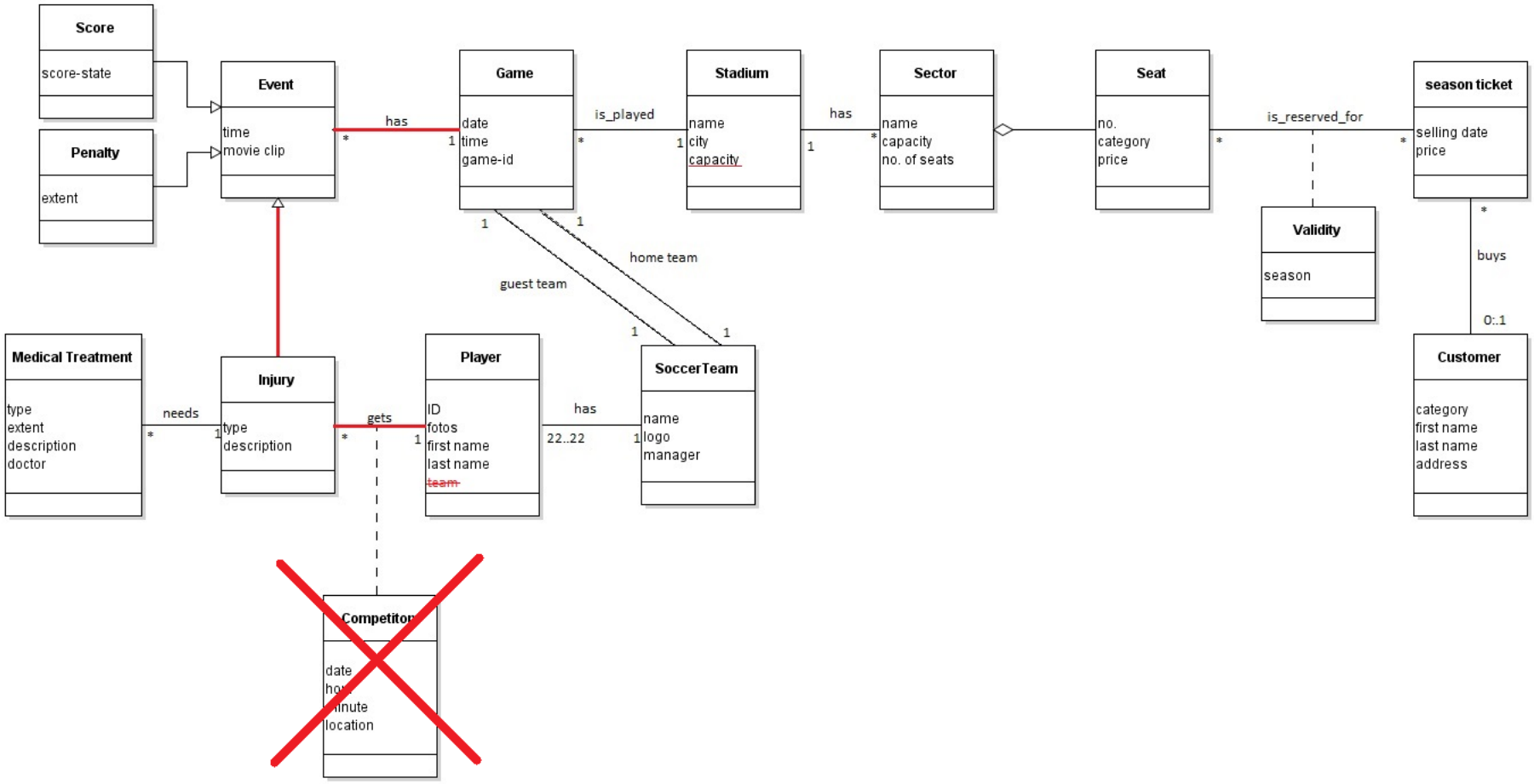
- Stadium – Stadium: (Mismatch, Compatible Concepts)
- Injury & MT – Injury & MT: (Synonym, Identical Concepts)
- Game, Event, Injury, Player – Injury, Player, Competition: (Incompatible Concepts)
- Player, Soccer Team – Player: (Mismatch, Compatible Concepts)

