Part III

Entity-Relationship Model
Entity-Relationship Model

1. Database Models
2. ER Model
3. Further ER Model Concepts
Educational Objective for Today . . .

- Knowing the concepts of the entity-relationship model
- Ability to conceptually model an application domain
A database model is a system of concepts to describe databases. It defines the syntax and semantics of database descriptions for a database system.

- Database descriptions = database schemata
A Database Model Defines . . .

1. **Static properties**
   - 1. Objects
   - 2. Relationships

   including the primitive data types, which can describe data about the relations and objects,

2. **Dynamic properties** such as
   - 1. Operations
   - 2. Relationships between operations,

3. **Integrity constraints** of
   - 1. Objects
   - 2. Operations
Database Models

- Classical database models are especially suited for
  - Large amounts of data with a relatively static structure and
  - Describing static properties and integrity constraints
- Design models: (E)ER model, UML, . . .
- Realization models: relational model, object-oriented models, . . .
## Databases versus Programming Languages

<table>
<thead>
<tr>
<th>Database concept</th>
<th>Type system of a programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Database Model</strong></td>
<td></td>
</tr>
<tr>
<td><em>Relation, Attribute</em></td>
<td></td>
</tr>
<tr>
<td><strong>Database schema</strong></td>
<td></td>
</tr>
<tr>
<td><code>relation WINE = (...)</code></td>
<td></td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td></td>
</tr>
<tr>
<td><code>WINE(4961, 'Chardonnay', 'White', ...)</code></td>
<td></td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td></td>
</tr>
<tr>
<td>42, 'Cabernet Sauvignon'</td>
<td></td>
</tr>
<tr>
<td><strong>Type system</strong></td>
<td></td>
</tr>
<tr>
<td><code>int, struct ...</code></td>
<td></td>
</tr>
<tr>
<td><strong>Declaration of variable</strong></td>
<td></td>
</tr>
<tr>
<td><code>var x: int, y: struct Wine</code></td>
<td></td>
</tr>
</tbody>
</table>
# Levels of Abstraction

<table>
<thead>
<tr>
<th>Models</th>
<th>Data</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
<td>entity-relationship model</td>
<td>structograms</td>
</tr>
<tr>
<td>concrete</td>
<td>hierarchical model</td>
<td>Pascal</td>
</tr>
<tr>
<td></td>
<td>network model</td>
<td>C, C++</td>
</tr>
<tr>
<td></td>
<td>relational model</td>
<td>Java, C#</td>
</tr>
</tbody>
</table>
Overview of Database Models
Overview of Database Models /2

- HM: hierarchical model, NWM: network model, RM: relational model
- NF$^2$: model of nested (non-first-normal form = NF$^2$) relations, eNF$^2$: extended NF$^2$ model
- ER: entity-relationship model, SDM: semantic data models
- OODM / C++: object-oriented data models based on object-oriented programming languages, such as C++, OEM: object-oriented design models (e.g., UML), ORDM: object-relational data models
The ER Model

**Entity:** object of the real or a virtual world, about which information is to be stored, e.g., *Products* (wine, catalog), winemaker or critic; but also information about events, e.g., *Orders*

**Relationship:** describes a relationship between entities, e.g., a customer *orders* a wine or wine is being *offered* by a winemaker

**Attribute:** represents a property of an entity or a relationship, e.g., *Name* of customer, *Color* of a wine or *Date* of an order
Entity-Relationship Model

ER Example

- **Grape**
  - Name
  - Color
  - Proportion
  - Made of

- **Area**
  - Name
  - Country
  - Region
  - Located in

- **Wine**
  - Name
  - Color
  - Year
  - Res. Sugar
  - Made of
  - Produced by

- **Producer**
  - Name
  - Vineyard
  - Address
  - Has

- **Critic**
  - Name
  - Organization
  - Recommends

- **Dish**
  - Name
  - Side dish
  - Made of

- **License**
  - LicenseNo.
  - Amount

- **Organization**
  - Name

- **Vineyard**
  - Name

- **Address**
  - Name
Values

- **Values**: primitive elements of data, which can be represented directly.
- Value domains are described by datatypes, which, apart from the set of possible values, also characterize the basic operations on those values.
- ER model: pre-defined primitive datatypes, such as the integers `int`, the character sequences `string`, dates `date` etc.
- Every datatype represents a domain, including operations and predicates on values of this domain.
Entities

- **Entities** are the pieces of information to be represented in a database.
- In contrast to values, entities cannot be represented directly. They can only be observed through their properties.
- Entities are grouped according to their **entity types**, such as $E_1, E_2, \ldots$

