Requirement Analysis & Conceptual Database Design

Problem analysis
Entity Relationship notation
Integrity constraints
Generalization
Introduction: Lifecycle

- Requirement analysis
- Conceptual Design
- Logical Schema Design
- Physical Schema Design
- Administration

- Text
- ER-Model
- Database schema
- Access paths
Introduction: Database Design Terminology

- Different from Software Engineering!
- Database Design
  - process of defining the structure of a database
  - layers of abstraction: Conceptual, logical, physical level
- Includes "Analysis" and "Design" from SE

<table>
<thead>
<tr>
<th>Database</th>
<th>Software Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Requirements</td>
</tr>
<tr>
<td>Conceptual modeling</td>
<td>Analysis</td>
</tr>
<tr>
<td>Logical modeling</td>
<td>Design</td>
</tr>
<tr>
<td>Physical modeling</td>
<td></td>
</tr>
</tbody>
</table>
Customer communication!

- Identify essential "real world" information
- 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!
"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 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 SciFi), director and year. 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.....
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 number, 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: Example

- Note: what is important depends on the applications

Essential information
- customers, tapes, movies, ...
- movies have many copies
- tape (or DVD) contains exactly one movie
- customers have customer identification (id)
- four weeks maximal rental time
- ...

Redundant, unimportant details
- "...DVDs since a few weeks"
- "... John Wayne.."
- Names of the categories (but categories are important!)
- Tapes on shelf (since we don't design a tape robot)
- ...
Requirement Analysis: Example

- Clarify unclear statements
  - Video club: admission / annual fee?
  - Charge per tape: is price for one day the minimum?
  - ...

- Fill gaps
  - Any discounts?
  - ...

- Distinguish data/operations
  - Processing a bill
  - Becoming a member of the club
  - ...
Conceptual Design: Modeling primitives

- **Entity**
  - something which exists
  - Examples: tape, movie

- **Attribute**
  - property of an entity
  - Examples: tape has id, movie has title

- **Relationship:**
  - connects two or more entities
  - Examples: tape contains a movie.

- **Attributes have values**
  - Example: “John Wayne is starring”
  - Not modelled!
Conceptual Design: Structuring the problem

- Modelling / Conceptual Design
  - typically: Entity-Relationship-Model, data-oriented
  - 1976 introduced by P.P. Chen
    

- sometimes: UML (problematic!)

```
<table>
<thead>
<tr>
<th>Student</th>
<th>Lecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>SId: Number</td>
<td>LId: Number</td>
</tr>
<tr>
<td>Name: String</td>
<td>Title: String</td>
</tr>
<tr>
<td>FName: String</td>
<td>hours: Number</td>
</tr>
<tr>
<td>Email: String</td>
<td></td>
</tr>
</tbody>
</table>

```

```
Student
- Name
- SID
- FName
- Email

attend

Lecture
- LId
- Title
- hours

```
Conceptual Design: E-R notation

- Entities & Attributes:
  - Entities have identifying (or key) attributes or sets of attributes

  ![Diagram of Entities and Attributes](image)

  - **Tape**
    - id
    - **key**
  - **Movie**
    - title
    - director
    - category

  These attributes *together* identify a movie
Relationships

- Always have a name
- Do not have a key!
- Not always reflected by the relationship name
  
  see example: "rents" is equivalent to "is-rented_by"

```
Customer  rents  Tape
```

1.12 FU-Berlin, DBS I 2006, Hinze / Scholz
“[…] We have […] 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, director and year. 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. […] Customers like to know each actor’s “real” birth name and age. We track only actors who appear in the movies in our inventory. […] 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.” [abbreviated]
Conceptual Design: Example (without customers)

**Tape**
- id
- belong_to
- hold
- Format
  - name
  - charge

**Movie**
- id
- title
- category
- year
- director
- Price_per_day
- length
- play

**Actor**
- id
- stage_name
- real_name
- birthday

**Format**
- name
- charge
Conceptual Design: E-R notation

- **Multiple relationships**

  - Student
  - Lecture
  - attend
  - hold

- **Recursive relationships**

  - Lecture
  - Person
  - predecessor
  - have
  - successor
  - parent
  - role name
  - child
Conceptual Design: E-R notation

- Relationships with attributes

  - Different semantics in separate entity:
Conceptual Design: Integrity constraints

- Integrity constraints (=Invariants)
  - Database must always meet these invariants