scenarios and interschema properties

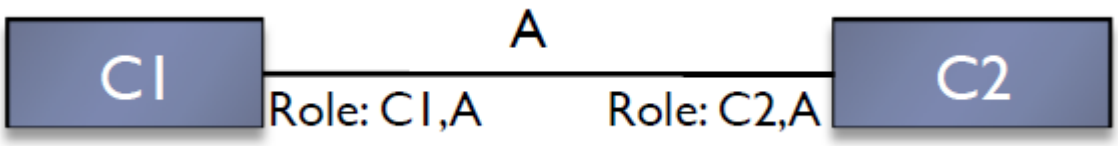




integrate the schemas



Exercise 2 – Determination of Quantities

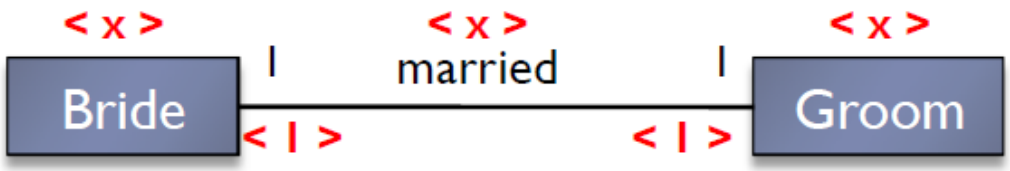


N(C) = average number of instances per class
N(A) = average number of association instances per association
N(C,A) = average number of class instances per association

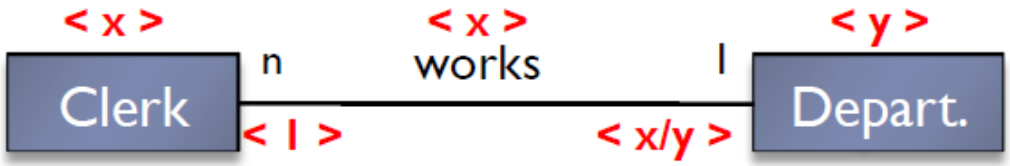
N(C1) x N(C1,A) = N(A) = N(C2) x N(C2,A)