Set of current entities:

$$\sigma(E_1) = \{e_1, e_2, \ldots, e_n\}$$
Attribute

- Attribute models properties of entities or relationships
- All entities of an entity type have the same kinds of properties; attributes are therefore declared for the entity type

Textual notation $E(A_1 : D_1, \ldots, A_m : D_m)$
Key-based Identification

- Key attributes: Subset of all attributes of an entity type
  \[ E(A_1, \ldots, A_m) \]
  \[ \{S_1, \ldots, S_k\} \subseteq \{A_1, \ldots, A_m\} \]

- In every state of the database, current values of the key attributes uniquely identify instances of the entity type \( E \)

- If multiple keys would be possible: Choice of a primary key

- Notation: Highlight by underlining:
  \[ E(\ldots, \underline{S_1}, \ldots, \underline{S_i}, \ldots) \]
Relationship Types

- Relationships between entities are grouped into relationship types.
- In general: arbitrary number $n \geq 2$ of entity types can participate in a relationship type.
- Every $n$-ary relationship type $R$ refers to $n$ entity types $E_1, \ldots, E_n$.
- Instances of a relationship type

$$\sigma(R) \subseteq \sigma(E_1) \times \sigma(E_2) \times \cdots \times \sigma(E_n)$$
Relationship Types /2

- **Notation**

  ![Diagram](attachment:diagram.png)

  - Textual notation: $R(E_1, E_2, \ldots, E_n)$
  - If an entity type participates in a relationship type multiple times: **roles** can be assigned

    married(Wife: Person, Husband: Person)
Relationship Attributes

- Relationships can also have attributes
- Attribute are declared at the relationship type; this also holds for the set of possible values \( \rightsquigarrow \) relationship attributes

Textual notation: \( R(E_1, \ldots, E_n; A_1, \ldots, A_k) \)
Characteristics of Relationships

- **Degree:**
  - Number of participating entity types
  - Often: binary
  - Example: *Supplier supplies Product*

- **Cardinality Constraints:**
  - Number of incoming instances of an entity type
  - Typical forms: 1:1, 1:n, m:n
  - Represent integrity constraints
  - Example: *maximum of 5 Products per Order*
Binary vs. N-ary Relationships

Diagram:

- Left side:
  - Dish
  - Wine
  - Recommends
  - Critic

- Right side:
  - Dish
  - Wine
  - D-W
  - D-C
  - C-W
  - Critic
Instances in the Example
Reconstruction of Instances

- $d_1 - c_1 - w_1$
- $d_1 - c_2 - w_2$
- $d_2 - c_2 - w_1$
- But also: $d_1 - c_2 - w_1$
1:1-Relationships

- Every $e_1$ of entity type $E_1$ is assigned to at most one entity $e_2$ out of $E_2$ and vice versa.
- Examples: *Brochure describes Product*, *Husband is married to Wife*.
1:N Relationships

- Every entity $e_1$ of entity type $E_1$ is assigned to an arbitrary number of entities $E_2$, but for every entity $e_2$, there is at most one $e_1$ in $E_1$
- Examples: *Supplier supplies Product*, *Mother has Children*
N:1 Relationship

- Inverse of 1:N, also **functional** relationship
- Binary relationships that define a **function**: Every entity of entity type $E_1$ is assigned to at most one entity of entity type $E_2$.

$$R : E_1 \rightarrow E_2$$

Wine \[\text{Produced By}\] Producer
1:1 Relationship

License \(\rightarrow\) Has \(\leftarrow\) Producer
M:N Relationships

- No restrictions
- Example: *Order consists of Products*
[min,max] Notation

- The notation [min, max] restricts the possible number of times an instance of an entity type can participate in a relationship by giving a minimum and a maximum value.
- Notation for expressing cardinalities in a relationship type:
  \[ R(E_1, \ldots, E_i[min_i, max_i], \ldots, E_n) \]
- Cardinality constraints: \[ min_i \leq |\{ r \mid r \in R \land r.E_i = e_i \}| \leq max_i \]
- Special notation for \( max_i \) is *
Expressing Cardinalities

- \([0, \ast]\) means “no restrictions” (default)
- \(R(E_1[0, 1], E_2)\) corresponds to a (partial) functional relationship \(R : E_1 \rightarrow E_2\), because every instance out of \(E_1\) is assigned to at most one instance out of \(E_2\)
- Total functional relationships are modelled by \(R(E_1[1, 1], E_2)\)
Expressing Cardinalities: Examples

- Partial functional relationship
  \[ \text{stored}\_\text{on}(\text{Product}[0,1],\text{Shelf}[0,3]) \]
  “Every product in the warehouse is stored on one shelf. However, products that are currently out of stock are not assigned to a shelf. At most three products can share the same shelf.”

