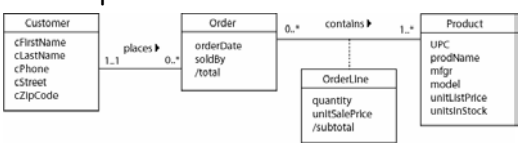
2 Conceptual Database Design
2.1 Requirement analysis
2.2 Modeling languages

References: Kemper / Eickler chap 2, Elmasri / Navathe chap. 3
Garcia-Molina / Ullmann / Widom: chap. 2

2.1.1. Overview Lifecycle

Requirement analysis -> Text

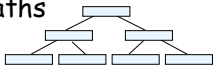
Conceptual Design -> Conceptual Model



Logical Schema Design -> Database schema

```
CREATE TABLE Order
 ODate  DATE
soldBy INTEGER FOREIGN KEY
       REFEREBCES Peronal(PID)
 CID INTEGER   FOREIGN
 KEY REFERENCES Customer (CID));
```

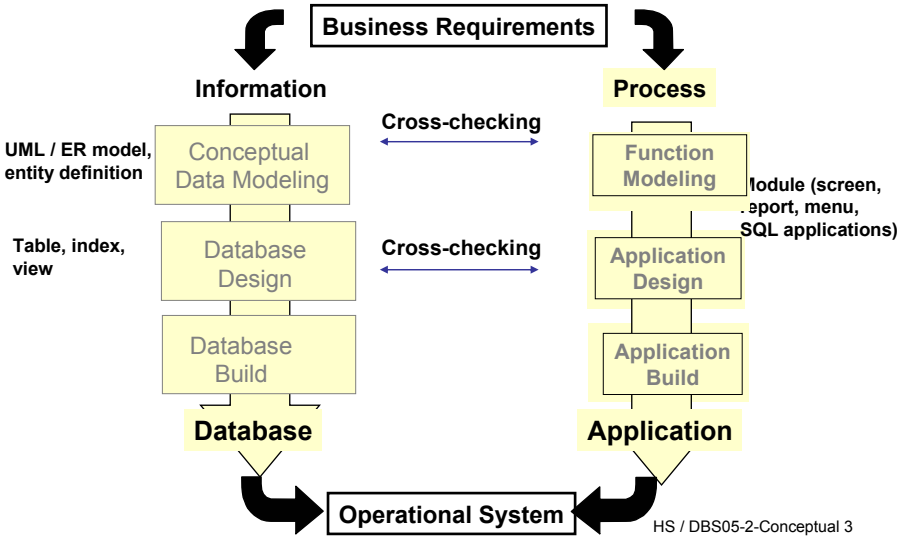
Physical Schema Design -> Access paths



Administration

Database Design: Overview

Hand-in-hand design of DB and intended applications



Database Design: Terminology

- Database Design
  - Process of defining the overall structure of a database on
  - Different layers of abstraction

    Conceptual, logical, physical level
- Includes "Analysis" and "Design" from SE
- DB Modeling: defining the "Static model" using formal or visual languages

DB	SE
Requirements	Requirements
Conceptual modeling	Analysis
Logical modeling	Design
Physical modeling	Implementation

2.1.2 Requirement Analysis

Most important: communicate with your customers!

Tasks during RA:

- Identify essential "real world" information (e.g. interviews)
- Remove redundant, unimportant details
- Clarify unclear natural language statements
- Fill remaining gaps in discussions
- Distinguish data and operations

Requirement analysis & Conceptual Design aims at focusing thoughts and discussions !

Requirements: Case study

"I'm the owner of a medium size video store. We have over 10.000 video tapes that we need to keep track of. Since a few month, we also have DVDs.
Each of our video tapes has a tape number. For each movie, we need to know its title and category (e.g. comedy, suspense, drama, action, or sci-fi). Yes, we do have multiple copies of many of our movies. We give each movie a specific id, and then track which movie a tape contains. A tape may be either Beta or VHS Format.
We always have at least one tape for each movie we track, and each tape is always a copy of a single, specific movie. Our tapes are adapted to the movie lengths, so we don't have any movies which require multiple tapes. The movies are stored on shelf according to their category sorted by movie title.
We are frequently asked for movies starring specific actors John Wayne and Katherine Hepburn are always popular. So we'd like to keep track of the star actors appearing in each movie.....
(cont. next page)

Requirements: Case study (cont)

.... Not all of our movies have star actors. Customers like to know each actor's "real" birth name and age. We track only actors who appear in the movies in our inventory. We have lots of customers. We only rent videos to people who have joined our "video club". To belong to our club, they must have good credit. For each club member, we'd like to keep their first and last name, current phone numbers, and current address. And, of course each club member has a membership number. Then we need to keep track of what video tapes each customer currently has checked out. A customer may check out multiple video tapes at any given time. Rentals are for one or more days, each movie with an individual price per day. Furthermore we additionally charge 1 $ per beta format tape, 2 $ for a DVD and another $ for movies longer than 2 hours. Maximum rental time is 4 weeks. The customer gets a bill with movie titles, individual prices and total amount, when he /she returns movies."

