# FIB - Disseny de Bases de Dades

## Relational translation - 2

### Knowledge Objectives

1. Explain the two possible implementations of symmetric reflexive associations in a RDBMS.
2. Remember where to place the attributes of the UML associations when they are implemented on a RDBMS, depending on their multiplicity.
3. Distinguish associative classes that appear just due to UML syntax constraints from those truly associative classes.

### Understanding Objectives

1. Translate from a UML class diagram (with around 10 classes, some maybe associative classes, and related by associations, generalizations and aggregations) into an SQL schema

### Application Objectives

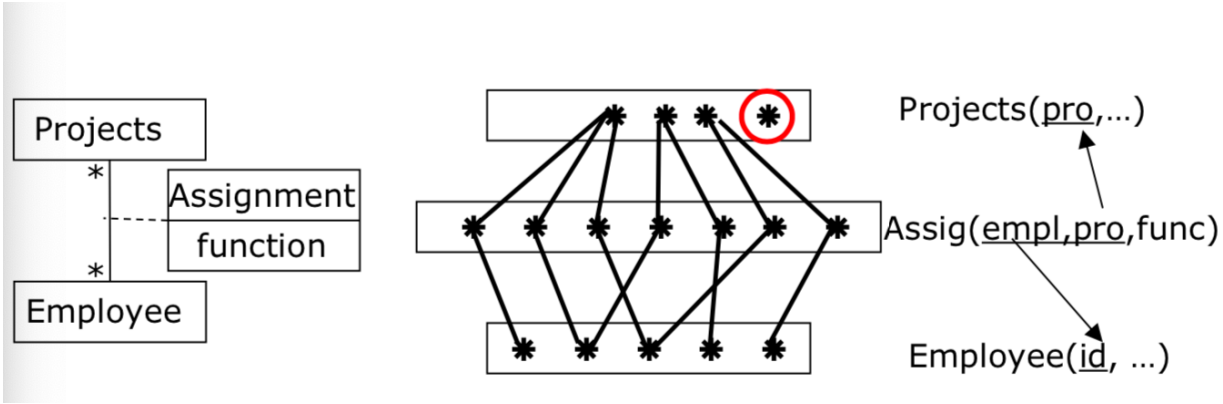
1. Choose and justify the best option to translate from a UML class diagram (with less than 10 classes, some maybe associative classes, related by associations, generalizations and aggregations) into an SQL schema, given the statistics of participation of the instances in the relationships and the queries.
2. Given a multivalued attribute and an explanation of its usage, choose and justify the best option to implement it in a RDBMS.

