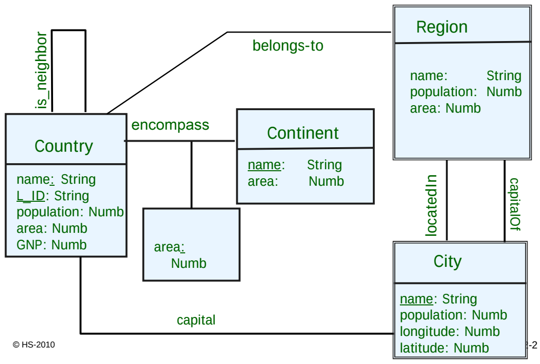
2 Conceptual Database Design

Bernstein et al.: chap. 4; Elmasri, Navathe: chap 3 + chap 4;
Kemper, Eickler: 2.7 – 2.13

Conceptual Design: case study

Freie Universität Berlin

Constraints !??



2.3.1 Integrity Constraints

Freie Universität Berlin

Important concept

Def.: An **Integrity constraint** is an invariant (assertion, restriction) of the state of a database.
ICs are **predicates**, a database must fulfill during its lifetime.

They **result from requirement analysis**, context and common sense knowledge
Formally stated in DB schema

Case study
From requirements
"Names of regions are not necessarily unique"
"A regions belongs to exactly one country"
Common sense knowledge
"Population is always >= 0 - or unknown"
"A country has one and only one capital"

Constraint types

Freie Universität Berlin

Attribute constraints
- Attribute value restriction
- Attribute value must / may exist **([not] NULL]**

General constraints
- Relations may be symmetric
  e.g. neighbor-rel of countries

Cardinality constraints

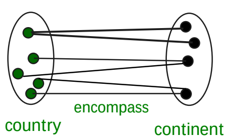
How many entities of type E may be in relationship R to an entity of type E'?
e.g. to how many countries can a region belong?
How many regions can a country have?

2.3.2 Cardinality constraints

Freie Universität Berlin

Important concept

Def.: A **cardinality constraint** of a relationship R between entity types E1', E2' **restricts** the **number of entities** E1, E2 participating R

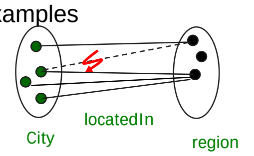


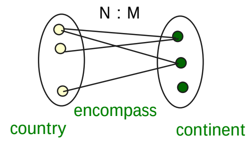
UML terminology: **multiplicity**

Cardinality constraints,N:M notation

Freie Universität Berlin

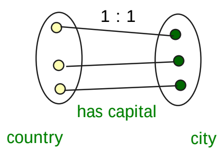
Examples





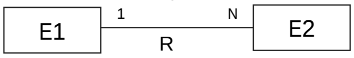
contradicts 1 : N, not allowed

arbitrary binary relation



1:N relationship

Graphical Notation with symbolic cardinalities



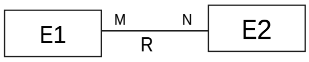
One E1-entity is related (R) to arbitrary many E2-entities, but one E2-entity is related (R) to only one E1-entity

Traditional ER-M notation for cardinality constraints

Formally**:** locatedIn:: city -> region **is a function**

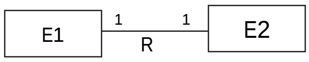
More relationships

M:N-Relationships
every instance of E1 may be related according to R to every instance of E2



R is M:N means: **no restriction on the pairs of R**

1:1-Relationships
every instance of E1 may be related according to R to excactly one instance of E2 and vice versa



(min,max)-Notation

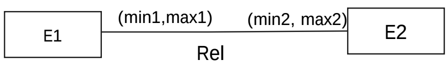
More **precise cardinality** restrictions by specifying **minimal and maximal number of entities**

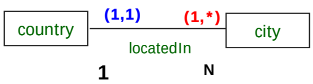
Many cities *locatedIn* one country, at least one
⇒ min=1, max = *
A country has zero, one or many *neighbors*
⇒ min=0, max = *

(min,max)-Cardinality constraint (multiplicity) notation also used **in UML associations**.