Requirement Analysis

- First step: filter essential information vs unimportant details
  - Essentials
    - There are customers, tapes, movies, ...
    - A movies may have many copies, but a tape (or DVD) contains exactly one movie
    - Customers have a customer identification (id)
    - Four weeks maximal rental time .....
  - Unimportant details
    - "...DVDs since a few month"
    - "... John Wayne.."
    - Names of the categories (but categories are important!)
    - Tapes on shelf (since we don't design a tape robot) ...
  - Note: what is important depends on the applications

Requirement Analysis

- Clarify unclear statements
  - Video club: admission / annual fee?
  - Charge per tape: price for one day the minimum?
  - ...
- Fill gaps
  - Any discounts?
  - What happens after maximal rental time?
  - ...
- Distinguish data from operations
  - Processing a bill
  - Becoming a member of the club ....

2.2.1 Basic modeling primitives

Conceptual modeling

- Distinguish between types (classes) and individual facts (metadata vs data)

  The name of this woman is *Kunz* with first name *Tamara*.

  As opposed to:

  A person is identified by first name, last name and birth date.
- Describe reality on a type level
- Use a graphical language in order to get an overall impression of the domain modeled
- The language should use the "right primitives"

This is an old problem of philosophy: how to describe the world in an appropriate, comprehensible way?

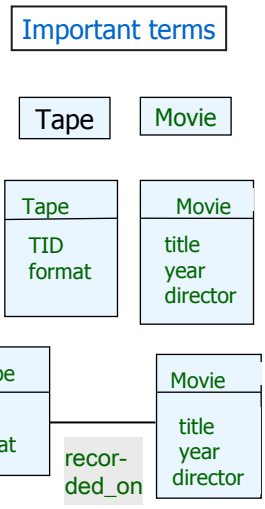
Basic modeling primitives

Modeling the "Real World"

– Entity (type)
  - something which exists, has a name

– Attribute
  - property of an entity

– Relationship
  - connects two or more entities



"Non sunt multiplicanda entia praeter necessitatem"
William van Ockham, English philosopher, 13th century
(Principle of Economy, Law of Parsimony)

Basic modeling primitives

Attributes have values

Example: Starring is *John Wayne*

Values live on the database level, not in the schema!

Design decision: attribute or entity ?

Example:

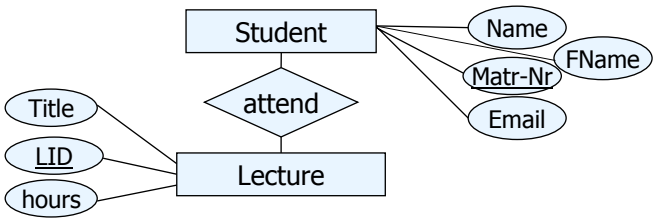
- *color* is an attribute for describing *cars*
- *color* is presumably an entity for the *car manufacturer* having attributes like *tone, quality, supplier, chemical formula*,..

2.2.2 Modeling notations and languages

Entity-Relationship-Model

- data-oriented: static modeling of data
- 1976 introduced by P.P. Chen

  (Peter P. Chen: The Entity-Relationship Model - Toward a Unified View of Data. ACM TODS 1(1): 9-36, 1976, see Reader)
- Classical Graphical notation with squares, bullets and diamond



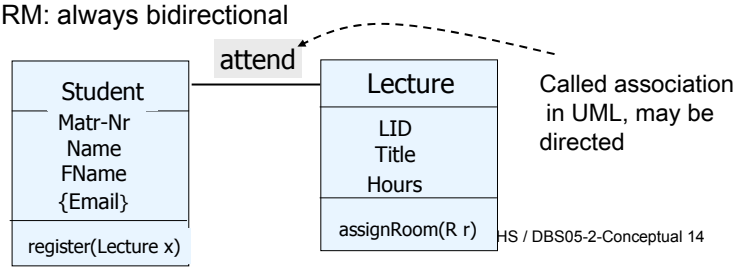
Graphical modeling languages

- Unified modeling language (UML)
  - Modeling of data and operations
  - Object oriented flavor

    e.g: each object (entity) has identity - a unique pointer

    ERM: entities having the same type and the same attribute values are indistinguishable
  - Attributes may be constructed (lists, sets, arrays,…)
  - Relationships are directed (uni- or bidirectional)

    ERM: always bidirectional



ERM and UML

- E-R modeling language and UML obviously related
  - E-R language simpler
  - More appropriate for modeling of data
  - Object-oriented modeling by extended E-RM concepts (see below)
  - semantic differences (see below)
- Drawing "attribute bullets" ?
  - Use UML notation for entities and attributes
  - Use association notation of UML instead of diamonds
- Basically use UML notation with ERM semantics

Entity

What is an "entity" ?