Exercise 2 – Calculation rules

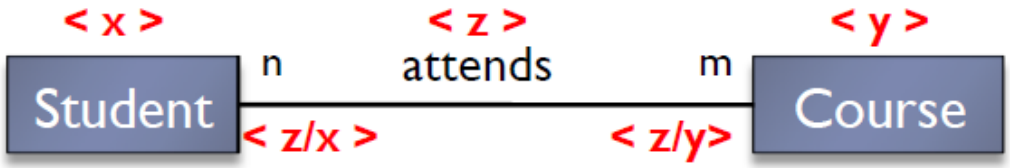
- 1:1 association



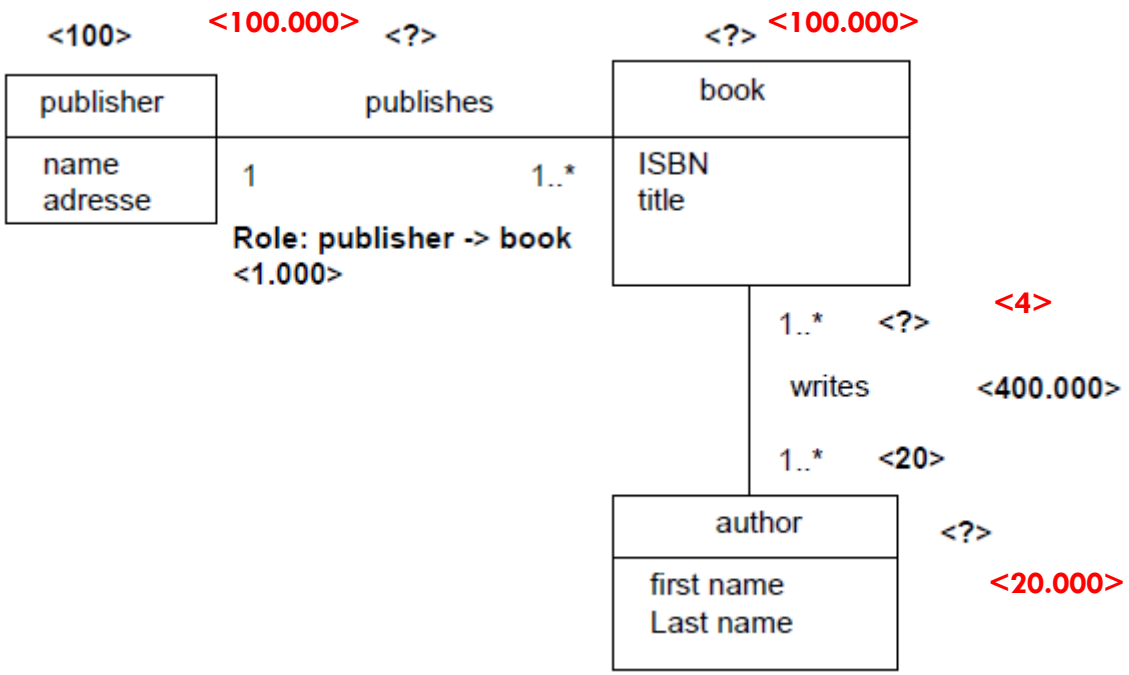
- 1:n association



- n:m association



Exercise 2 – Determination of Quantities



N(publisher) x N (publisher, publishes) = N (publishes) = N(book)
⇒ N(book) = N(publishes) = 100 x 1.000 = 100.000

N(author) = N(writes) / N(writes, author)
⇒ N(author) = 400.000 / 20 = 20.000

N(book, writes) = N (writes) / N(books)
⇒ N(book, writes) = 400.000 / 100.000 = 4

Exercise 2 - Navigation

- To get the result of a query or to execute an operation, multiple entity types and relations might have to be accessed.
- Navigation path shows, which entity types and relations are accessed and in which order.
- Necessary for evaluating costs of database operations

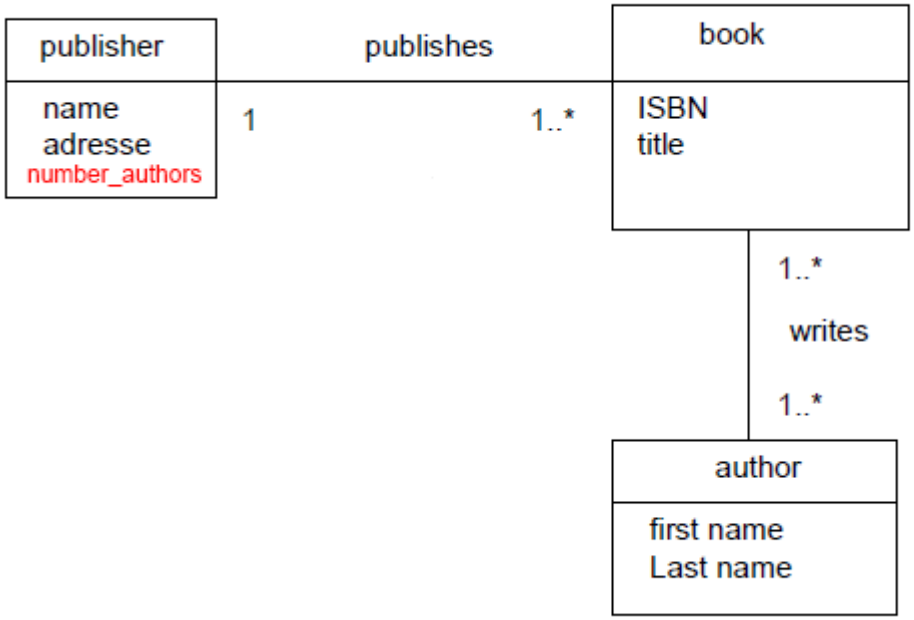
Exercise 3 – Redundancy

- Redundancy occurs, if the same information or derivable information is stored multiple times.

- Pros:
  - Speeds up data access (navigation path is shortened for some operations)

