

Databases

Lecture 13

Conceptual Modeling

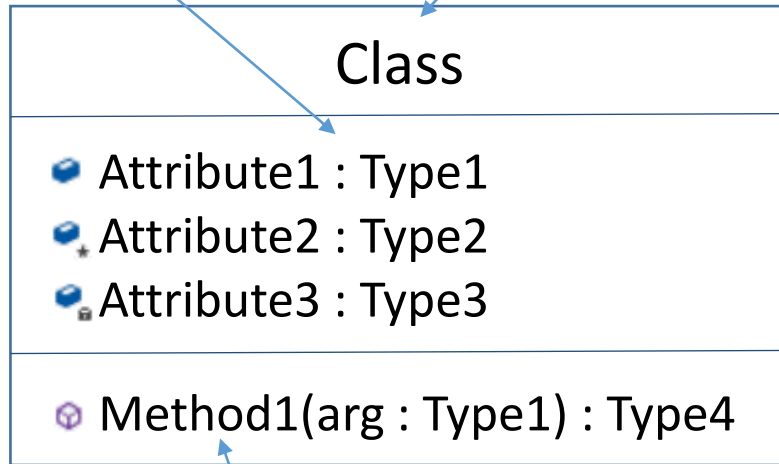
- database design - stages
 - requirements analysis
 - what data will the DB store?
 - what are the main operations to be supported?
 - what apps will be powered by the DB?
 - conceptual DB design
 - high level description of data and integrity constraints
 - logical DB design
 - translate the conceptual DB design to a DB schema in terms of the model supported by the DBMS (e.g., relational)
 - schema refinement
 - normalization
 - eliminate redundancy and associated problems

- database design - stages
 - physical DB design
 - create indexes
 - redesign parts of the schema