### Multiplicities

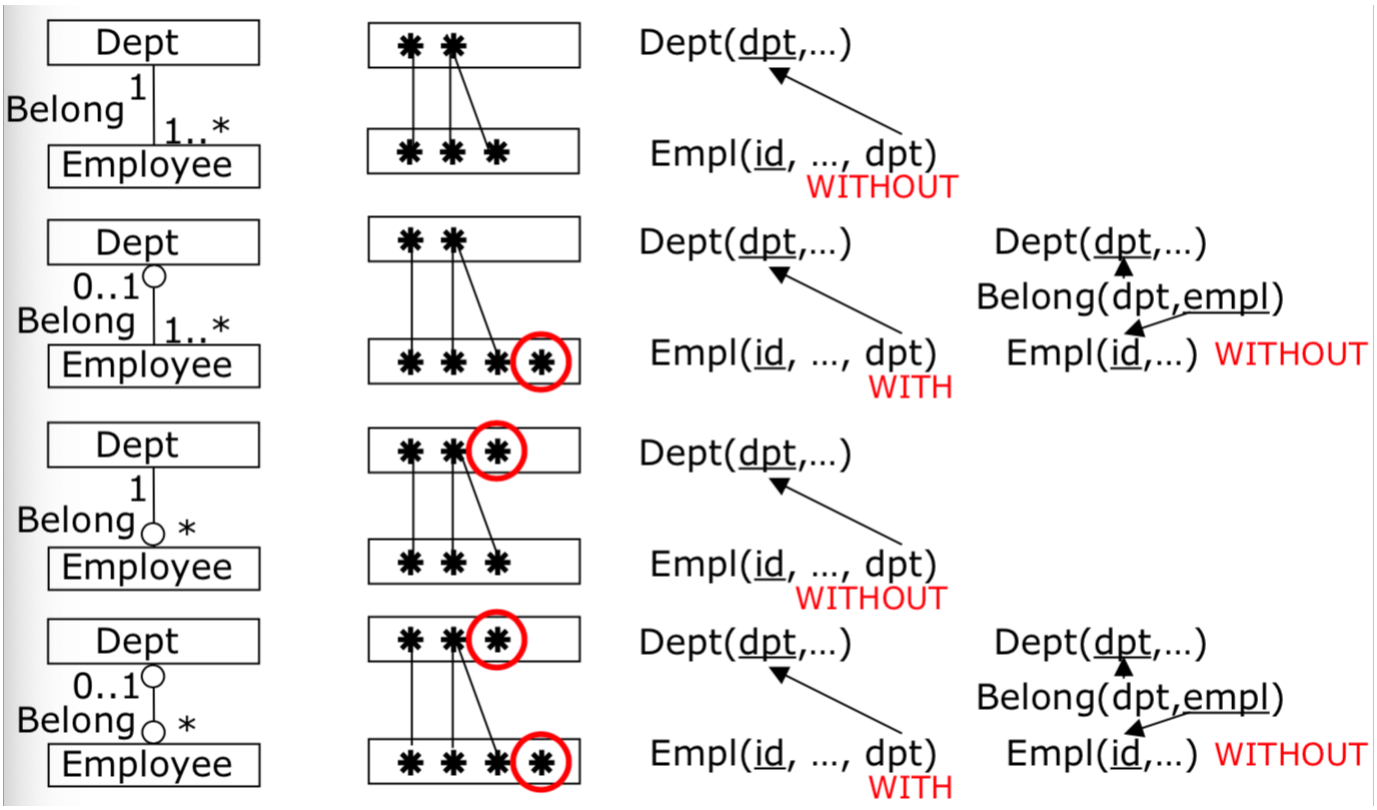
1. Maximum multiplicity:
   - Each one, how many at most? Give raise to:
     - `*–*`
     - `1–*`
     - `1–1`
2. Minimum multiplicity:
   - Could zeros exist (possible no participation of an instance in the relationship)?
     - Above cases split into subcases
     - If there are zeros, do they give rise to nulls?