(min,max)-Notation

Graphical notation

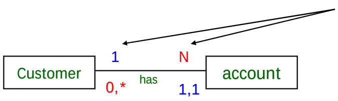




Note
- **1:N** notation characterizes **relationship R**.
- (min,max) characterizes entity ***and*** relationship R, in general different for E1 and E2

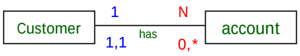
CAVEAT: Misleading Notation

Traditional ER-Model, **(min,max)**-Notation **does not conform to N:M-Notation**



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

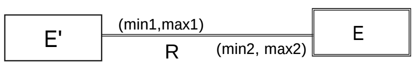
UML-multiplicity conformant to 1:N notation



WE USE THIS NOTATION

Use (min,max) annotation which conforms to UML, min,max ∈ {0,1,*}

Cardinality of weak entities

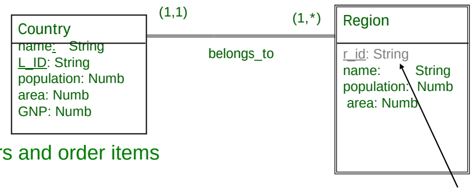


e is **existentially dependent on e'**

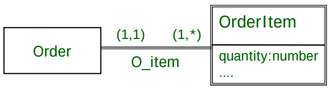
Cardinality:
min1 = max1 = 1
min2 = 0 | 1
max2 = 1 | *

Weak entity: example

Countries and regions



Orders and order items



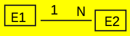
Sometimes an artificial key makes sense, not in this case

© HS-2010 03-DBS-Conceptual-2-13

Cardinality constraints: semantics

Let $R \subseteq E1 \times E2$ be a relationship between entity sets E1 and E2

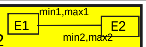
R is 1:N $\Leftrightarrow$ R is a function R: E2 $\rightarrow$ E1
$\Leftrightarrow$ for all extensions of R $\forall e2 \in E2$:
$|\{e1 \mid e1 \in E1 \wedge (e1,e2) \in R\}| \leq 1$



Traditional ER-M notation!

R is 1:1 $\Leftrightarrow$ E2 $\rightarrow$ E1 is an injective function
R is M:N $\Leftrightarrow$ R is a relation, but not a function

E1-R has (min1, max1) cardinality
$\Leftrightarrow$ for all extensions of R and for all $y_0 \in E2$
$min1 \leq |\{x \mid x \in E1 \wedge (x,y_0) \in R\}| \leq max1$



E2-R has (min2, max2)
$\Leftrightarrow$ for all extensions of R and for all $x_0 \in E1$
$min2 \leq |\{y \mid y \in E2 \wedge (x_0,y) \in R\}| \leq max2$

© HS-2010 03-DBS-Conceptual-2-14

Cardinality constraints notations

	mandatory/ multiple	optional/ multiple	optional/ single	mandatory/ single
ERM / (UML)	(1,*) (1,n)	(0,*) (0,n)	(0,1)	(1,1)
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

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

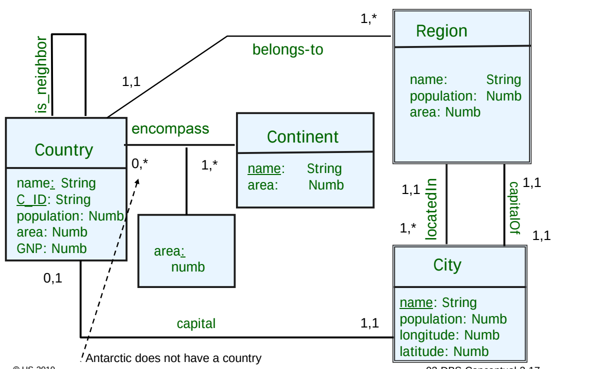
© HS-2010 03-DBS-Conceptual-2-15

2 Conceptual Database Design

2.3 Integrity Constraints
2.3.1 Constraint types
2.3.2 Cardinality constraints
2.4 Extended ER Modeling
2.4.1 Inheritance / Generalization
2.4.2 Modeling historical data
2.4.3 N-ary relationships