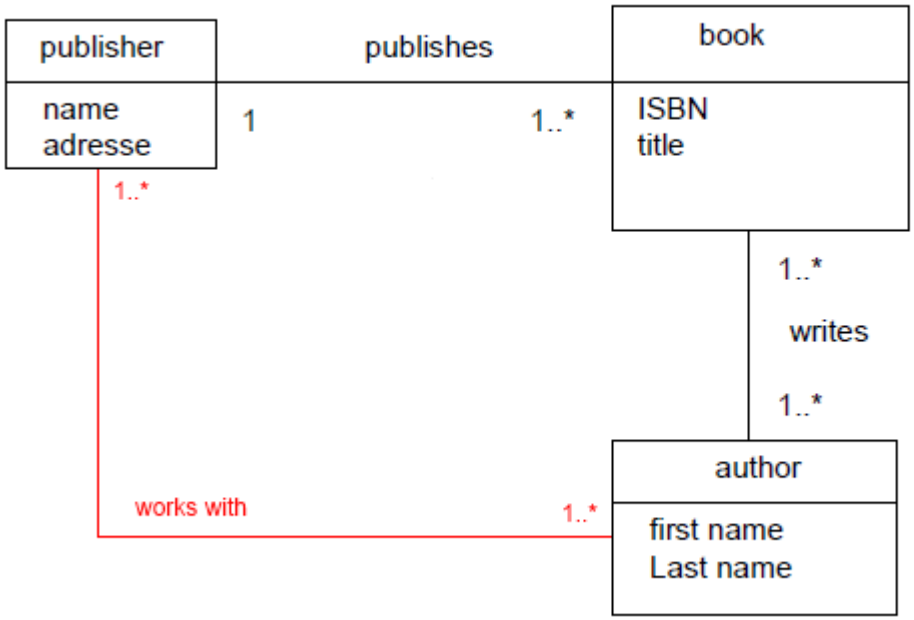
- Cons:
  - Additional updates
  - Risk of inconsistency (=> more consistency checks needed)
  - Additional storage space

Redundant Attribute



READ: Saves access to *book* and *writes*
WRITE: Every insert and delete and some updates in *book* and *writes* forces update of number_authors in *publisher*

Redundant Association



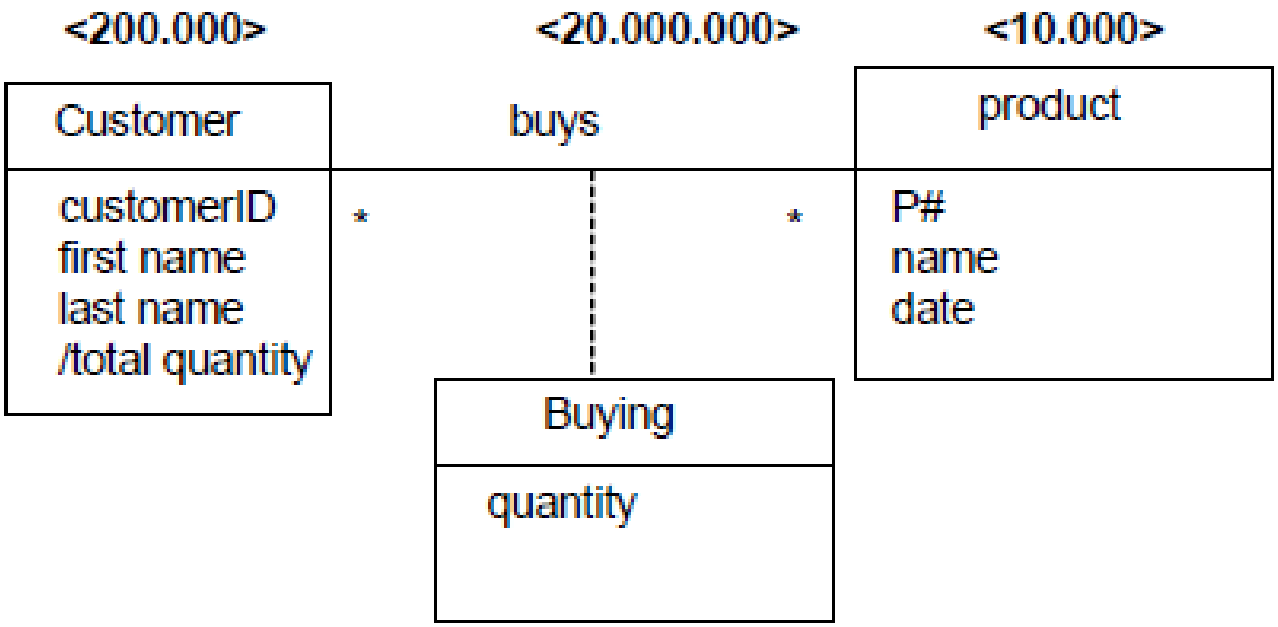
READ: *book* class is not needed in navigation path (w.r.t publisher -> author)
WRITE: Insert/Update/Delete operations on *book* and *writes* might trigger Insert and Delete operations on *works_with*

