2 Conceptual Database Design

2.3 Integrity Constraints

2.3.1 Constraint types

2.3.2 Cardinality constraints

2.3.3 Weak entities

2.4 Modeling patterns

2.4.1 Modeling historical data

2.4.2 N-ary relationships

2.4.3 Generalization / specialization …
… and more

Elmasri, Navathe: chap 3 + chap 4; Kemper, Eickler: 2.7 – 2.13

- Context

Requirements analysis	
Conceptual Design	System analyst DB designer Application programmer
Schema design - logical ("create tables")	Application programmer DB administrator
Schema design - physical ("create access path")	
Loading, administration, tuning, maintenance, reorganization	DB administrator

2.3.1 Constraint types

Important concept

- Integrity constraints
  (Invariants, assertions, restrictions)
  - A set of predicates, the database must *always* fulfill during its lifetime
  - Taken from requirements, formally stated in DB schema
- Case study
  "There is always at least one tape for each movie we track, and each tape is always a copy of a single, specific movie"
  "Not all of our movies have star actors" (Negative constraint)
- Implicit assertions: context knowledge
  A tape cannot be loaned by more than one customer at a time
  An actor may be starring in more than one movie

Constraint types

Assertions: constraints which must hold for each state of the database

Similar: Object constraint language (OCL) for UML

- Types of constraints:
  - Attribute constraints
    Movies are made after 1.1.1900
  - Cardinality constraints
    Tape can have been lent by zero or one customer at any time
  - General constraints
    If there exists only a DVD copy of a film, then no extracharge
    Can be regarded as business rules

Constraint types

Attribute constraints

- Attribute must / may have a value

  Movie has a title, but director not necessarily known

- Value restriction

  Movies are made after 1900 : movie.year > 1900

Typical ERM constraint

- Attributes must not be structured

  attribute address with fields city street etc. not allowed

- Attributes must have *at most* one value

  Phone number: only one allowed

  Use set notation for multivalued attributes:

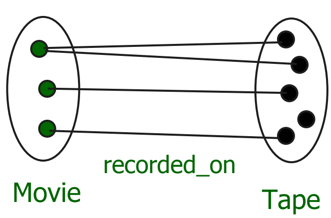
  {phone_Number}

2.3.2 Cardinality constraints

Important concept

- Restriction of relationships:

  let <r> be a relationship of <E1> and <E2>
  how many instances of <E1> may be related according to <r> to a single instance of <E2> and vice versa?

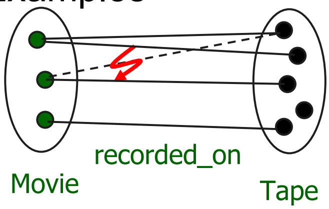


- Number of copies of a movie ≥ 1
- A tape can be loaned by at most one customer at a time
- Number of tapes a customer has rented ≥ 0
- Exactly one movie on a tape

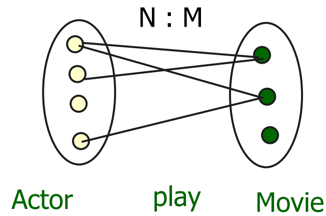
UML terminology: multiplicity

Cardinality constraints N:M notation

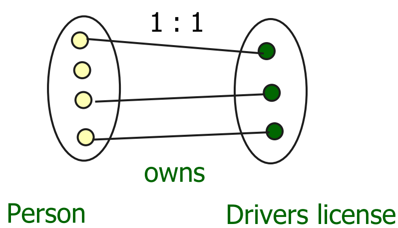
- Examples



contradicts 1 : N, not allowed

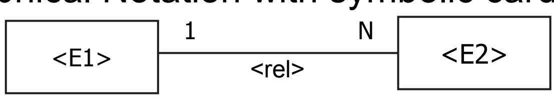


arbitrary binary relation



Cardinality constraints N:M notation

- Graphical Notation with symbolic cardinalities



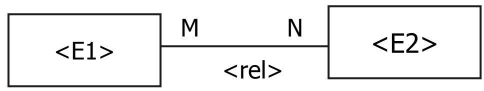
Classical ER-M notation for cardinality constraints

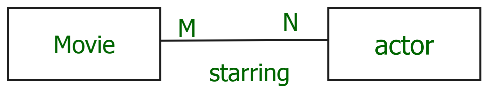
A particular movie may exist (as a copy) on many tapes,
but a particular tape stores a copy of only one movie.