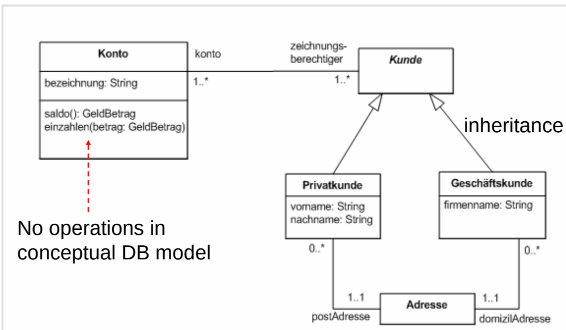
Bernstein et al.: chap. 4; Elmasri, Navathe: chap 3 + chap 4; Kemper, Eickler: 2.7 – 2.13

Conceptual Design: case study



© HS-2010 03-DBS-Conceptual-2-17

UML class diagram



from Wikipedia (Klassendiagramm)

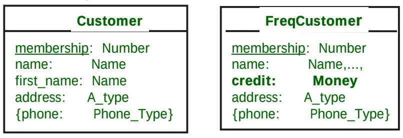
© HS-2010 03-DBS-Conceptual-2-18

2.4 Extended ER (EER)

Example:

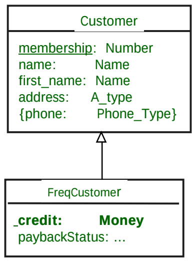
Suppose two types of customers of a video-shop:
- frequent customers
- regular customers

Generalization



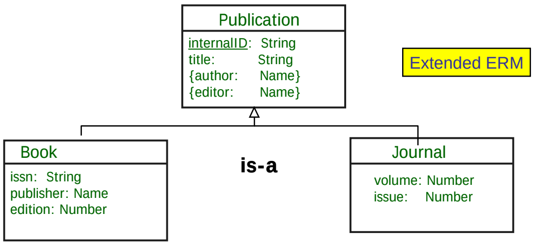
Redundant:
relationships of Customer has to be duplicated for FreqCustomer
⇨employ object oriented principle of generalization/ inheritance

Generalization: Example



2.4.1 Generalization / specialization

Factorize common attributes of different entities



Standard relationship **is-a** between subtypes and super types

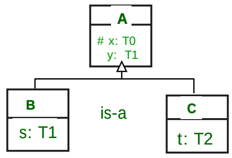
Generalization / Spezialization

Semantics of generalization: type versus set

Instances of A, B and C are different but share some attributes (OO-interpretation)

All instances of B and of C are also instances of A (DB interpretation)

B ⊆ A and C ⊆ A



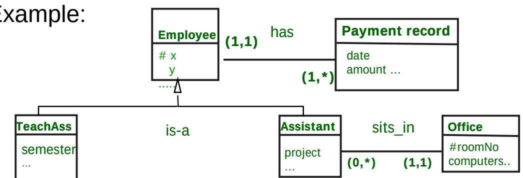
Def.: **Specialization** is called
- **disjoint** iff C ∩ B =∅
- **complete** iff A = B ∪ C, and every tuple is either B or C

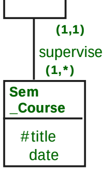
more general definition: n>2 specializations

No overwriting... why not?

Generalization

Example:

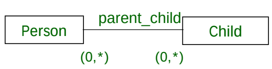




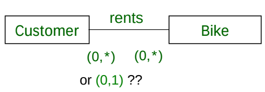
Note: 'sits_in' only defined for the subset 'Assistant' of Employees, 'has' defined for all employees.

2.4.1 Modeling historical data

Important



Time invariant:
a particular relationship between e1 and e2 will never change.



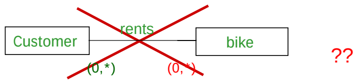
Time variant:
A particular relationship (c1, v1) may disappear, a new one may be established

In many cases:
History of time variant relationships has to be recorded

Case study and historical data

Keeping track of changes...

Use case: Bike rental

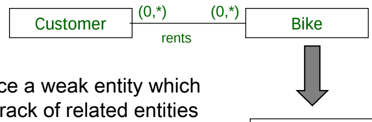


A bike may be rented by many customers…

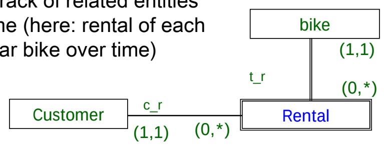
… but not at the same time

Conceptual Modeling: historical data

Solution:



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

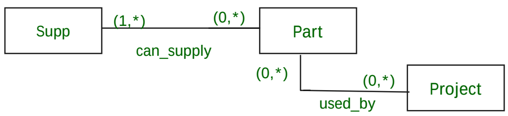


Question: Why 'Rental' existentially dependent on bike, not customer?