Redundancy – How to Decide

- Look at the operations influenced by the redundancy
- Calculate number of accesses for these operations
  - with redundancy
  - without redundancy
- If #Accesses with redundancy < #Accesses without redundancy => Keep redundancy
- Else: Remove redundancy

Exercise 3b

Check, if we should keep the redundant attribute `total_quantity`



Exercise 3b - Operations

Operation	Description	Frequency
T1	Insert a new customer	50 times / month
T2	Insert a new product	5 times /month
T3	Insert of a „Buying“	50 times / day (= 1000 times / month) *
T4	Listing of total quantitiy for each customer	2 times /month

*) **Note: A month has 20 working days.**

Next: calculate accesses per operation with and without redundancy, where …

… read accesses have weight 1
… write accesses have weight 2
… update accesses have weight 3

Exercise 3b – Data Access Table

Operation	Accesses with redundancy	Accesses without
T1	100 customer (write 50 * 2) = 100 accesses	100 customer (write 50 * 2) = 100 accesses
T2	10 product (write 5 * 2) = 10 accesses	10 product (write 5 * 2) = 10 accesses
T3	2.000 buys (write 1.000 * 2) + 3.000 customer (update 1.000 * 3) = 5.000 accesses	2.000 buys (write 1.000 * 2) = 2.000 accesses
T4	400.000 customer (200.000 read twice) = 400.000 accesses	400.000 customer (read 200.000 twice) + 40.000.000 buys (read 20.000.000 twice) = 40.400.000 accesses
Sum	= 405.110 accesses per month	= 40.402.110 accesses per month

Exercise 3b – Conclusion

- Accesses per month without redundancy:

  40.402.110

- Accesses per month with redundancy:

  405.110

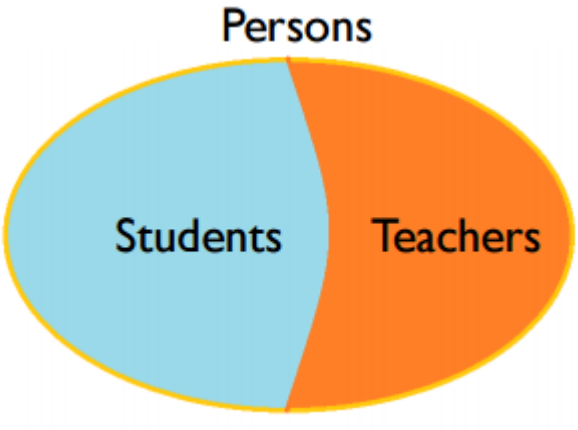
- **Therefore: Keep redundancy!**

Exercise 4

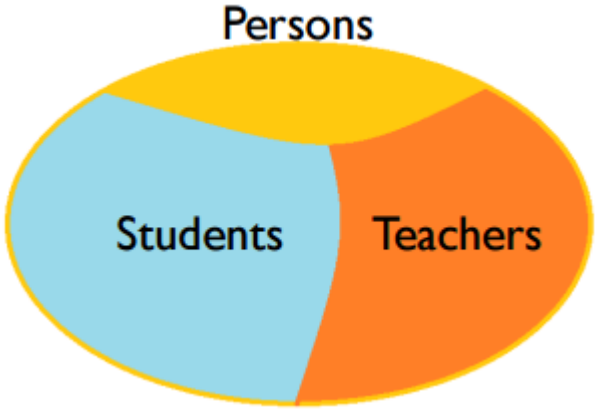
- Explain the notions total, partial, exclusive, and overlapping in the context of generalization.
- What kind of flattening strategies exist for generalization hierarchies? Explain them with an example of your own.
- How can you determine, which flattening strategy should be used? What data do you need for the decision?

total vs partial

- total
  - every instance of a super-class is also an instance of a (direct) sub-class children