Formally: recorded_on :: Tape -> Movie is a function
Expresses the fact that the movie on a tape is unique

Cardinality constraints N:M notation

- M:N-Relationships
  every instance of <E1> may be related according to <rel> to every instance of <E2>

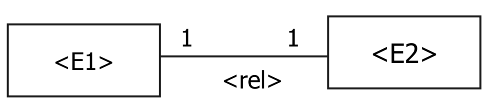


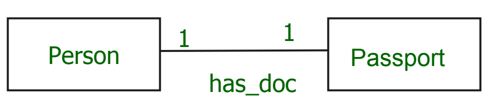


Actors play in one or many movies,
in a movie typically many actors play.

Cardinality constraints N:M notation

- 1:1-Relationships
  every instance of <E1> may be related according to <rel> to every instance of <E2>



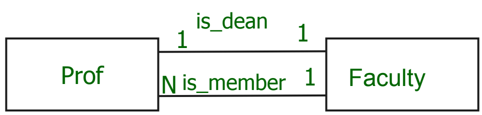


1:1 relationships: not frequently used

Cardinality constraints and modeling alternatives

- Case: University administration
  among others: faculties and professors
  how to model the *dean* of a faculty?

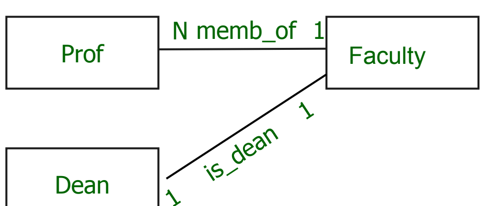
As an attibute:



As a relationship:
Faculty has only one dean, prof may be dean of only one faculty.

Cardinality constraints and modeling alternatives (2)

- Case study continued:



Entity:
Could make sense, if the dean must not be a professor

Note: in both cases "is_dean" is a 1:1 – relationship

But: every dean *entity* is the dean of one faculty.
As opposed to: every prof is dean *or not.*
Means: Function is_dean::Fac -> Prof is not surjective

Difference cannot be expressed until now!

Cardinality constraints (min,max)-Notation

(min,max)–Notation for cardinality constraints

1: N – Notation not strong enough to express all cardinality constraints

Minimal values

e.g. zero tapes rented by a particular customer

or each tape stores a copy of a movie ("at least one")

Maximal values

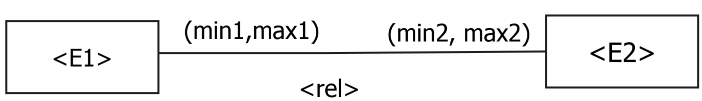
e.g. on a tape there is at most one movie ("at most one")

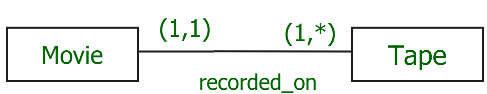
a customer may rent arbitrary many tapes ("many")

Cardinality constraint (multiplicity) notation is also used in UML

Cardinality constraints (min,max)-Notation

Graphical notation





"A a particular movie may occur 1 or many times in this relation"
or: "For each movie in the DB there is at least one tape" and
"There are no empty tapes"

min := 0	| 1	means: optional	| mandatory
max := 1	| *	means: single	| multiple

Sometimes natural numbers used for min, max
Does not make much sense, since systems are unable to check these fine granular constraints

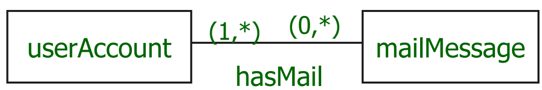
Cardinality constraints example

- Another example

A database supported email system is designed to have user accounts and mail messages related by the relationship "has mail".