- UML class diagram
 - classes

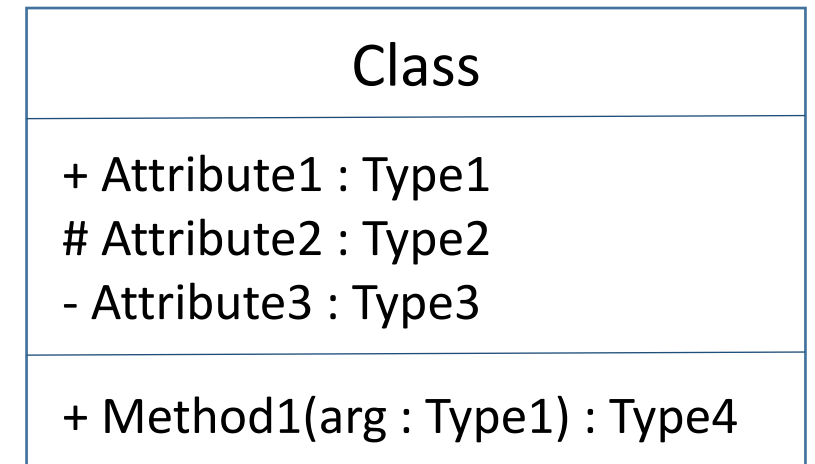
attributes

name

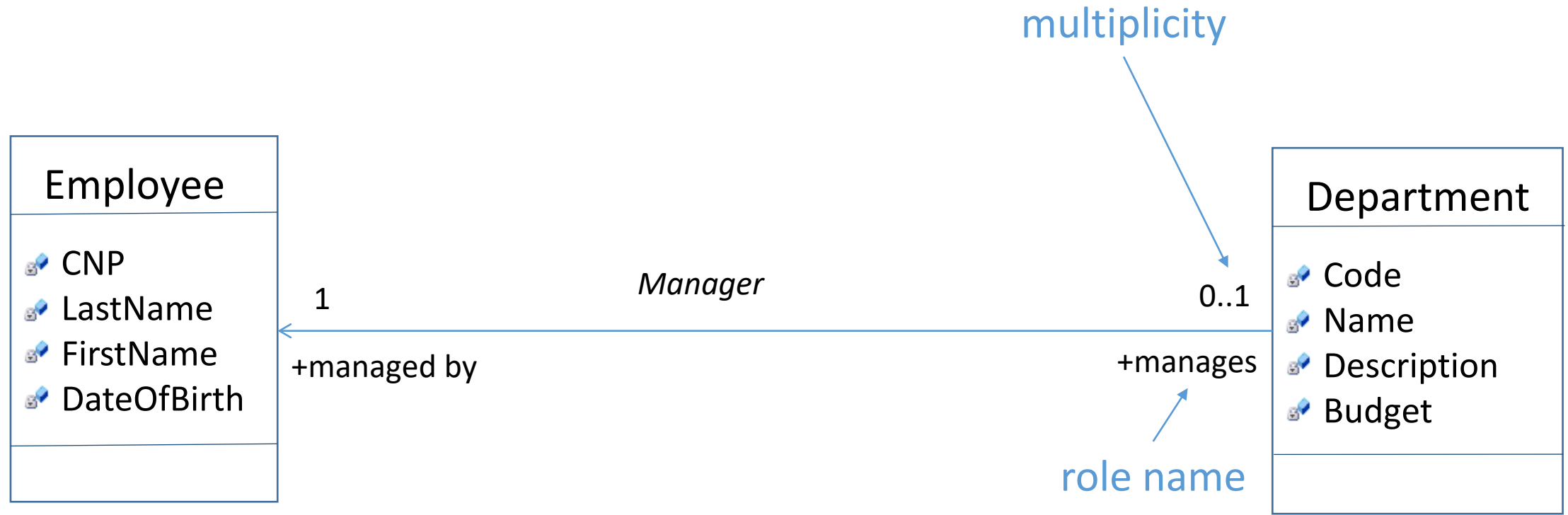


methods

public
protected
private

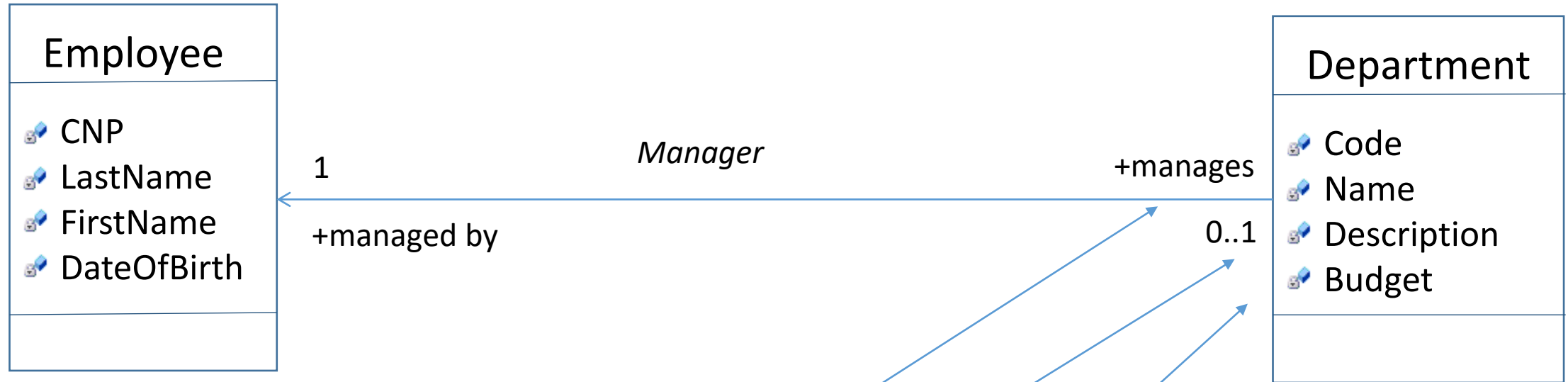


- UML class diagram
 - associations



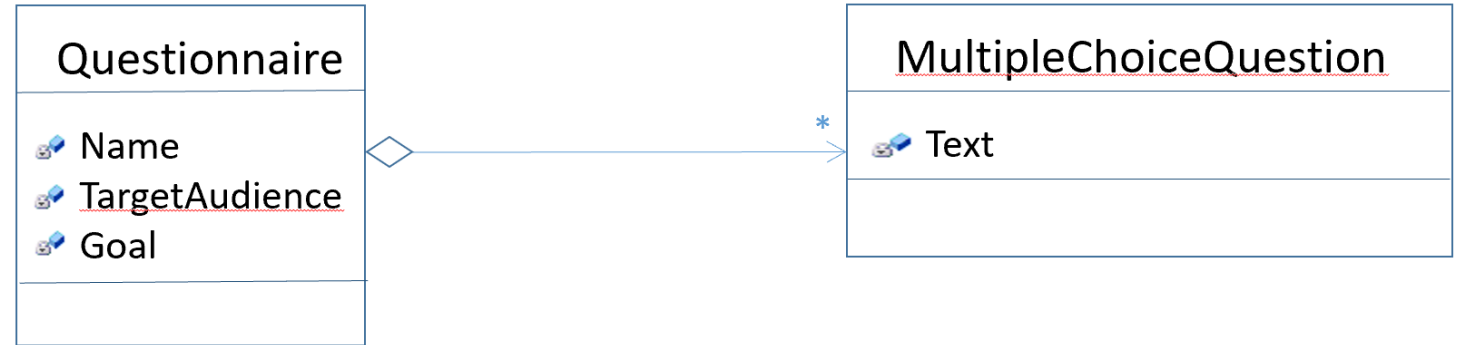
- navigability – unidirectional, bidirectional
- multiplicity – examples
 - 0..1
 - 5
 - 0..*
 - 7..10

- UML class diagram
 - associations

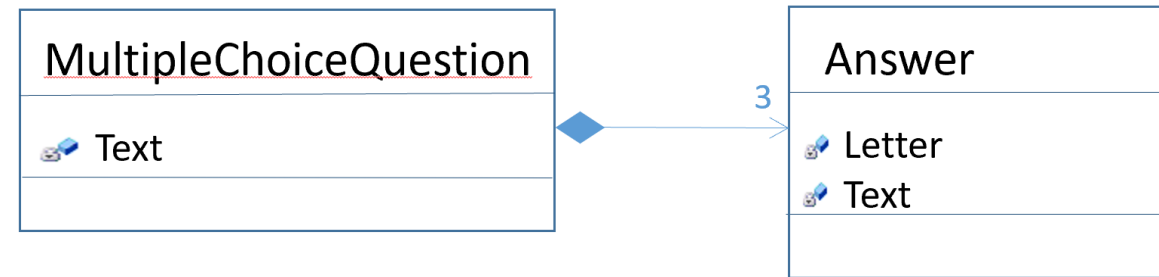


An employee manages 0 or 1 departments.

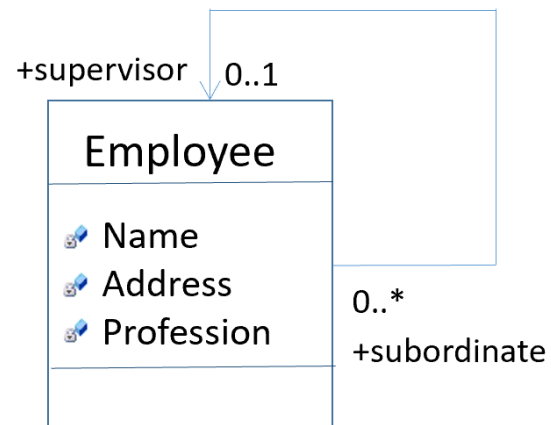
- UML class diagram
 - aggregation



- composition

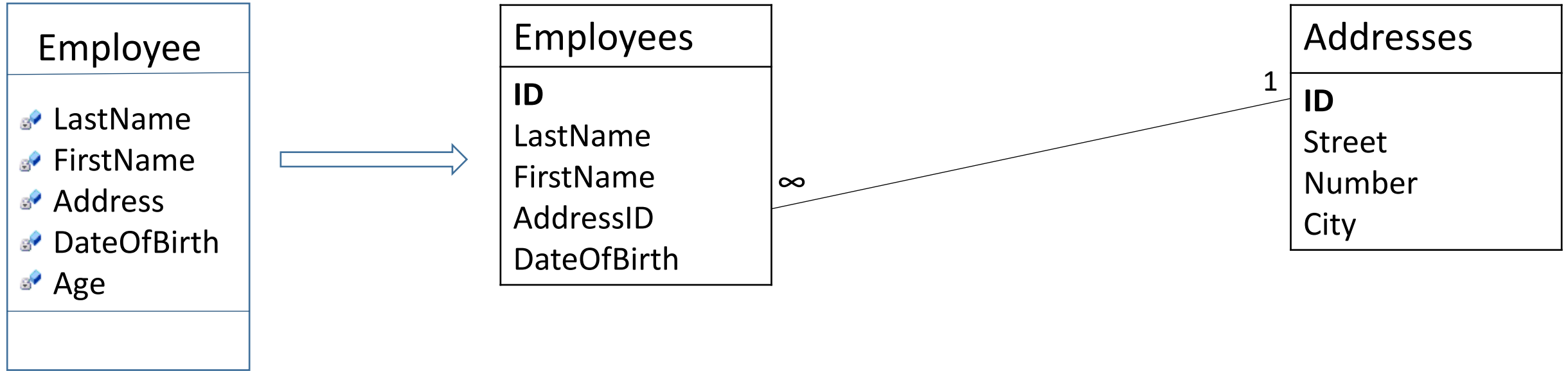


- reflexive association



- conceptual model => relational database
- 1:1 mapping, i.e., classes become tables
- drawbacks
 - one could create too many tables
 - too many tables => too many join operations
 - necessary tables could be omitted; m:n associations require a third table (join table)
 - inheritance is not properly handled