### Binary association ( `*–*` )

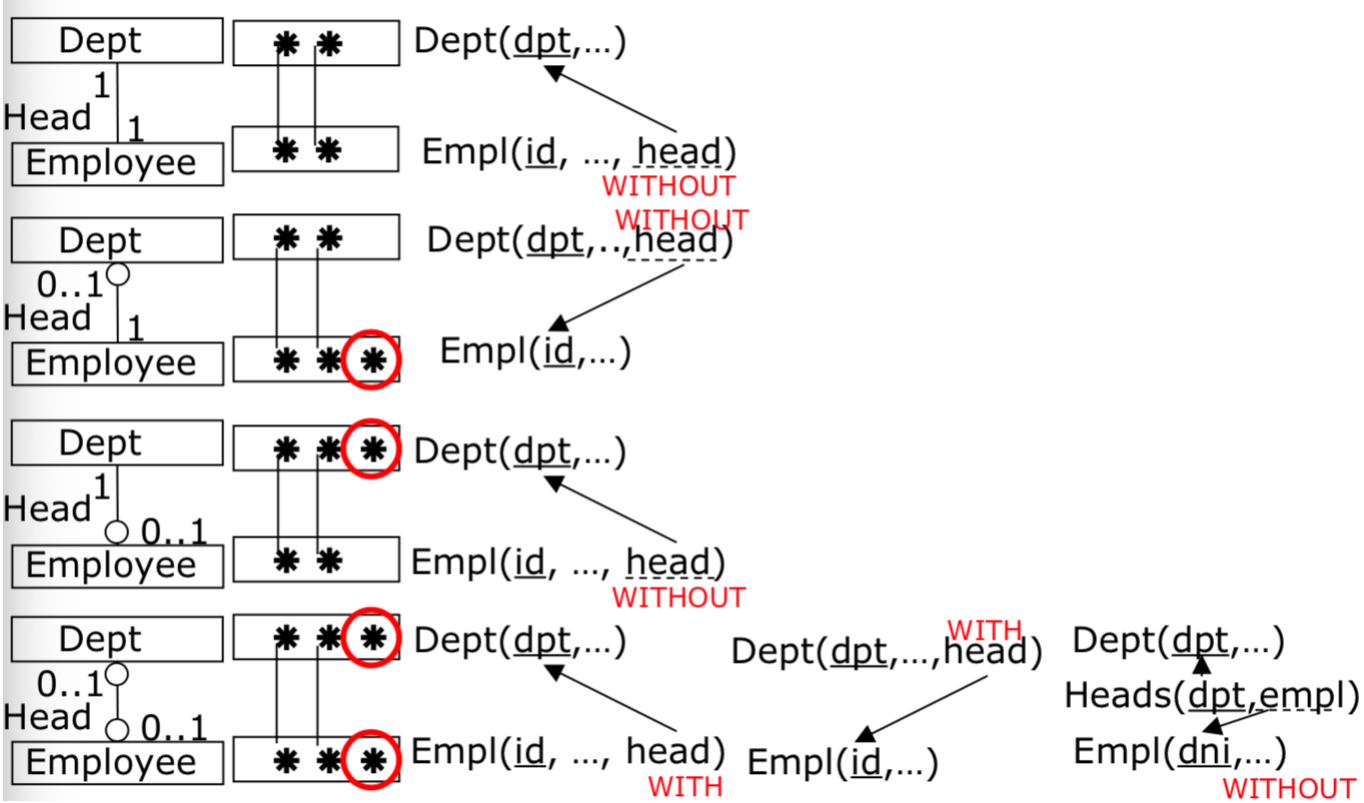
ALWAYS A NEW TABLE!




## Binary association ( 1–* )

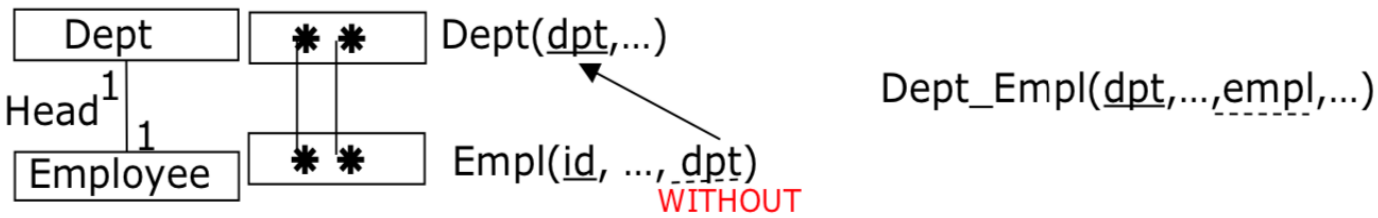



## Binary association ( 1–1 )




## Fusing classes/Choosing PK

- Time
- Space
- Change frequency




Which candidate key would you choose?

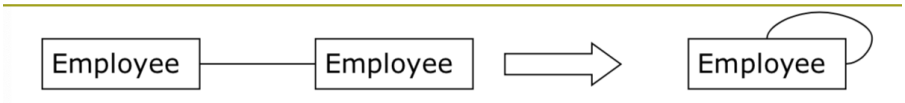
| Country_President( | country, ..., | president,...) |
|---|---|---|
| | USA | B. Obama |
| | Spain | M. Rajoy |

## Attributes of relationships

- `*-*` or `n-ary` (common)
  - In the table representing the association
- `1-*` (uncommon)
  - If any, in the table representing the association
  - Otherwise?
- `1-1` (rare)

- If any, in the table representing the association
- If only one table (fusion), in it
- Otherwise?