Constraint :

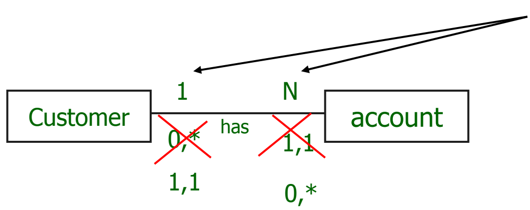
- user has zero or more mails
- mail message belongs to at least one user, perhaps to many users (those with many receivers)



Cardinality constraints notations

Important note

In the classical ER-Model, (min,max)-Notation does not conform to the N:M-Notation



You find this in many text books, 1:N and (min,max) interchanged

Good news:
UML-multiplicity is conformant to 1:N notation.

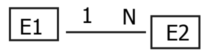
We use UML-multiplicity with (min,max) annotation, min,max $\in$ {0,1,*}

laxly : "1:N – relationship", "N:M – relationship"

Cardinality constraints semantics

Let R ⊆ E1 X E2 be a relationship between entity sets E1 and E2

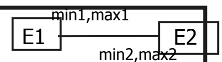
R is 1:N ⇔ R is a function R: E2 → E1
⇔ for all extensions of R ∀e2 ∈ E2:
|{e1 | e1 ∈E1 ∧ (e1,e2) ∈ R }| ≤ 1



R is 1:1 ⇔ E2 → E1 is an injective function
R is M:N ⇔ R is a relation, but not a function

Classic ER-M notation!

E1-R has (min1, max1) cardinality
⇔ for all extensions of R and for all y ∈E2
min1 ≤ |{x| x ∈ E1 ∧ (x,y) ∈ R }| ≤ max1



E2-R has (min2, max2)
⇔ for all extensions of R and for all x ∈E1
min2 ≤ |{y | y ∈ E2 ∧ (x,y) ∈ R }| ≤ max2

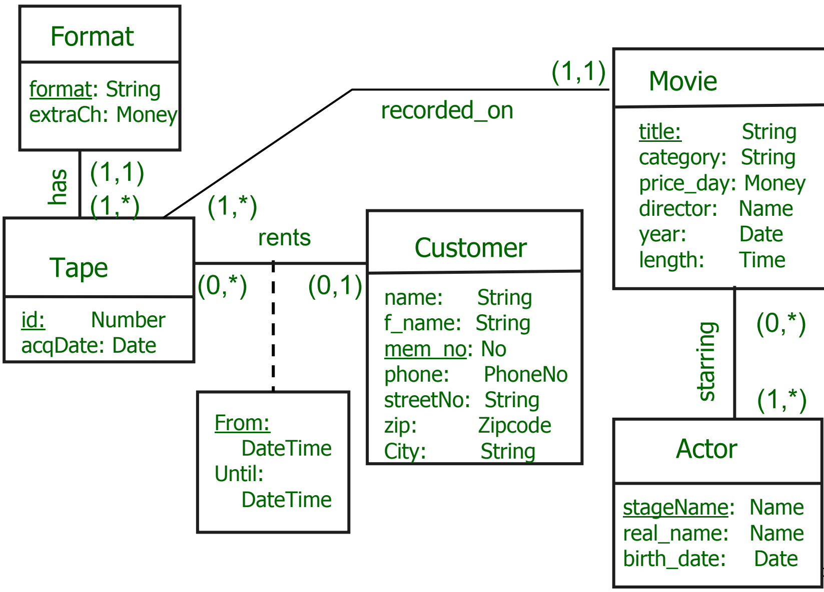
Cardinality constraints notations

	mandatory/ multiple	optional/ multiple	optional/ single	mandatory / single
E-RM / (UML)	(1,*) (1,n)	(0,*) (0,n)	(0,1)	(1,1)
E-RM/1:N	N or M	N or M	1	1
UML+	1 ..* k..j k	0 ..* * 0 .. k	0..1	1