- class -> table



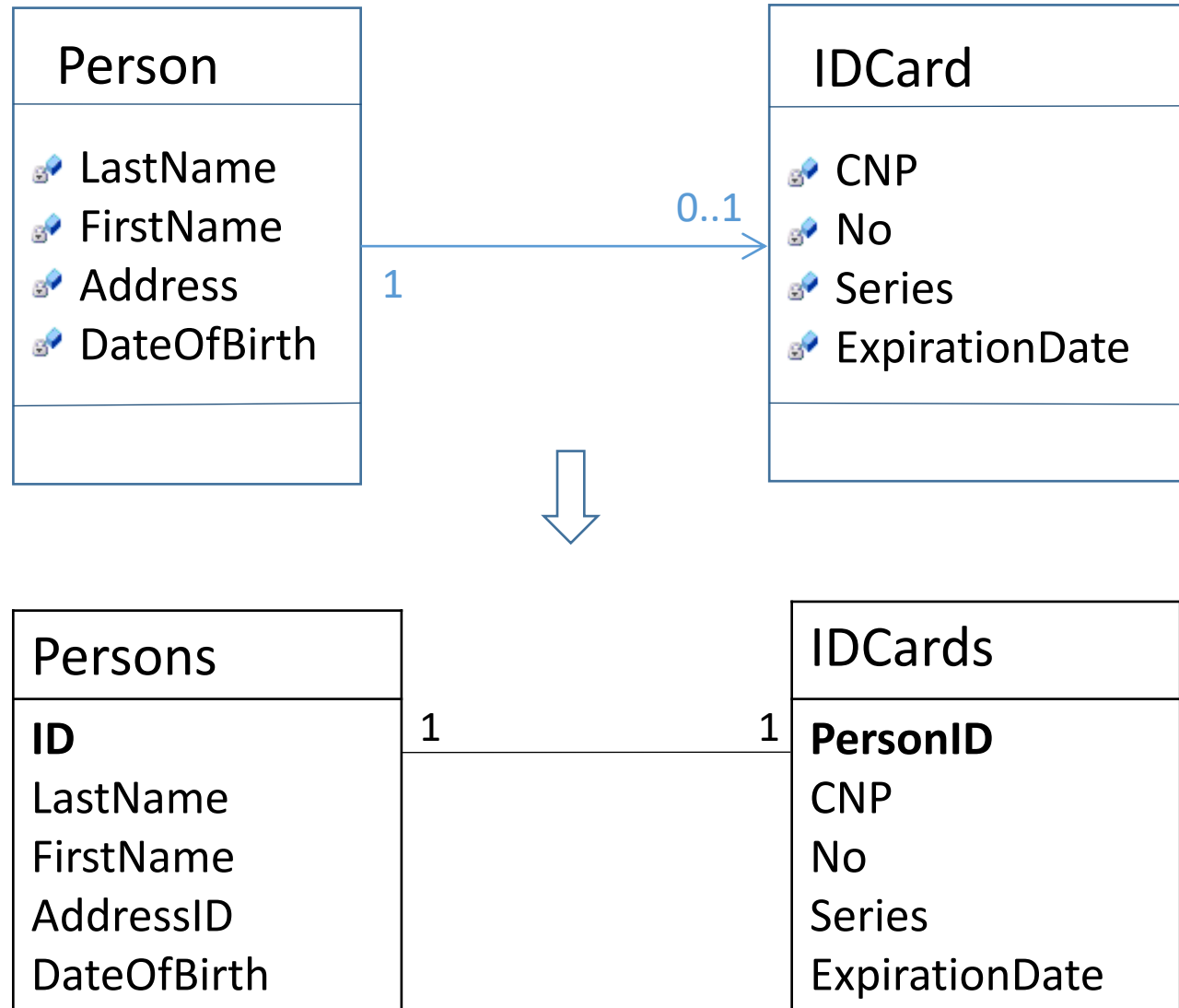
- the plural of the class name becomes the name of the table
- simple class attributes become table fields
- composite attributes become tables
- derived attributes are not mapped to table fields
- surrogate keys are added

- class -> table
 - surrogate key
 - key that isn't obtained from the domain of the modeled problem
 - when possible, use integer keys that are automatically generated by the DBMS
 - easy to maintain - the responsibility of the system
 - efficient approach (fast queries)
 - simplified definition of foreign keys
- possible approach
 - surrogate key name: *ID*
 - foreign key name: *<SingularTableName>ID*

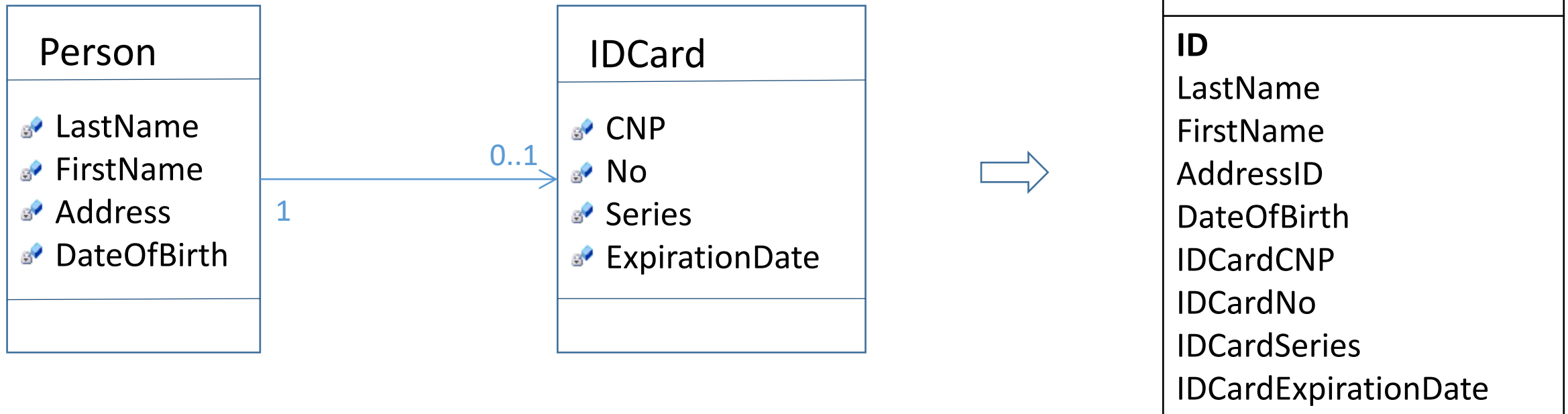
- mapping simple associations
- 1 : 0..1
 - create 1 table per class
 - the key of the 1 table (i.e., table at the 1 end of the association) becomes a foreign key in the 2nd table
 - usually, only one key is automatically generated (the one corresponding to the 1 table)

->

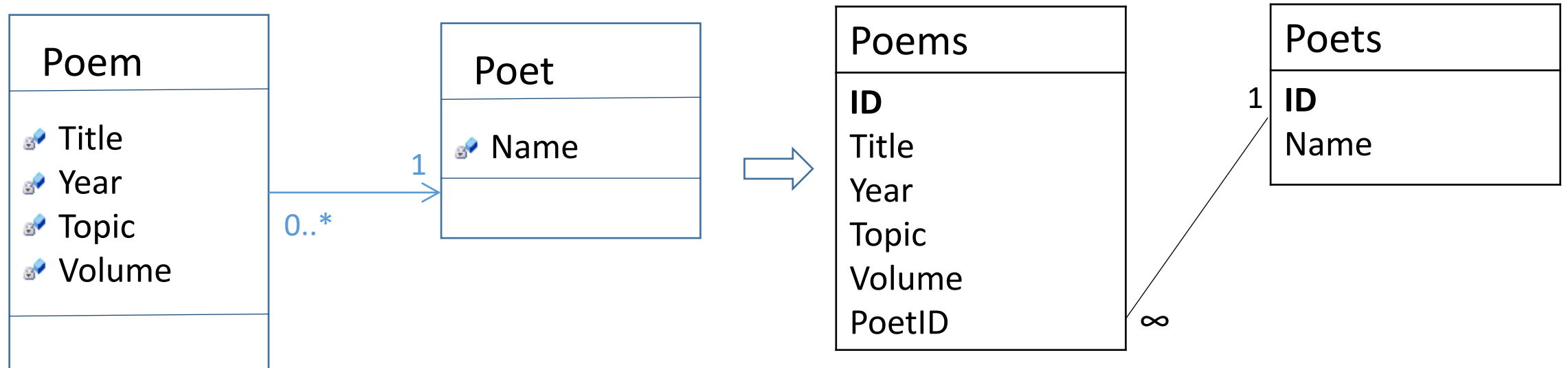
- mapping simple associations
- 1 : 0..1