## Reflexive associations

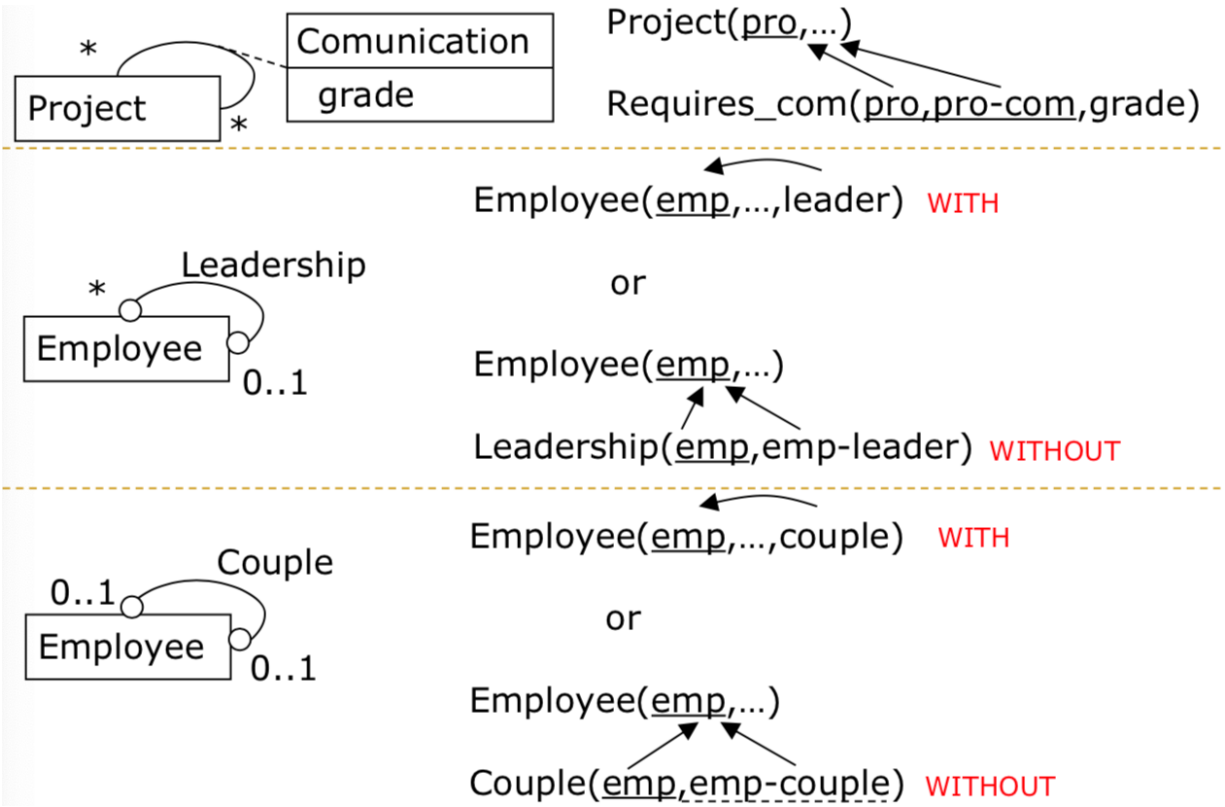



- Valid multiplicities:
  - `*–*` (Relatives)
  - `1–*` (Mother)
  - `1–1` (Couple)
- Singularity:
  - May be symmetric or not

| Brother1 | Brother2 |
|---|---|
| John | Peter |
| Peter | John |

| Friend1 | Friend2 | Grade |
|---|---|---|
| John | Peter | 10 |
| Peter | John | 2 |