- particular instance (e.g. the movie "Jurassic Park")
- The type of an instance / all instances ("movie")
- sometimes: a set of instances (e.g. all movies of the rental shop)

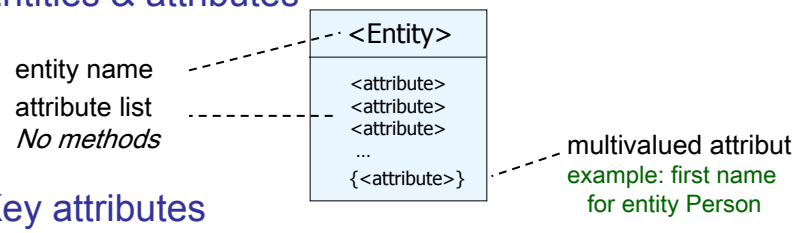
The context should clarify the meaning

Sometimes explicitly: entity set / entity type

Analog terminology for relationships

2.2.3 Conceptual Design: Basics

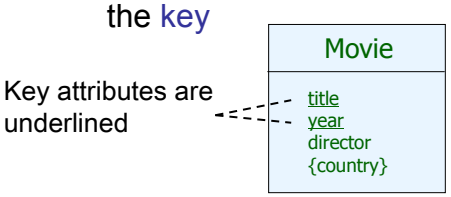
- Entities & attributes



- Key attributes

"Axiom" of ER and Relational DB:

Two individual entities can always be distinguished by the values of some of its attribute(s), together called the key

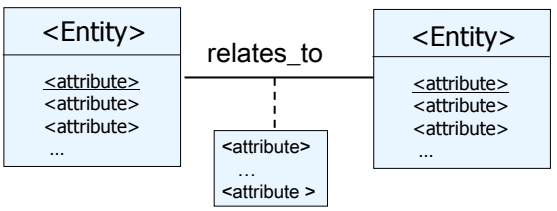


Key attributes *together* identify an entity.

Frequently: just one attribute, e.g. matrNr

Conceptual Design: Basics

- Relationships

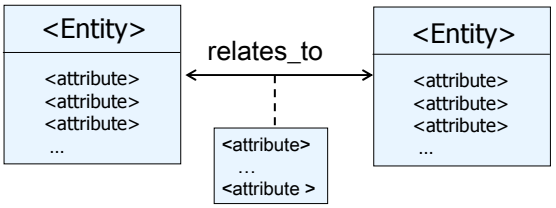


- Always have a name
- Do not have a direction
- May have attributes
- Do not have a key

Terminology:
Relationships often called relations. Different from relations of RDM !

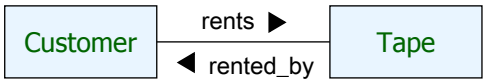
Conceptual Design: Comparison with UML

- UML-Terminology
  - Class = entity type (- Attribute = member)
  - Object = entity
  - Association = relationship
  - NO keys
  - Relationship may have a direction

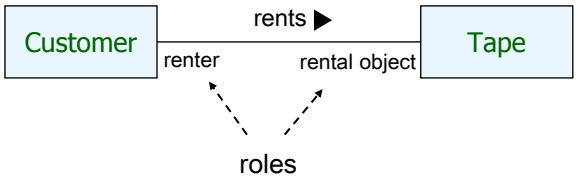


Conceptual Design: Basics

- Notation
  - sometimes attributes are omitted
  - Infrequent: order of relationship

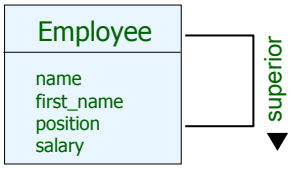


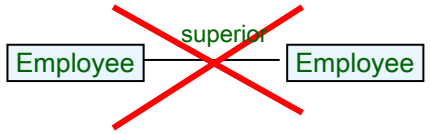
- Role names



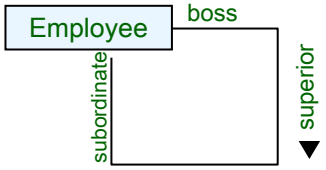
Conceptual Design: Basics

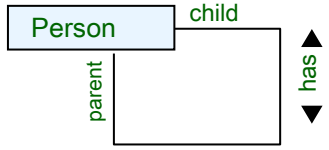
- Recursive relationships





Roles: particularly useful in recursive relations

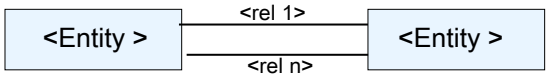




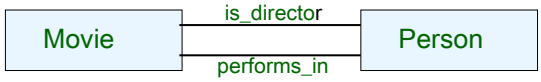
Both example model hierarchical relationships

Conceptual Design: Basics

- Multiple relationships



There may be no, one or many relationships between entity types



Note: nothing said about representation (implementation) at this point

Conceptual Design: Basics

Domains and types

- Attributes may be typed
- Representation of data not really relevant at this stage

  title: VARCHAR /* variable character string ?