- mapping simple associations
- 1 : 1
 - create 1 table containing the attributes of both classes
 - this approach can also be used for 1 : 0..1 associations (when only a few objects in the 1st class are not associated with objects in the 2nd class)

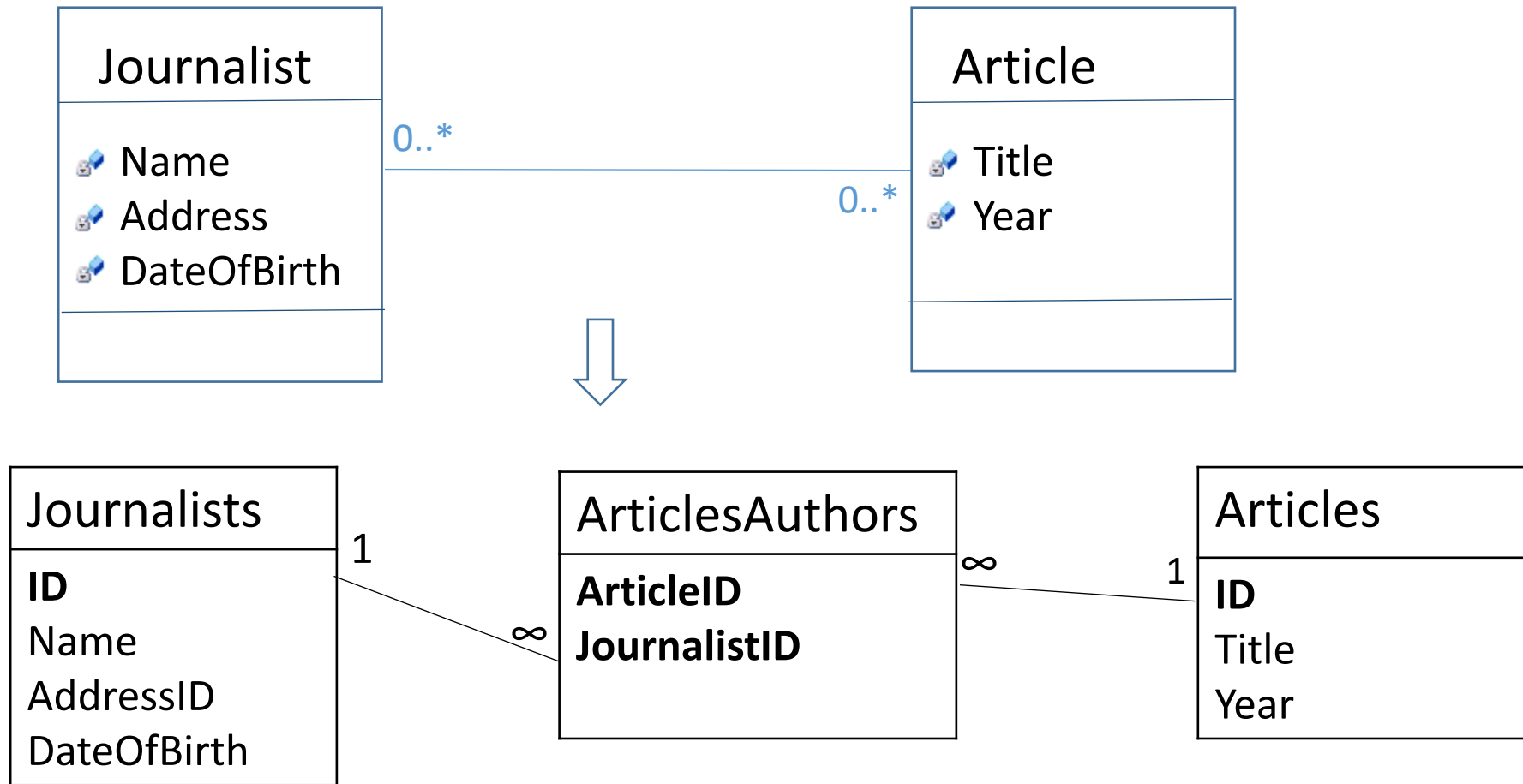


- mapping simple associations
- one-to-many
 - create 1 table / class
 - the key of the 1 table becomes a foreign key in the 2nd table

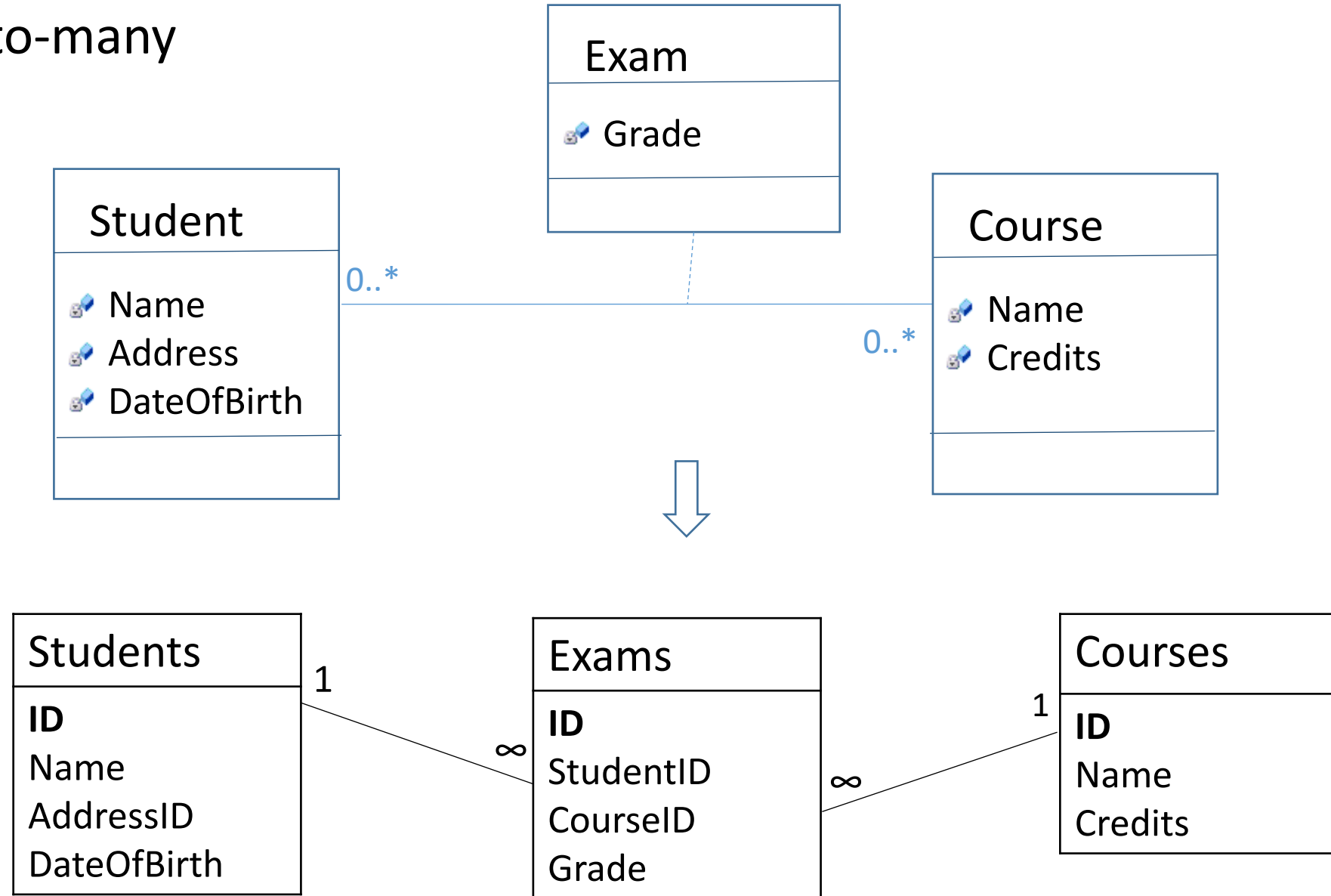


- mapping simple associations
- many-to-many
 - create one table / class
 - create an additional table, i.e., the *join table*
 - the primary keys of the 2 initial tables become foreign keys in the join table
 - the primary key of the join table:
 - composite, containing the 2 foreign keys
 - surrogate key
 - the name of the join table is usually a combination of the names of the 2 initial tables (not mandatory)
 - if an association class exists, its attributes become fields in the join table