2.4.2 N-ary relationships

Motivation example

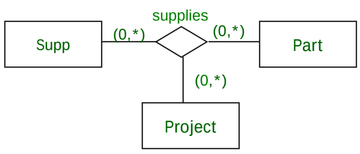
Represent the following facts in a database:
supplier X delivers part Y to project Z
supplier A delivers part P to project Z
supplier B delivers part Q to project S



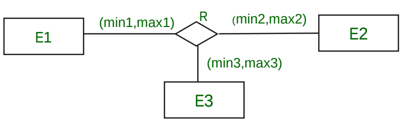
Wrong: Conceptual model does NOT represent the information given above

N-ary relationships

Def.: A relationship is call **n-ary relationship R**, if more than 2 entity sets are involved in the R



N-ary relations and cardinalities



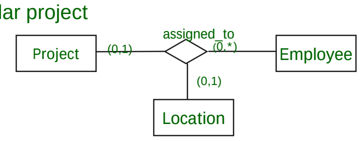
Def.: E1-R has (min1, max1) cardinality
$\Leftrightarrow$ for all extensions of R and for all (y,z) $\in$E2 X E3
min1 $\leq$ |{x| x $\in$ E1 $\wedge$ (x,y,z) $\in$ R }| $\leq$ max1

E2-R, E3-R correspondingly.

N-ary relationships

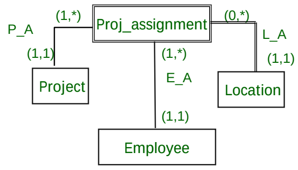
Example:

- Employees assigned to a project, work at one location for this project.
- Employees work for one project at a particular location
- At each location several employees may work for a particular project



Question: May an employee work for different projects?
Which constraints cannot be expressed?

N-ary relationships by N binary relationships

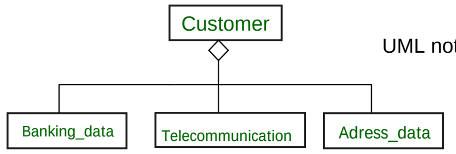


Introduce a **weak entity** type for the relationship and binary relationships to the other entity types.

Different constraints expressed than n-ary relationship

Extended ER: Aggregation

Aggregate: different entity types form a new one



UML notation

Not frequently used in database design

No particular notation for **composition** as in UML

Conceptual Design

Def.: **View integration** is the process of integrating conceptual models, which are related but have been designed separately, into one single model.

For big projects different "views" of the application make sense: model different, more or less independent parts of the "real world". (compare "partitioning approach" to "top down approach")

Important: model **data *and processes*** the data are used for

e.g. student administration, exams, teachers and human resources

View integration

Integrate different partial designs into the conceptual design of the overall DB

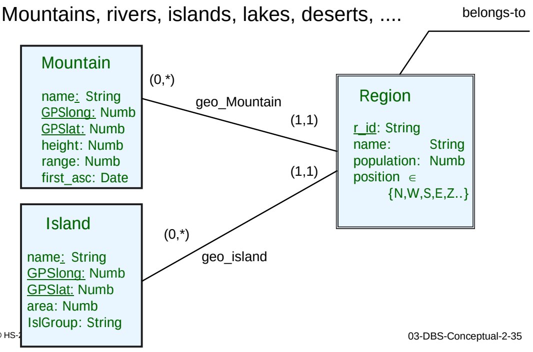
Running example:

a) countries, cities...
b) Organizations (Government, national / internat. organization
c) geography: lakes, mountains, rivers...

Not as easy as it sounds….

View integration: "Geography"

Mountains, rivers, islands, lakes, deserts, ...



DB design and constraints

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 but sufficient in many situations

Uniform modeling "patterns"

Historical / time related data
N-ary relationships: model with binary relationships and a another entity type
Generalization