+ : k and j are natural numbers; n, N,M in the ERM are literals

More notations in use!, eg. Oracle 'crow's feet'-Notation

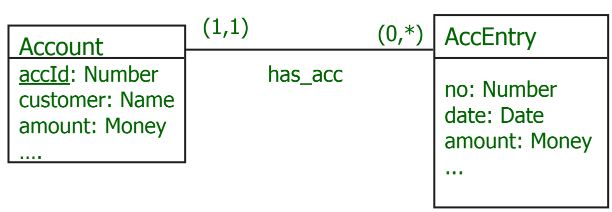
Constraints case study



2.3.3 Weak entities

Motivating example:

modeling of bank accounts and the transaction history for each account



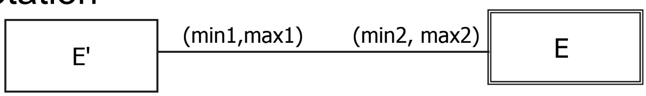
- Issue

  one of the entities (accounting entry, AccEntry) does not have a key. Uniqueness cannot be guaranteed without referring to a related entity (here: account).

  "There may be many entries "4711" but only one for a particular acount"

Conceptual Modeling Weak entities

- Weak entity:
  an entity e of type E , which is only identifiable by a value k and the key k' of <u>one</u> entity e' of a different type E' .
- e is said to be existentially dependent on e' (on the entity type level: E dependent on E')
- Notation



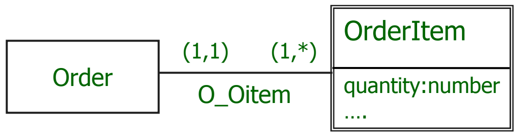
- Cardinality: min1 = 1
  max1 = 1 : why?
  min2, max2 ?

Conceptual Modeling Weak entities and UML

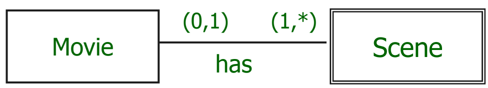
- No weak entities in UML
  each object has identity by its "object id",
  which is a pointer, referencing the object
- Database modeling paradigm:
  Objects (entities) with identical values for all attributes are identical (like in mathematical sets), except for weak entities
- Object oriented modeling paradigm
  Any two objects are distinguishable by their oid (a physical address!), even if all attributes have the same value

Conceptual Modeling Weak entities

- More examples



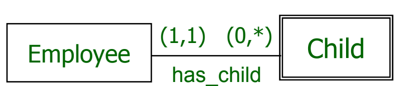
Orders and the items ordered



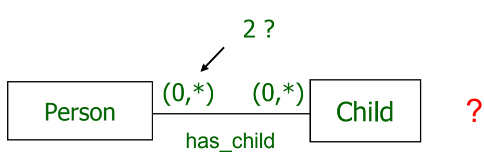
A movie and its scenes

Conceptual Modeling Weak entities

- Modeling decision not always evident



or



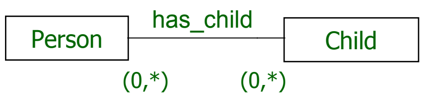
pros and cons?

2.4.1 Modeling historical data

- What are historical data?

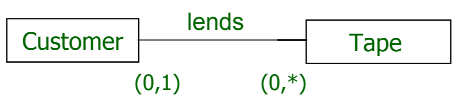
Important

Not time related:



time invariant: a particular relationship between e1 and e2 will never change.
Rare case.

Time variant



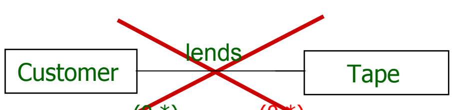
A particular relationship (c1, v1) disappears when tape has been returned

Acceptable but in most cases we want to keep track of the history

Historical data

Keeping track of changes…

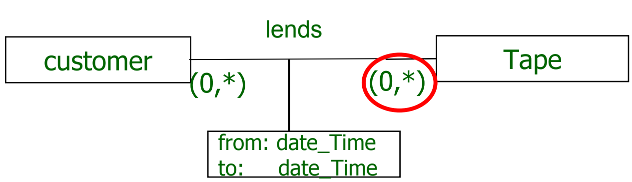
A tape may be rented by many customers…



??