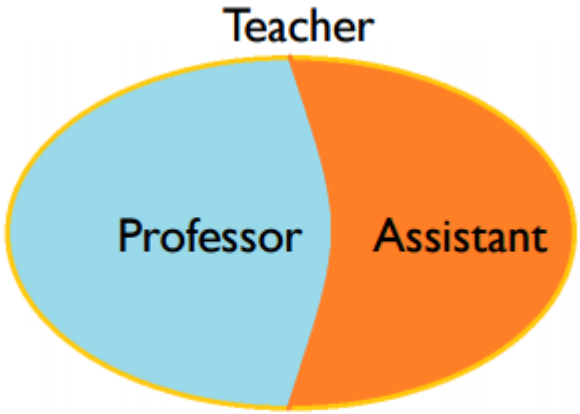


- partial
  - otherwise

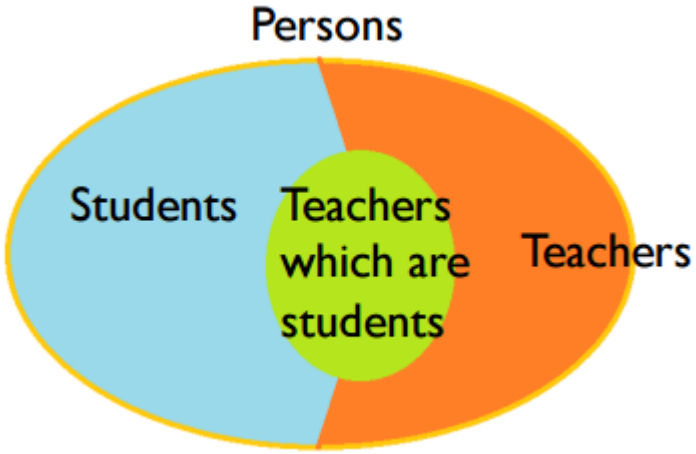


exclusive vs overlapping

- exclusive
  - no instance of the super-class belongs to more than one sub-class

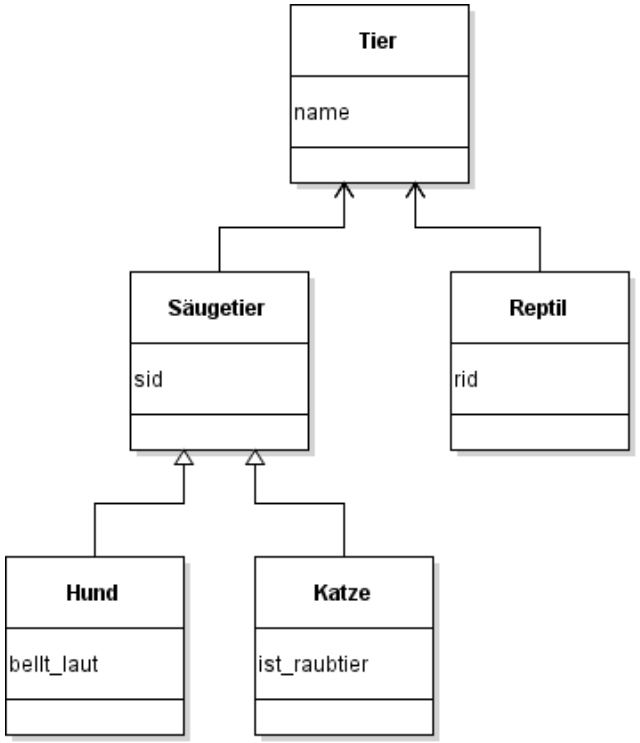


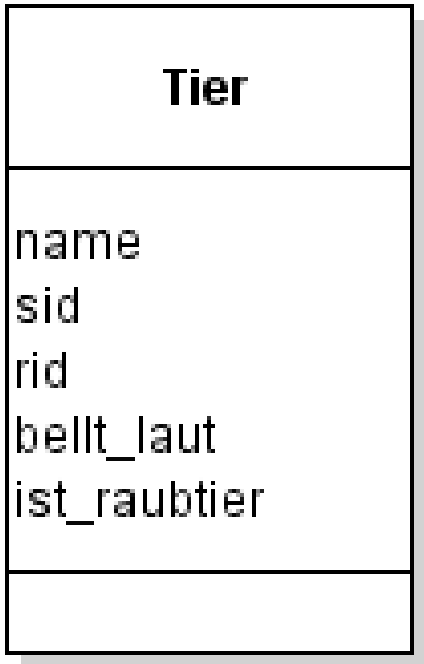
- overlapping
  - otherwise



Flattening(1 /3)