- Total functional relationship
  \[ \text{supplies}(\text{Supplier}[0,*],\text{Product}[1,1]) \]
  “Every product is supplied by exactly one supplier. However, a supplier can very well supply more than one product.”
Alternative Ways to Express Cardinalities

Entity: Product
- [1,1] relationship with Deliver By

Entity: Supplier
- [0,*] relationship with Deliver By

1. One Product can be delivered by one Supplier.
2. Many Products can be delivered by many Suppliers.

Product Delivered By Supplier

N

1
Dependent Entity Types

- **Dependent Entity Type**: Identification through functional relationship

  - **Vintage Year** ∈ **Wine**
    - **Res. Sugar**
    - **Year**
  - **Belongs To**

- Dependent entities in the ER model: Functional relationship used as key
Dependent Entity Types /2

- Possible instantiations for dependent entities

- Year: 2004, Res. Sugar: 1,2
  - Name: Pinot Noir
    - Color: Red

- Year: 2003, Res. Sugar: 1,4
  - Name: Zinfandel
    - Color: Red

- Year: 1999, Res. Sugar: 6,7
  - Name: Riesling Reserve
    - Color: Weiß
Dependent Entity Types /3

- Alternative notation

Entity: Vintage Year
- Year
- Res. Sugar

Entity: Wine
- Name
- Color

Relationship: Belongs To
- 1
The IS-A Relationship

- Specialization/generalization relationship or IS-A relationship
- Textual notation: $E_1 \text{ IS-A } E_2$
- IS-A relationship semantically corresponds to an injective functional relationship
Properties of the IS-A Relationship

- Every sparkling wine instance is assigned to exactly one wine instance
  \( \sim \) sparkling wine instances are identified by their functional IS-A relationship
- Not every wine is a sparkling wine
- Attributes of the entity type Wine also apply to sparkling wines: “inherited” attributes
  \[
  \text{Sparkling\_wine(Name,Color,Production)} \]
  \[\text{of Wine}\]
- Not only attribute declarations are inherited, but also the current values of each instance
Instantiations of IS-A Relationship

![Diagram showing the relationship between Wine and Sparkling Wine]

Sparkling Wine

Wine
Alternative Notation for IS-A Relationship
Expressing Cardinalities: IS-A

- It holds for every relationship $E_1$ IS-A $E_2$ that: $\text{IS-A}(E_1[1, 1], E_2[0, 1])$
- Every instance of $E_1$ participates exactly once in the IS-A relationship, whereas instances of the supertype $E_2$ do not have to participate
- This does not affect aspects like attribute inheritance
Optionality of Attributes

Entity: Producer
- Vineyard
- Address

Relationship: Located In
- Country
- Region

Entity: Area
- Name
## Overview of Concepts

<table>
<thead>
<tr>
<th>Term</th>
<th>Informal Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>The piece of information to be represented</td>
</tr>
<tr>
<td>Entity type</td>
<td>Grouping of entities with the same properties</td>
</tr>
<tr>
<td>Relationship type</td>
<td>Grouping of relationships between entities</td>
</tr>
<tr>
<td>Attribute</td>
<td>Property value of an entity or a relationship</td>
</tr>
<tr>
<td>Key</td>
<td>Identifying property of an entity</td>
</tr>
<tr>
<td>Cardinalities</td>
<td>Restrict relationship types with regards to the number of times an entity can participate in a relationship</td>
</tr>
<tr>
<td>Degree</td>
<td>Number of entity types that participate in a relationship type</td>
</tr>
<tr>
<td>Functional relationship</td>
<td>Relationship Type with functional property</td>
</tr>
<tr>
<td>Dependent entities</td>
<td>Entities that cannot exist independently from other entities</td>
</tr>
<tr>
<td>IS-A relationship</td>
<td>Specialization of entity types</td>
</tr>
<tr>
<td>Optionality</td>
<td>Attribute or functional relationships as partial functions</td>
</tr>
</tbody>
</table>
Summary

- Database model, database schema, database (instance)
- Entity-relationship model
- Further concepts of the ER model
- Based on: chapter 3 in Datenbanken - Konzepte und Sprachen von Gunter Saake, Kai-Uwe Sattler und Andreas Heuer and chapter 7 in Fundamentals of Database Systems by Ramez Elmasri and Shamkant B. Navathe
Control Questions

- What defines a database model? What is the distinction between model and schema?
- Which concepts does the ER model define?
- Which properties characterize relationship types?
- How are dependent entity types different from regular entity types?