… but not at the same time.

Yet another way to model reality…

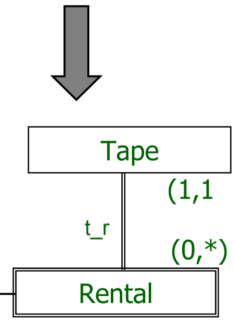


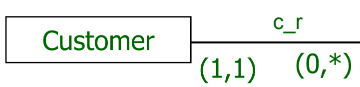
…but constraint lost: a tape is lent to at most one customer

Conceptual Modeling: historical data

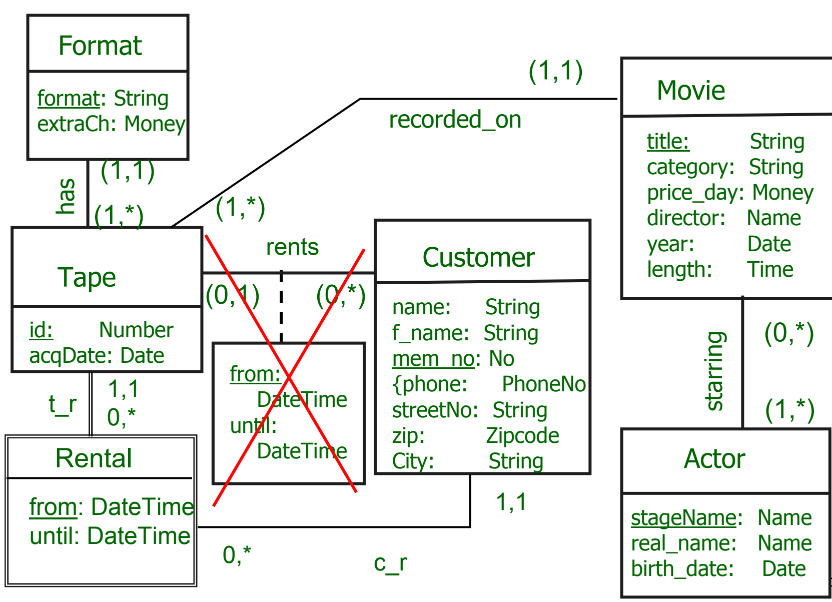
Solution:

Introduce a weak entity which keeps track of related entities over time (here: rental of each particular tape over time)





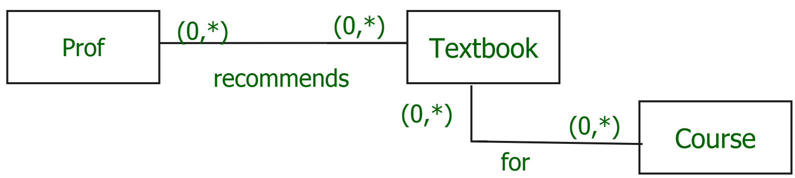
Case study revised



2.4.2 N-ary relationships

- Motivation example

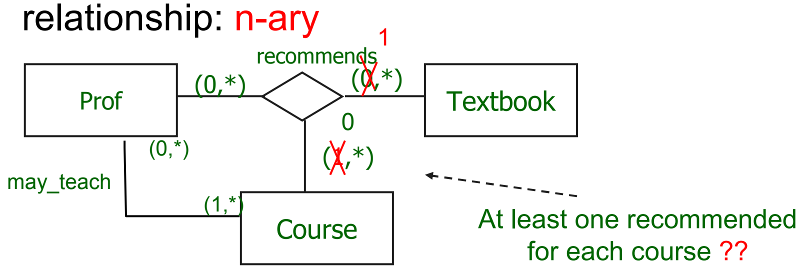
  Suppose you want to represent the following facts in a university database:
  prof X suggests textbook Y for course A
  prof X suggests textbook T for course B
  prof Z suggests textbook T for course A