## Reflexive multiplicities



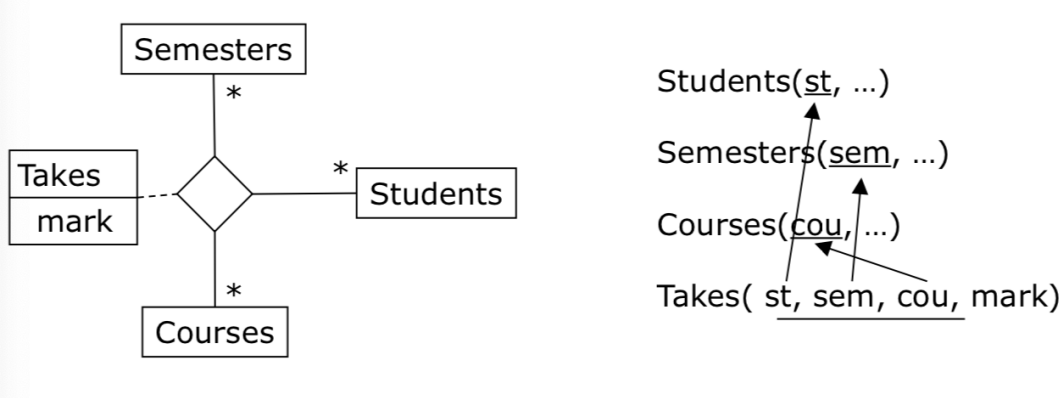

## Symmetric reflexive associations

- We must preserve the property. Triggers may bring a satisfactory solution:
  - `INSERT (a, b) -> INSERT (b, a)`
  - `DELETE (a, b) -> DELETE (b, a)`

    - `UPDATE...`
- We may store only half of the pairs
    - `CREATE VIEW` to simulate the whole set of pairs
    - Trigger `INSERT (a, b)` -> look if (b, a) already present
    - Trigger `DELETE (a, b)` -> look if (b, a) is present instead
    - Trigger `UPDATE...`

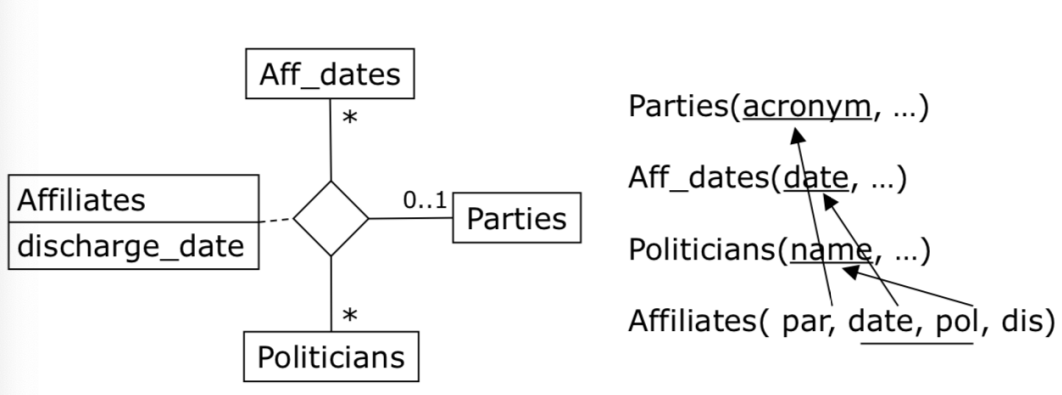
## Ternary associations ( `*-*-*` )

ALWAYS A NEW TABLE!



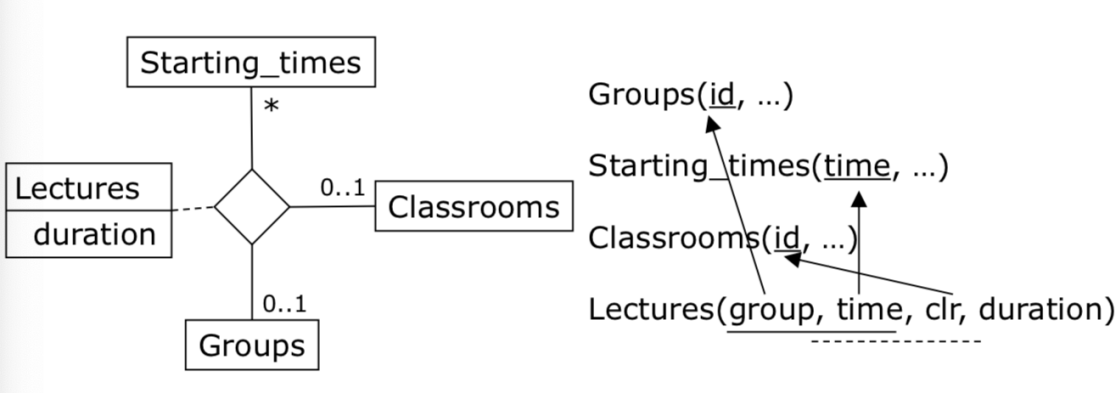

## Ternary associations ( `*-*-1` )

ALWAYS A NEW TABLE!