- Explicit state restrictions from requirements
  - Examples:
    - "There is always at least one tape for each movie ..., and each tape is always a copy of a single, specific movie"
    - "Not all of our movies have star actors"

- Implicit invariants common
  - Examples:
    - Tape cannot be loaned by more than one customer a time
    - Actor may be starring in more than one movie
Conceptual Design: Integrity constraints

- **Constraints:**
  - Attribute restrictions
  - Cardinality restrictions

- **Attribute restrictions**
  - Attribute contain at most one value: unstructured attribute restriction of E-R M.
  - Key attributes are unique
  - Attribute must / may have a value
    - e.g., movie must have title, director is not necessarily known
  - Value may be restricted
    - e.g., movies are made after 1900: movie.year > 1900
  - Special restrictions not in E-R model!
Conceptual Design: Integrity constraints

- **Cardinality restrictions**
  - # entities-instances in relationship (UML: multiplicity)

Examples: one movie on a tape, tape loaned by at most one customer at a time, rented tapes per customer $\geq 0$
Conceptual Design: Cardinalities

- **Simple Notation**
  
  "N tapes are rented by 1 customer"

- **Min-Max Notation**:

  "Each tape is rented by 0-1 customers"
  "Each customer has rented 0-N tapes"
Conceptual Design: Cardinalities

- **1:1 relationship**
  - person
    - have
    - driversLicence
    - (0,1) (1,1)

- **1:N relationship**
  - Movie
    - hold
    - Tape
    - (1,*) (1,1)
  - Lecture
    - hold
    - Lecturer
    - (1,1) (0,*)

- **N:M relationship**
  - User_account
    - has
    - emailMessage
    - (0,*) (1,*)

*Do not mix notations!*
Conceptual Design: Semantics N:M, (min, max)

Let \( R \subseteq E_1 \times E_2 \) be a relationship between entity sets \( E_1 \) and \( E_2 \)

- **R is 1:N** \( \iff \) \( R \) is a partial function \( R: E_2 \to E_1 \)
  \[ \iff \forall e_2 \in E_2: \left| \{ e_1 | e_1 \in E_1 \land (e_1,e_2) \in R \} \right| \leq 1 \]

- **R is 1:1** \( \iff \) \( E_2 \to E_1 \) is an injective partial function

- **R is M:N** \( \iff \) \( R \) is a relation, but not a function

**Important concept**

Let \( R \subseteq E_1 \times E_2 \) be a relationship between entity sets \( E_1 \) and \( E_2 \)

- \( (R,E_1) \) has \((min_1, max_1)\)
  \[ \iff \forall e_2 \in E_2: min_1 \leq \left| \{ y | y \in E_2 \land (x,y) \in R \} \right| \leq max_1 \]

- \( (R,E_2) \) has \((min_2, max_2)\)
  \[ \iff \forall e_1 \in E_1: min_2 \leq \left| \{ x | x \in E_1 \land (x,y) \in R \} \right| \leq max_2 \]
Conceptual Design: Example (without customers)

- **Tape**
  - id (1,1)
  - belong_to (0,*)
  - format (0,*)

- **Format**
  - name
  - charge

- **Movie**
  - title
  - category
  - year
  - director
  - id
  - Price_per_day
  - length
  - play (1,*)
  - (0,*)

- **Actor**
  - stage_name
  - real_name
  - birthday
  - id
  - charge
  - (1,*)

Relationships:
- Tape holds Movie (1,*)
- Tape belongs to Format (1,1)
- Movie plays Actor (1,*)

Without customers, the price per day is (1,1) and length is (0,*) (1,*).
### Conceptual Design: notational overview

<table>
<thead>
<tr>
<th></th>
<th>mandatory/ multiple</th>
<th>optional/ multiple</th>
<th>optional/ single</th>
<th>mandatory/ single</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E-RM (min,max)</strong></td>
<td>(1,*)</td>
<td>(0,*)</td>
<td>(0,1)</td>
<td>(1,1)</td>
</tr>
<tr>
<td><strong>E-RM 1:N</strong></td>
<td>N or M</td>
<td>N or M</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>UML</strong></td>
<td>1..*</td>
<td>0..*</td>
<td>0..1</td>
<td>1</td>
</tr>
</tbody>
</table>

k,j,N,M are natural numbers

**More notations in use, e.g., oracle-special notation...**
Conceptual Design: Weak entities

- Existentially dependent entities
- Not globally identifiable
- Example:

![Entity Relationship Diagram](image)