Wrong: Conceptual model does NOT represent the information given above

N-ary relationships

- More than two entity sets involved in one relationship: n-ary

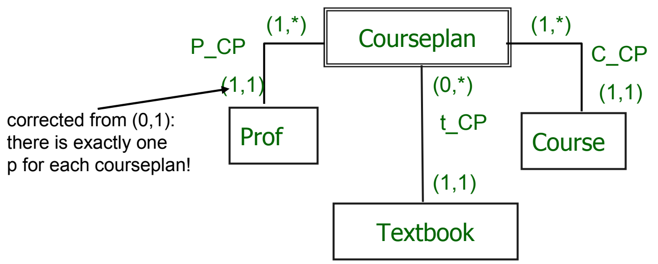


- Cardinality

  Each prof is entitled to recommend at least one book for a course -> (1,*)

N-ary relationships modeled binary

- N-ary relationships expressed by binaries



  – Introduce a weak entity type for the relationship and binary relationships to the other entity types, weak entity may be dependent from any of the other three entity types.

2 Conceptual Database Design
2.3 Integrity Constraints

2.3.1 Constraint types

2.3.2 Cardinality constraints

2.3.3 Weak entities

2.4 Modeling patterns

2.4.1 Modeling historical data

2.4.2 N-ary relationships

➔ 2.4.3 Generalization / specialization …
… and more

Elmasri, Navathe: chap 3 + chap 4; Kemper, Eickler: 2.7 – 2.13

2.4.3 Generalization / Specialization

- Modeling similar objects by totally different entities is confusing

  Example:

  Suppose two types of customers of the video-shop:
  - frequent customers
  - regular customers

  both have most attributes in common, e.g. membership, address, name

  Frequent customers have a "credit line" and some more attributes

Customer

membership: Number
name: Name
first_name: Name
address: A_type
{phone: Phone_Type}

FreqCustomer

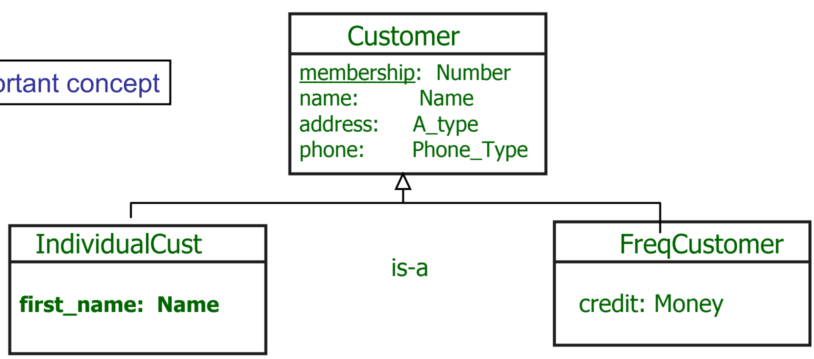
membership: Number
name: Name,...,
credit: Money
address: A_type
{phone: Phone_Type}

Redundant: employ OO principle of generalization / inheritance

Generalization / specialization

- Generalization / specialization hierarchy allows to factorize common attributes of different entities

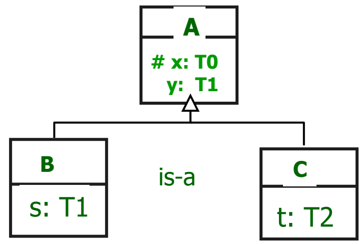
Important concept



Standard relationship is-a between subtypes and super types
Note: not really types but sets, see next slide

Generalization / Spezialization

- Different semantics of generalization: type versus set
  - Instances of A, B and C are different (OO-interpretation) but share some attributes
  - All instances of B and of C are also instances of A (DB-interpretation)
    $B \subseteq A$ and $C \subseteq A$
  - "is-a" therefore different from ordinary relationships