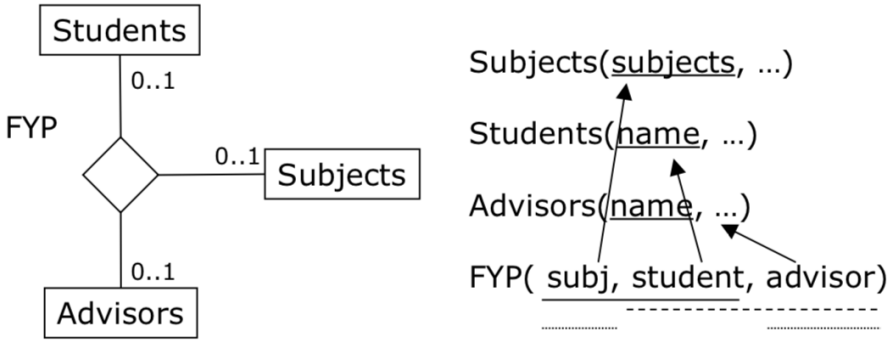

## Ternary associations ( `*-1-1` )

ALWAYS A NEW TABLE!




## Ternary associations ( `1-1-1` )

ALWAYS A NEW TABLE!

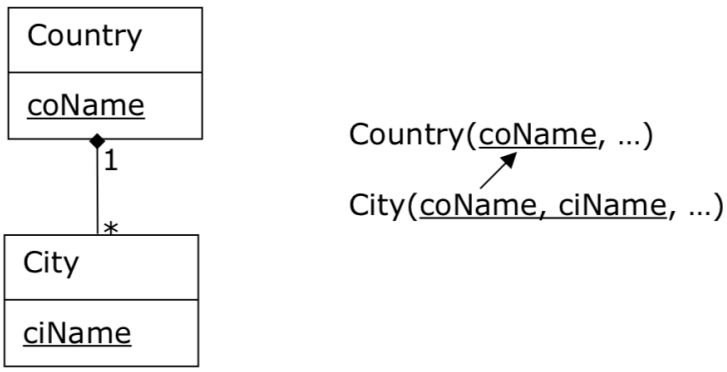



## N-ary associations

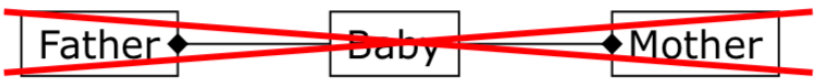
- Binary: A new table or foreign key
- Ternary: A new table
- Quaternary: A new table
- ...

## Compound aggregation

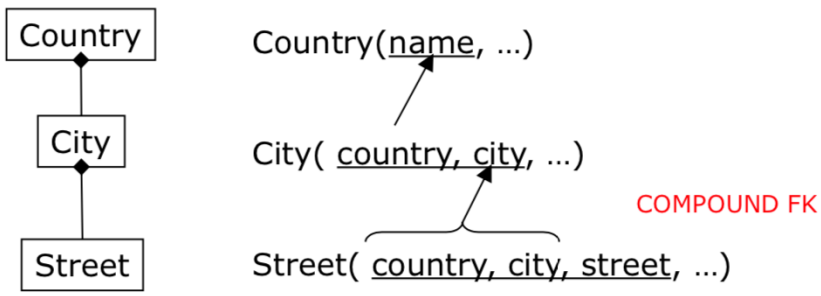
- Weak class, with regard to the external key of the classical relational model