- mapping simple associations
- many-to-many



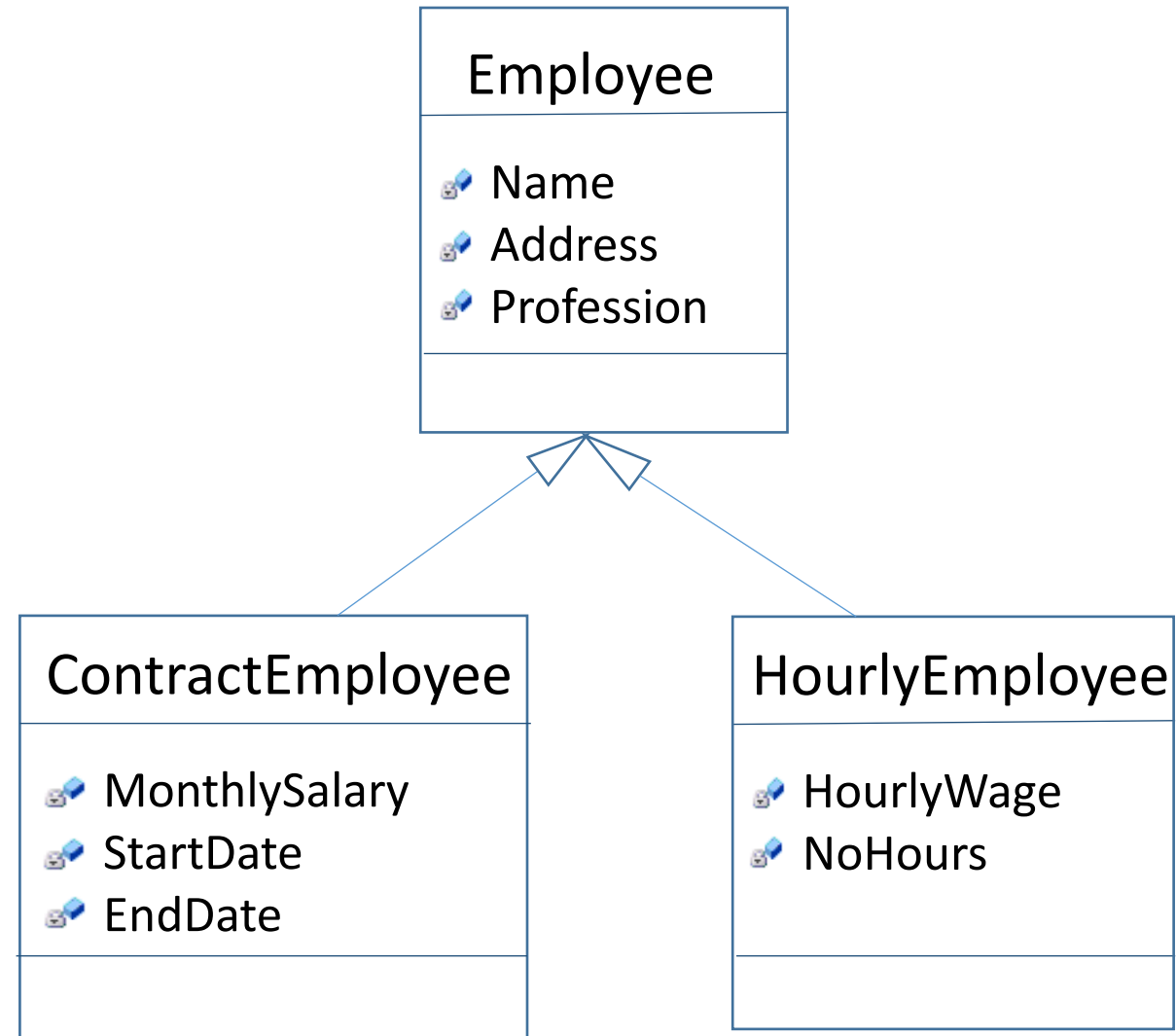
- mapping simple associations
- many-to-many



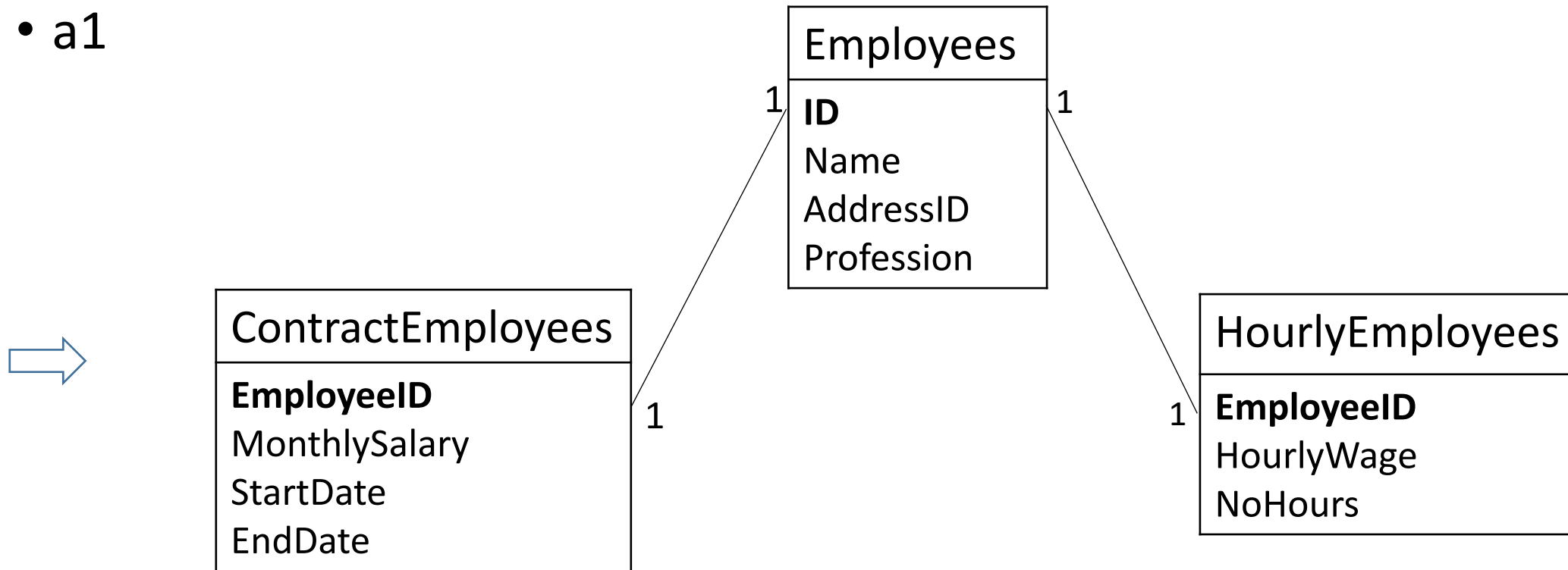
- mapping inheritance
- a1
 - create one table / class
 - create one view / superclass-subclass pair
 - it generates the largest number of objects (tables, views)
 - flexibility - no impact on existing tables / views when adding other subclasses
 - possible performance problems – every access requires a join through the view
 - can be used when the number of records is relatively small (so performance is not a concern)