- More important: domain
  Semantic category which characterizes attributes of the same kind

  title: nameOrSentence

- Domains not used frequently
- Type / domain assignment mostly done in a separate step

2.2.3 From requirements to models

Text to conceptual model

- The only step which cannot be automated
- Given: Requirements as "cleaned" text
- Wanted: conceptual database design

Rough guideline: correspondance between…

entities and nouns

For each movie, we...

relationships and verbs

... which movie a tape holds ...

attributes and adjectives or phrases like “has a..”, “is...a”

...has a tape number.

From requirements to models

Some heuristics: *entities*

- If it cannot be uniquely identified, it usually isn't an entity

  video club
- If only one instance exists, it might not be an instance

  video store
- Find out synonyms (in this context)

  customer and club member
- Give a definition of the entity and eliminate synonyms

  a customer is a member of "the club". Only customers may rent videos. *Customer is synonym to club member*.

From requirements to models

Heuristics: *attributes*

- Find the attributes for each entity.
  Sometimes expressed by auxiliary verb:

  Tape… has a tape number
- Which one *identifies* an instance?

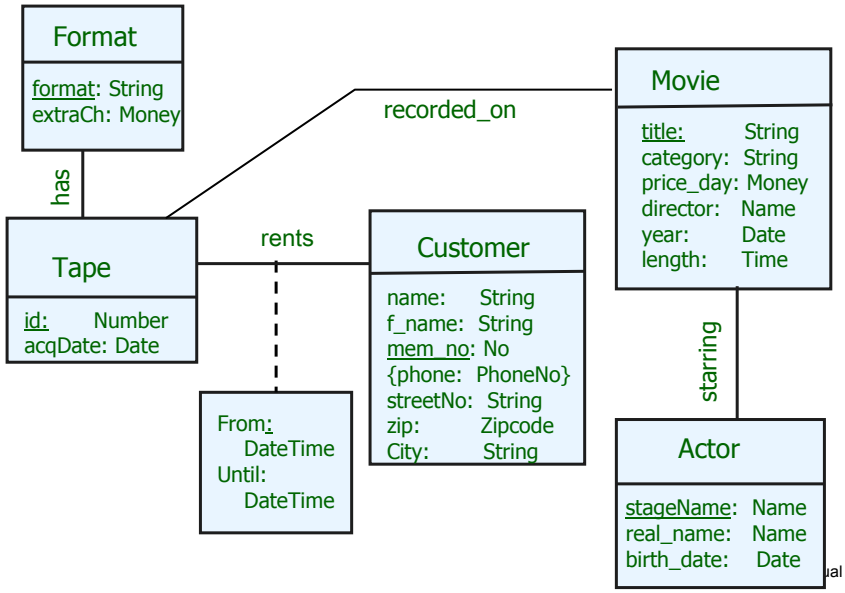
  In most cases not expressed explicitly but context knowledge

  Customer: name, first_name, phone, address, membership#
- Decompose compound attributes
  (e.g. address ⇨ location, street, zip)
- Check for multiple values
- Use lower case for attributes, uppercase for entities

From requirements to models: case study

I'm the owner of a small video store. We have over 3.000 video tapes that we need to keep track of. Since a few weeks, we also have DVDs.
Each of our video tapes has a tape number. For each movie, we need to know its title and category (e.g. comedy, suspense, drama, action, or sci-fi). Yes, we do have multiple copies of many of our movies. We give each movie a specific id, and then track which movie a tape contains. A tape may be either Beta or VHS Format. We always have at least one tape for each movie we track, and each tape is always a copy of a single, specific movie. Our tapes are adapted to the movie lengths, so we don't have any movies which require multiple tapes. The movies are stored on shelfs according to their category sorted by movie title.
We are frequently asked for movies starring specific actors John Wayne and Katherine Hepburn are always popular. So we'd like to keep track of the star actors appearing in each movie. Not all of our movies have star actors. Customers like to know each actor's "real" birth name and age. We track only actors who appear in the movies in our inventory. .... etc

Conceptual Design: case study



Summary

- Conceptual modeling: the art of structuring the data of an application domain
- Base: careful requirement analysis
- Simple, powerful base constructs: entities, attributes, relationships
- Visual (graphical) language
- E-R modeling language and UML related
  - E-R language simpler
  - More appropriate for modeling of data
  - many dialects
  - Compatibility to UML makes sense
  - Some differences, e.g. no keys in UML