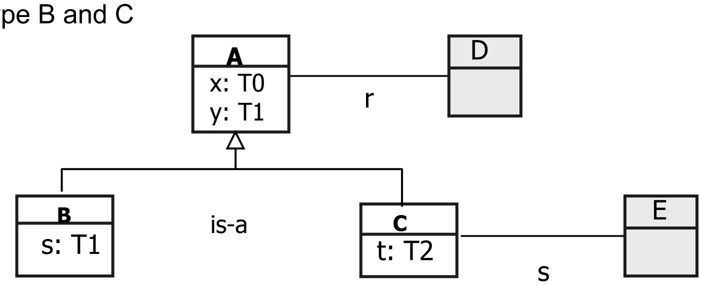


  - Special cases:
    - Disjoint specialization: $C \cap B = \varnothing$
    - Complete specialization: $A = B \cup C$, no extra tupel in A

Generalization and relationships

- Different relationships may be defined for different entity types of the generalization hierarchy

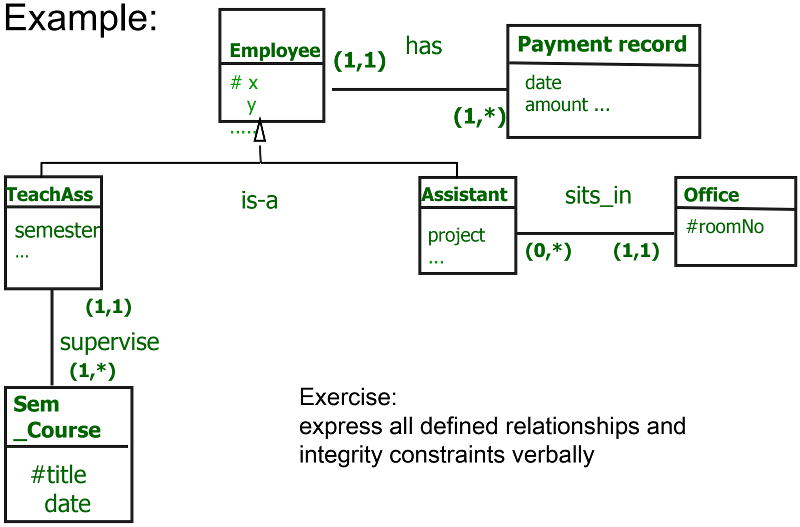
If A is a generalization of B and C, then all relationships defined for A are implicit relationships for all entities of type B and C



- Entities from entity set A - and therefore those of B and C are related by r to entities from D
- Only entities from set C are related by s to entities from E

Generalization

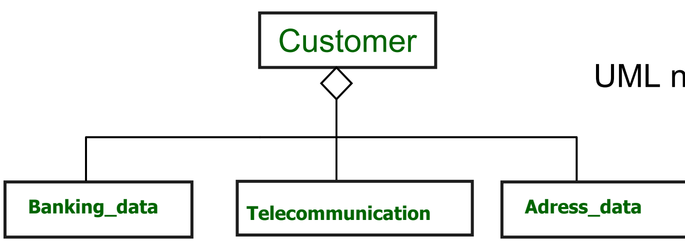
- Example:



Exercise:
express all defined relationships and integrity constraints verbally

Aggregation

Aggregat: different entitiy types form a new one



UML notation

Not frequently used in database design

Conceptual Design

View integration

For big projects different "views" of
the application are modeled independently

Very important: model data and processes the data are used for

e.g. student administration, exams, teachers and personel

Integrate different partial designs
→ Conceptual design of the overall DB

Not as easy as it sounds….

DB design and constraints — Short summary

- Constraints
  - Restrict the state of the database
  - Database should always be coherent with real world
  - Types of constraints
    - Value restriction
    - Cardinality restriction
  - 1:N notation imprecise, use only for oral communication
  - Use (min,max)-Notation coherent with UML
- Uniform modeling "patterns"
  - Historical / time related data
  - N-ary relationships: model with binary relationships and a another entity type
  - Generalization