->

- mapping inheritance
- a1



- mapping inheritance
- a1



```
CREATE VIEW ContractEmployeesComplete(...)
```

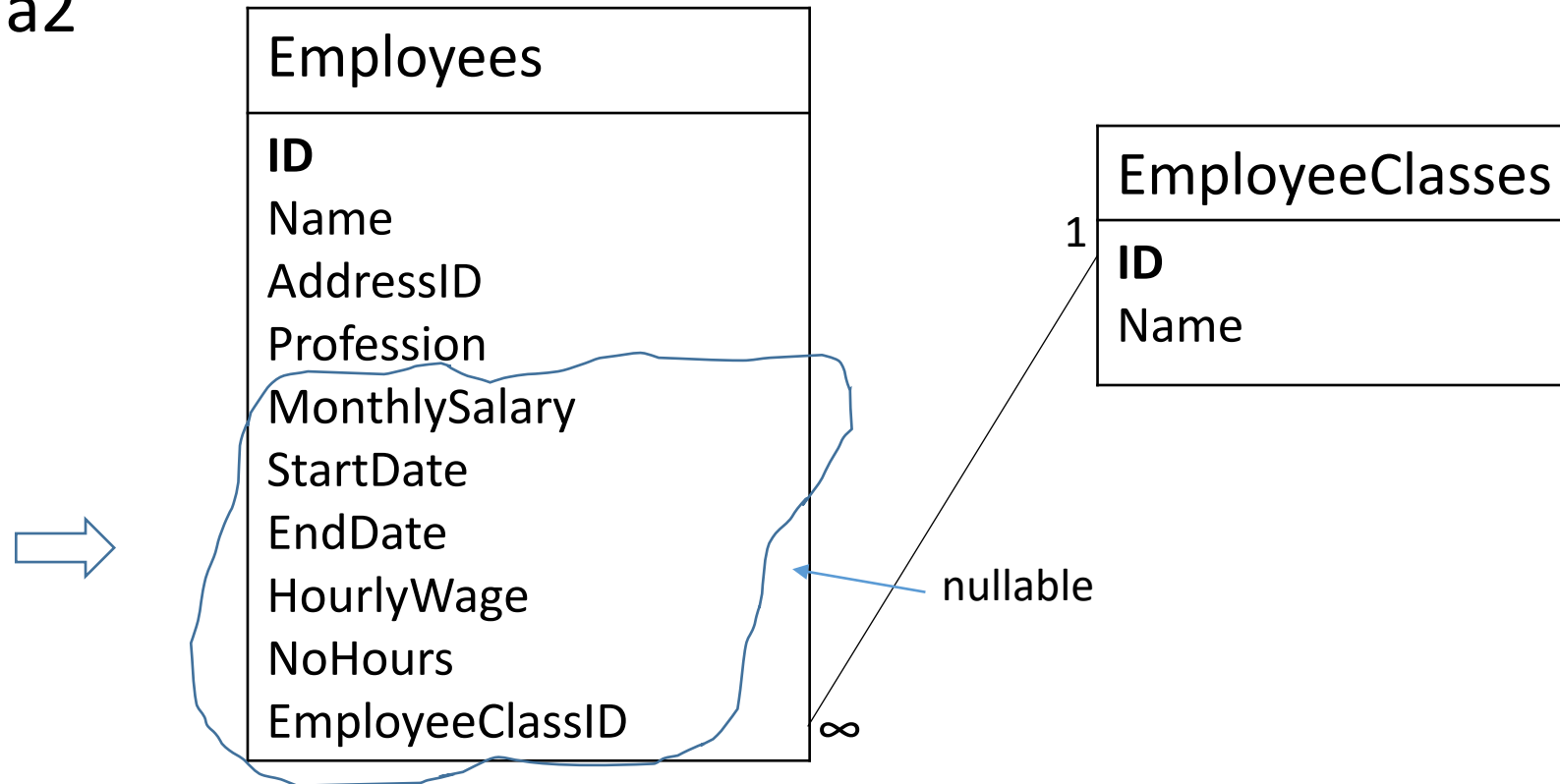
```
AS
```

```
SELECT Employees.*, MonthlySalary, StartDate, EndDate  
FROM Employees INNER JOIN ContractEmployees  
ON Employees.ID = EmployeeID
```

- mapping inheritance
- a2
 - create one table for the superclass
 - the attributes of the subclasses become fields in the table
 - it generates the smallest number of objects
 - optionally, a subclasses table and a view / subclass can be added
 - usually – best performance
 - when adding a subclass, the existing structure has to be changed
 - "artificial" increase of used space

->

- mapping inheritance
- a2



EmployeeClasses

ID	Name
1	Unknown
2	ContractEmployee
3	HourlyEmployee

```
CREATE VIEW ContractEmployees(...)
AS
  SELECT ID, Name, AddressID, Profession, MonthlySalary, StartDate,
         EndDate
  FROM Employees
 WHERE EmployeeClassID = 2
```

- mapping inheritance
- a3
 - create one table / subclass
 - the attributes of the superclass become fields in each of the created tables
 - satisfactory performance
 - subclasses can be subsequently added without affecting existing tables
 - changing the structure of the superclass impacts all existing tables