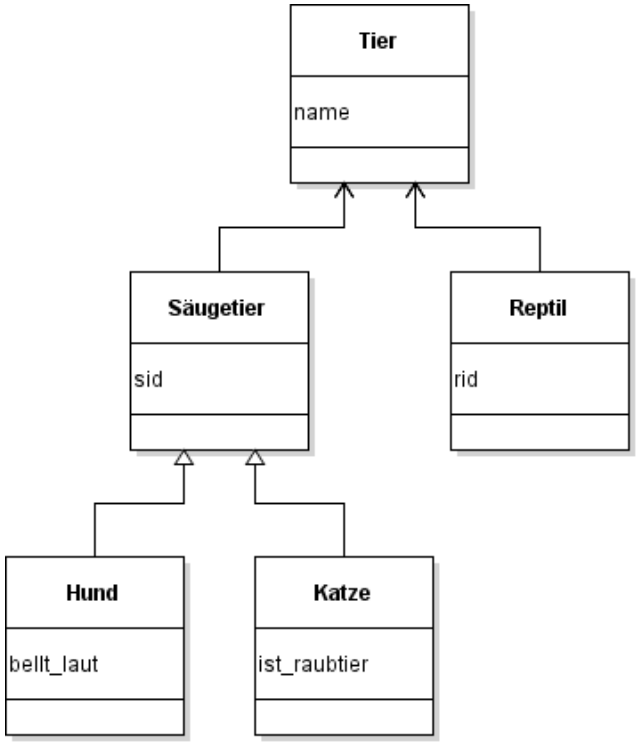
- Ceiling
  - nur Superclasse
  - hat alle Attrubute von Subclassen

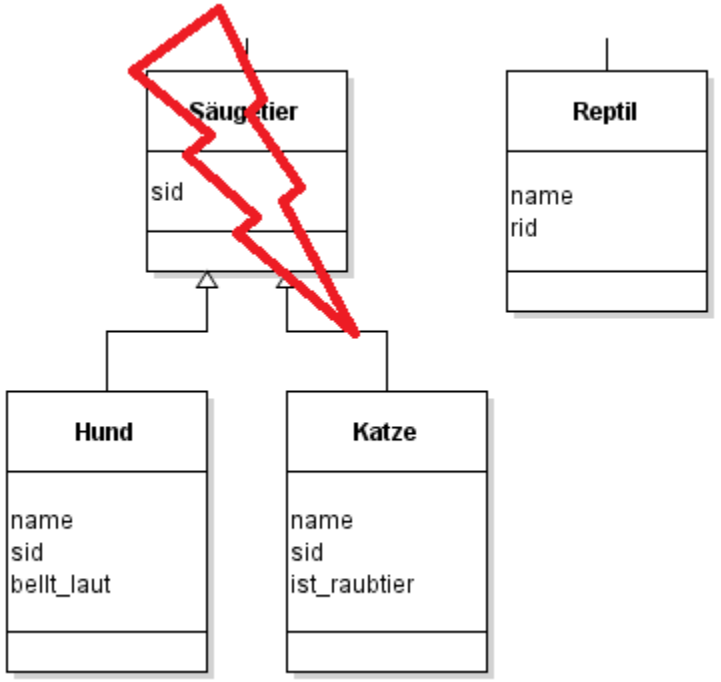




Flattening(2/3)

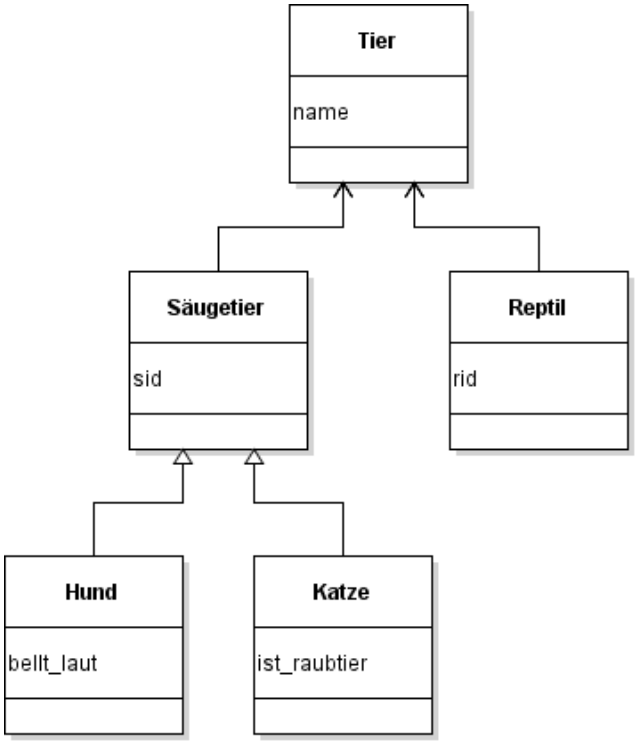
- Floor
  - nur Klassen die Blätter im generalization hirarchy tree sind