- Notation:

  \[
  \text{min}_2 = 1 : \text{at least one e1 for each e2, otherwise e2 could not be identified}
  \]
  \[
  \text{max}_2 = 1
  \]
Conceptual Design: Weak entities

- Example: Movies and their scenes

- Example: Order and Delivery
Conceptual Design: Weak entities

Modelling decision:

- **employee** \((0,\ast)\) **has** \((1,1)\) **child**

or

- **employee** \((0,\ast)\) **has** \((1,1)\) **child**

Advantages and disadvantages?
Conceptual Design: Temporal data

1. Condition: tape only rented by one customer at the same time:

   - Customer
   - Video tape
   - Rent
   - From
   - Until

   cid \rightarrow \text{customer} \rightarrow \text{rent} \rightarrow \text{video tape} \rightarrow \text{vid}

   \text{(0,*)} \rightarrow \text{(0,1)}

   from \rightarrow until

2. Condition: tape rented by many customers one after the other

   - Customer
   - Video tape
   - Has
   - Rental
   - Video tape
   - Jis in
   - Until
   - From

   cid \rightarrow \text{customer} \rightarrow \text{has} \rightarrow \text{rental} \rightarrow \text{video tape} \rightarrow \text{vid}

   \text{(0,*)} \rightarrow \text{(1,1)}

   \text{jis in} \rightarrow \text{(1,1)}

   from \rightarrow until
Conceptual Design: Example

![Conceptual Design Diagram]

- **Movie**
  - year
  - title
  - category
  - director
  - id
  - Price_per_day
  - length
  - (0,*)

- **Tape**
  - id
  - id
  - (0,*)
  - is_in
  - (1,1)

- **Format**
  - name
  - charge
  - (0,*)
  - belong_to
  - (1,1)

- **Rental**
  - from
  - (1,1)
  - until
  - (1,1)

- **Actor**
  - First_name
  - Last_name
  - Address
  - Telephone
  - Mem_no
  - stage_name
  - birthday
  - real_name
  - (1,*)

- **Customer**
  - have
  - (0,*)
Conceptual Design: n-ary relationships

- Example:
  - Lecturer recommends textbook for lecture

- Cardinalities:
  - A particular lecturer recommends zero or more "(textbook,lecture) pairs"
  - For each lecture at least one textbook is recommended by lecturers
  - Here, (min,max)-constraints not very expressive
Conceptual Design: n-ary relationships

- 1:N notation in n-ary relationships
  - Remember: R is 1:N ⇔ R is function R:E2 → E1

- Generalization: R: E₁ ×...× E_(j-1) × E_(j+1) ×...× Eₙ → Eⱼ
Conceptual Design: n-ary relationships (Example 2)

- Student is allowed to have only one topic for each Prof.
- Student is allowed to have only one Prof. for each topic
- Profs. are allowed to use the same topic again
- A topic can be used by many Profs., but for different students
N-ary relationships expressed by binaries

1. Connect pairs of entities

   ![Diagram showing connections between entities]

   - Less constraints expressible!

2. New “central” entity, binary relationships to old ones

   ![Diagram showing new central entity and connections]

   - Lecture
   - Lecturer
   - Lectureplan
   - Textbook
   - Lecture
Consider lecturing students:
- similar objects in different entities
- Redundancy ⟹ errors!

Generalization (specialization) helps
- DB Interpretation of Generalization differs from OO!
- instances of Student and Lecturer also instances of Person
- Student $\subseteq$ Person, Lecturer $\subseteq$ Person
Conceptual Design: Generalization

- **OO-interpretation:**
  - Instances of A, B and C different but share some attributes

- **DB interpretation:**
  - Instances of B and C also instances of A: $B \subseteq A$ and $C \subseteq A$
  - Key-inheritance
  - "is-a" different from ordinary relationships

- **Special cases:**
  - Disjoint specialization: Instances of B and C are disjoint
  - Complete specialization: $A = B \cup C$, no extra tuple in $A$
Conceptual Design: Short summary

- **E-R Modelling**
  - Standard semi-formal language for conceptual modelling
  - Different notations: we use E-R and (min,max)

- **Important terms**
  - **Entity (type, set):** set of objects with the same attributes
  - **Attribute:** property of an entity, at most one value per attribute.
  - **Relationship (type):** association between entity types
  - **Role:** name of one side of relationship
  - **Cardinality (multiplicity):** number of entity objects that participate in relationship
  - **Weak entity:** existentially dependent entity
  - **Generalization**