->

- mapping inheritance
- a3



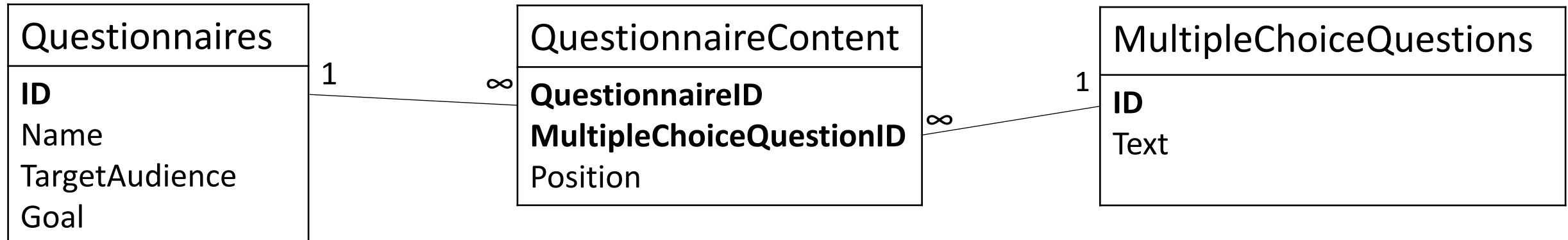
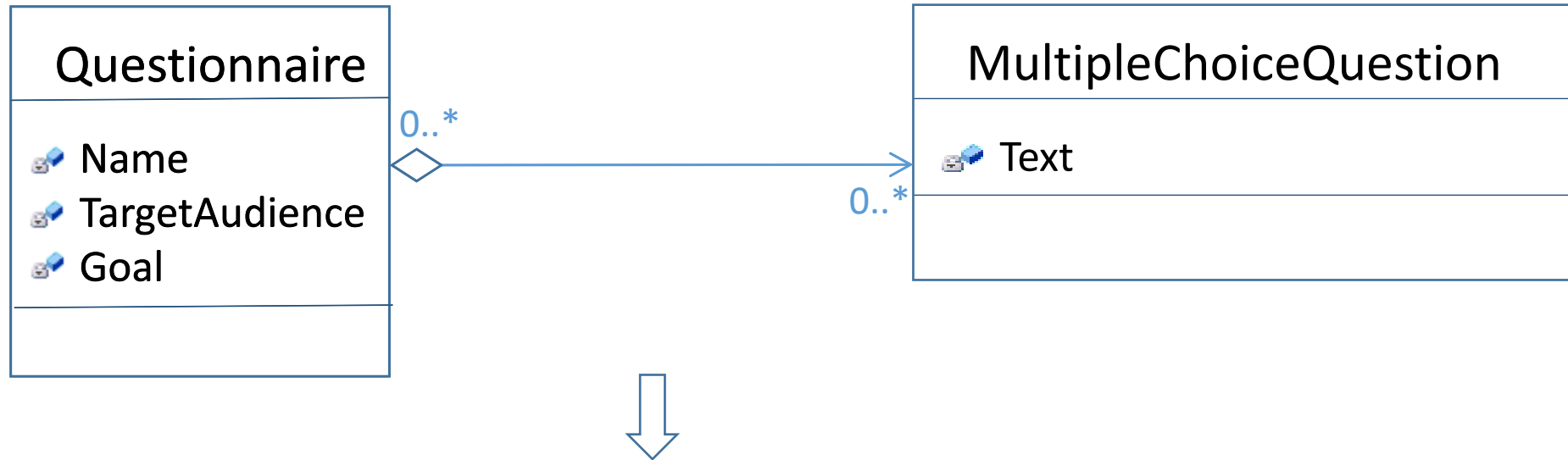
ContractEmployees
ID
Name
AddressID
Profession
MonthlySalary
StartDate
EndDate

HourlyEmployees
ID
Name
AddressID
Profession
HourlyWage
NoHours

- mapping aggregation / composition
 - similar to mapping simple associations
 - fixed number of *parts* in a *whole* => can declare the same number of foreign keys in the *whole* table
 - composition - ON DELETE CASCADE option (not required for aggregation)