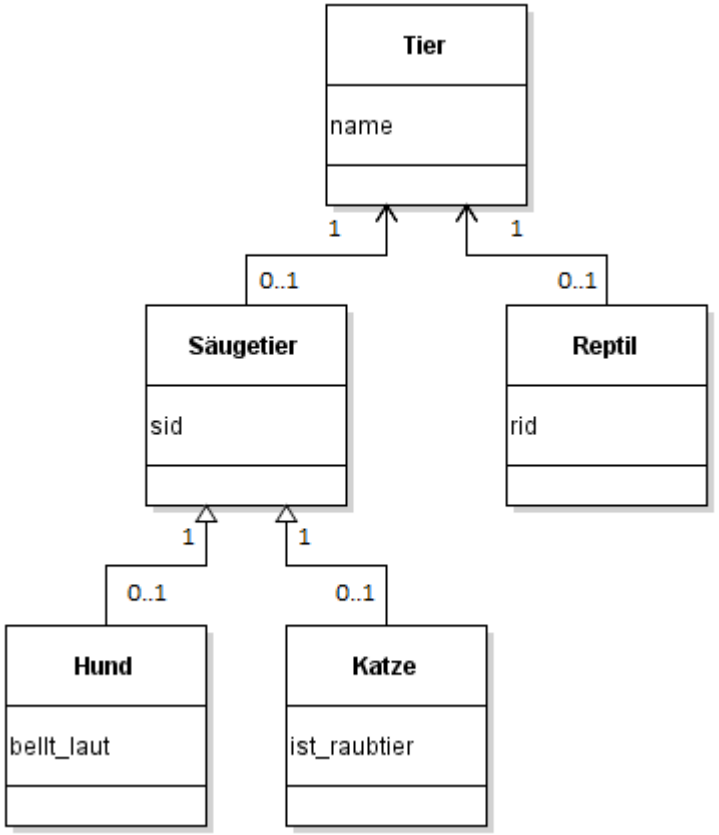




Flattening(3/3)

- Cohesion
  - each generalization is represented as 1:1 relation





Which Flattening Strategy?

1. Two operation sets
   - S1 set of operations, which access attributes of the super-class
   - S2 set of operations, which access attributes of super-class and one sub-class
2. Calculate access counts for S1 and S2
3. Decide

if S2 dominates → floor

else analyse data manipulation operations

if domination of operations that ..

..access attributes of both super- and sub-class → ceiling

..access attributes of either super- or sub-class → cohesion

Exercise 5

- What is vertical partitioning, what is horizontal partitioning? Explain it with an example of your own.
- Why might partitioning be necessary?
- What do you have to consider before partitioning – how do you decide?

Vertical partitioning(1/2)

- split class vertically
- new classes have different set of attributes
- needs join operation to retrieve original set of instances

Horizontal partitioning

- split class horizontally
- new classes have same set of attributes
- multiplication of original associations required
- needs union operation to retrieve original set of instances

vertical / horizontal example

vertical

horizontal

ID	Name	Age	Address
1	Lukas	23	Street 21
2	Dominic	24	Address 54
3	Bob	25	Str 19
4	Alice	26	Way 46
5	Eve	27	Abc 39

ID	Name	Age	Address
1	Lukas	23	Street 21
2	Dominic	24	Address 54
3	Bob	25	Str 19

ID	Name	Age	Address
4	Alice	26	Way 46
5	Eve	27	Abc 39

ID	Name
1	Lukas
2	Dominic
3	Bob
4	Alice
5	Eve

ID	Age	Address
1	23	Street 21
2	24	Address 54
3	25	Str 19
4	26	Way 46
5	27	Abc 39

why partitioning?

- horizontal partitioning
  - many operations on different sets of instances
- vertical partitioning
  - many operations on different sets of attributes
- split huge attributes (BLOBs)
- security aspects
- reduce networking traffic in distributed databases

consider before partitioning

- which sets of instances are in use?
- which sets of attributes are in use?

  - change programmaticaly access to db