- A given class cannot be part of two

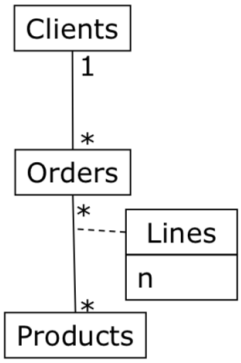



- Composition cannot have zeros at “to-one” side
- Compositions can be chained

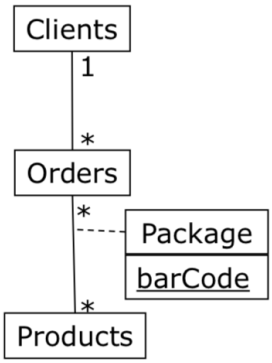



## Class vs Association

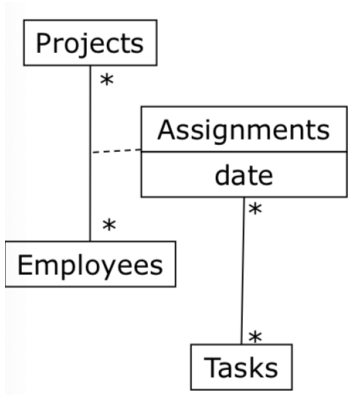
ASSOCIATION

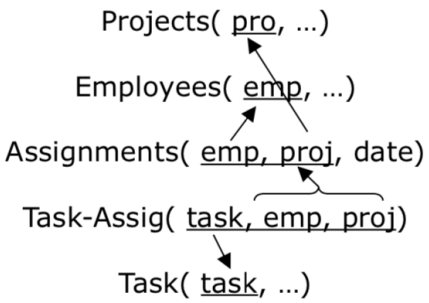



ASSOCIATION-CLASS

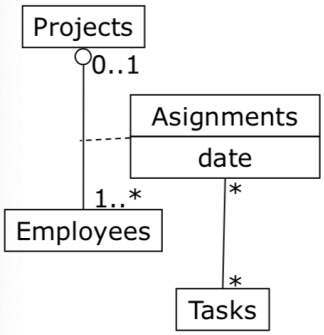


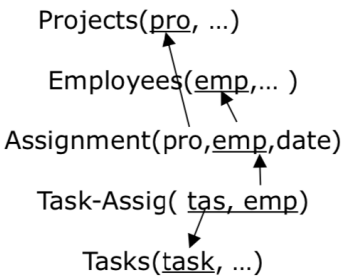

## Association classes



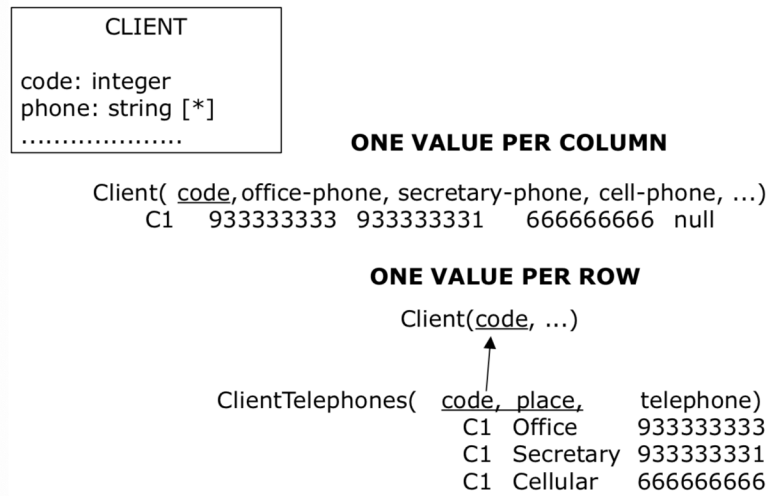

FOREIGN KEY (emp, proj) REFERENCES Asignaments



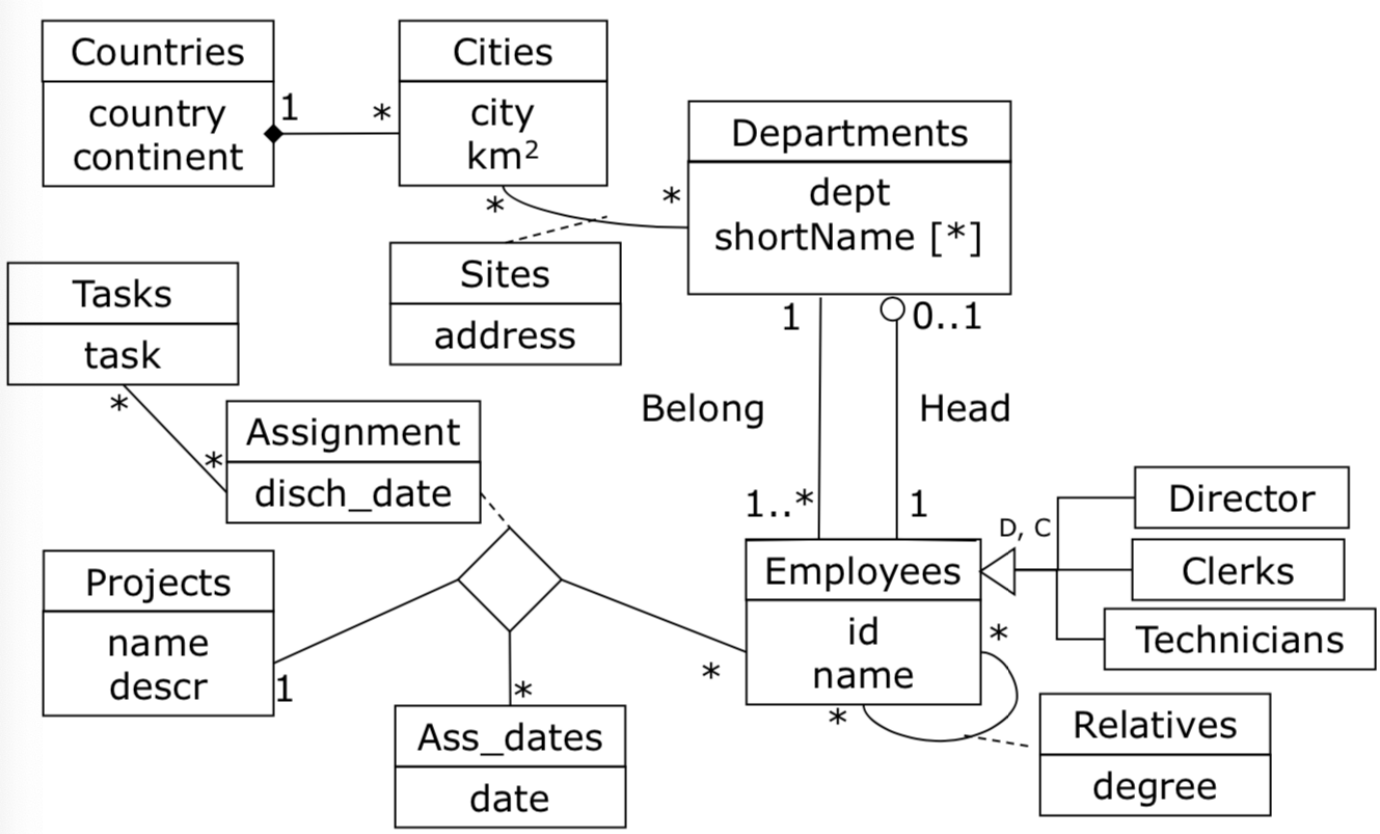

FOREIGN KEY (emp) REFERENCES Assignments

## Multivalued attributes




| Per column | Per row |
|---|---|
| Fixed number of values | Variable number of values |
| Few values | Many values |
| Generates nulls | There are no null values |
| One I/O | Many I/O |
| Global processing | Partial processing |
| Natural PK | Artificial PK |
| Less space | More space |
| Hard to aggregate | Easy to aggregate |
| Many CHECKs | One CHECK |
| Lower concurrency | Higher concurrency |

# Example: Conceptual schema




## Summary

- Translation of relationships
- Possible overlooking of classes
- Attributes of relationships
- Class or relationship
- Multivalued attributes