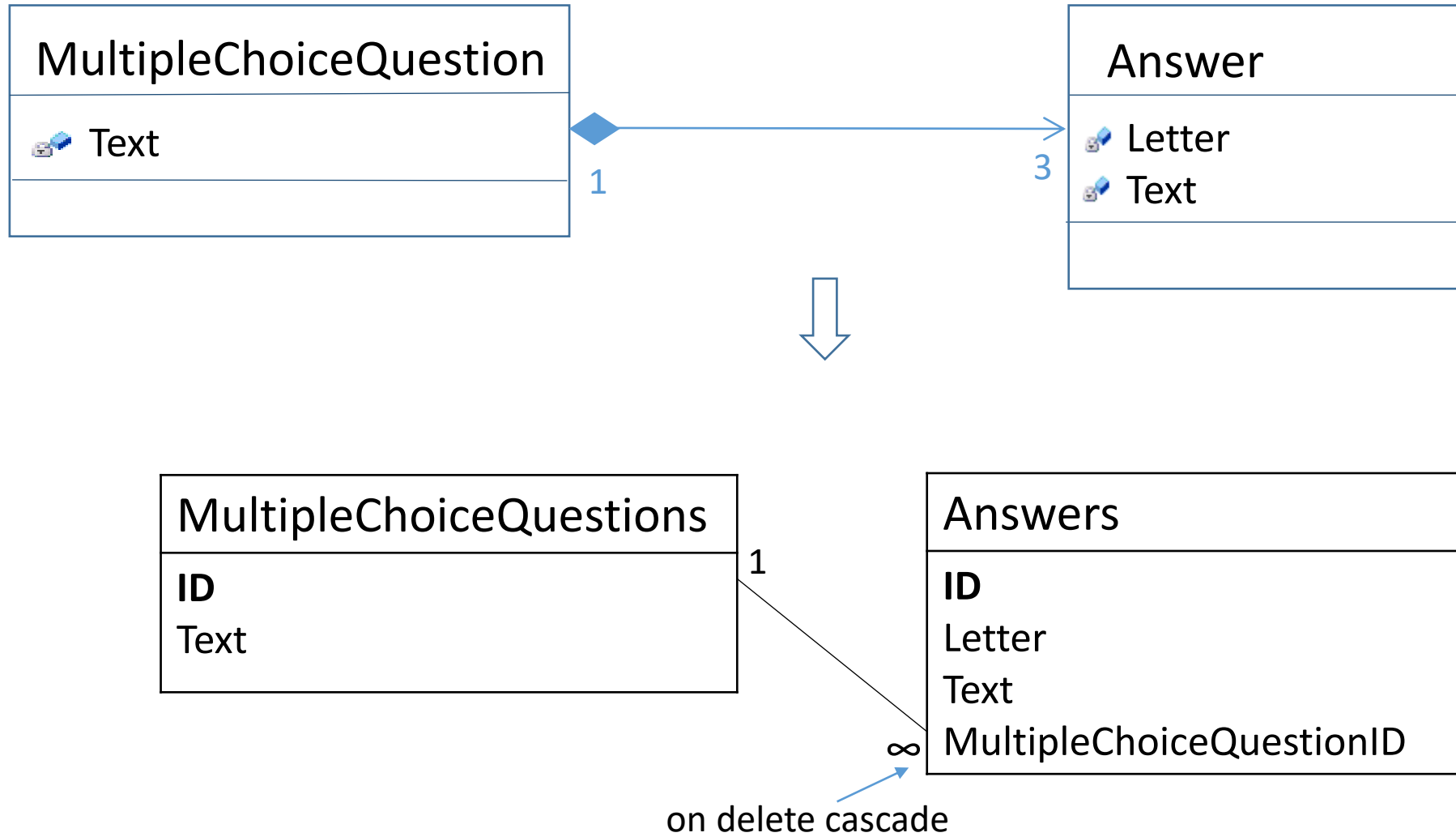
->

- mapping aggregation / composition

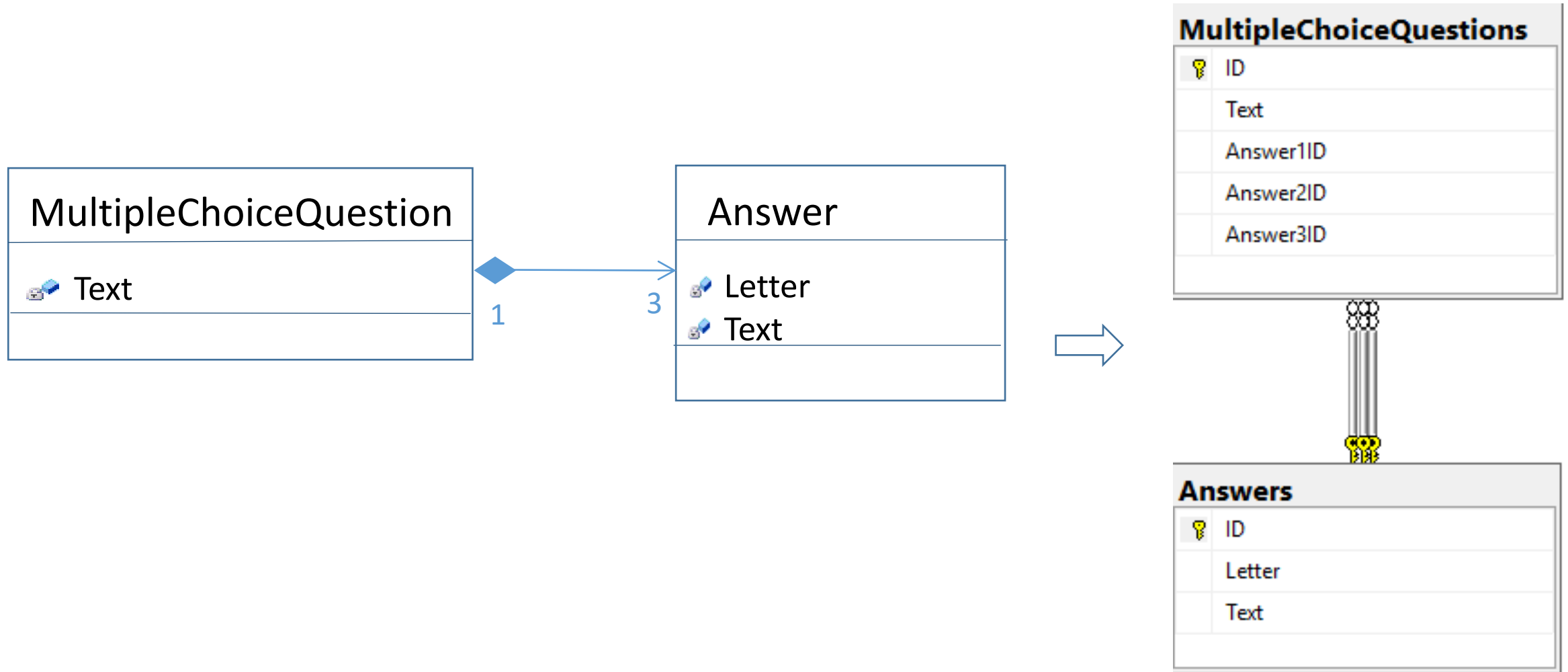


- obs. a questionnaire can also have open answer questions, etc.

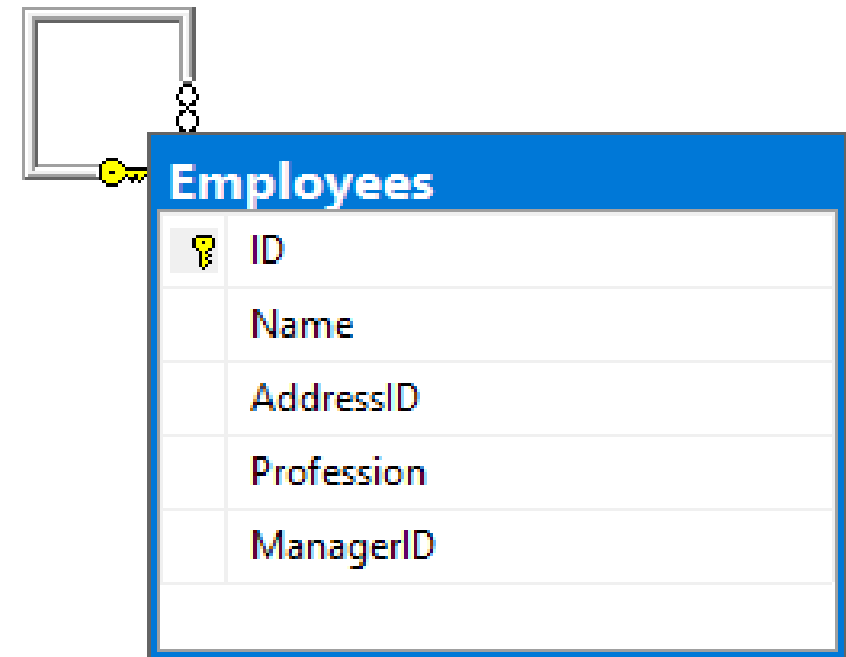
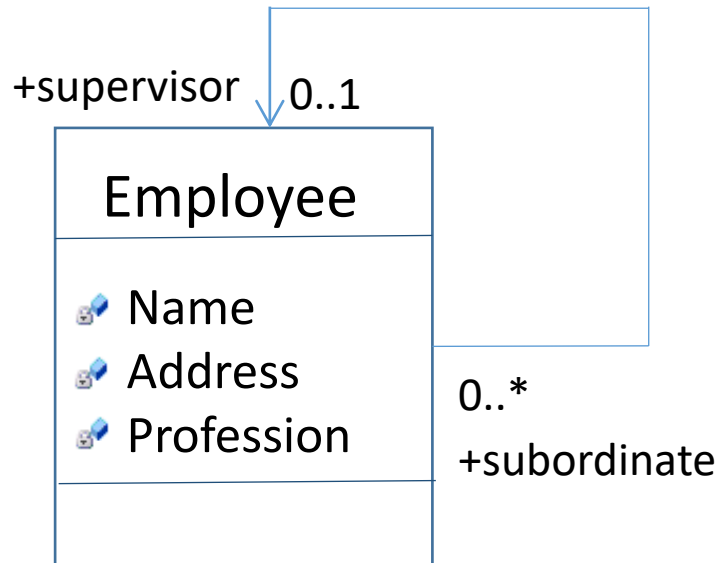
- mapping aggregation / composition



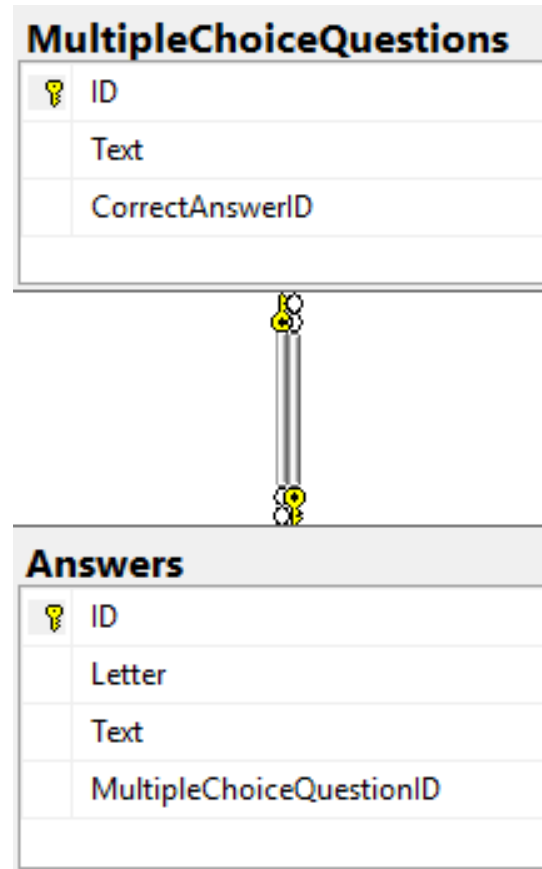
- mapping aggregation / composition



- mapping reflexive associations
- add a new field, referencing the same table (recursive relationship)
- ON DELETE CASCADE - error



Obs. 2 different tables, each with a foreign key referencing the other one, ON DELETE CASCADE - error



References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Si11] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts (6th Edition), McGraw-Hill, 2011
- [Ga09] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book (2nd Edition), Pearson Education, 2009