Entity-Relationship Data Model

SWEN304 / SWEN435

Trimester 1, 2024

Lecturer: Kevin Shedlock

Engineering and Computer Science

VICTORIA UNIVERSITY OF WELLINGTON TE HERENGA WAKA

Database Design

We will study:

1. Database Design Process
   - Requirements collection and analysis
   - Conceptual design
   - Logical design
   - Physical design
2. Entity-Relationship Model
   - Entities and relationships
   - Entity-Relationship diagram
3. Extensions of the Entity-Relationship Model
   - Specialisation
   - Aggregation and higher-order relationship types
   - Generalisation and clusters
4. MovieDB Examples

Database Design

- **Motivation:** According to the Project Management Institute:
  - 60% of all "IT" projects failed or were stopped before completion
  - more than 75% of all projects exceeded their budgets by 30%

- **Why do IT projects fail to meet customer expectations?**

- There can be many reasons:
  - the client is not sure what they want
  - the requirements were not properly documented
  - the lack of appropriate development methodology
  - the desired functionality was difficult to develop
  - the budget and resources were not sufficient
  - the lack of team and work management

Database Design – Overview

- **Database Design** is the process of constructing a **detailed schema of a database** that can support the organization's business needs and objectives for the database system under development

- The database design process has four phases:
  1. **Requirements Collection and Analysis**
  2. **Conceptual Design**
  3. **Logical Design**
  4. **Physical Design**

Database Design – Overview

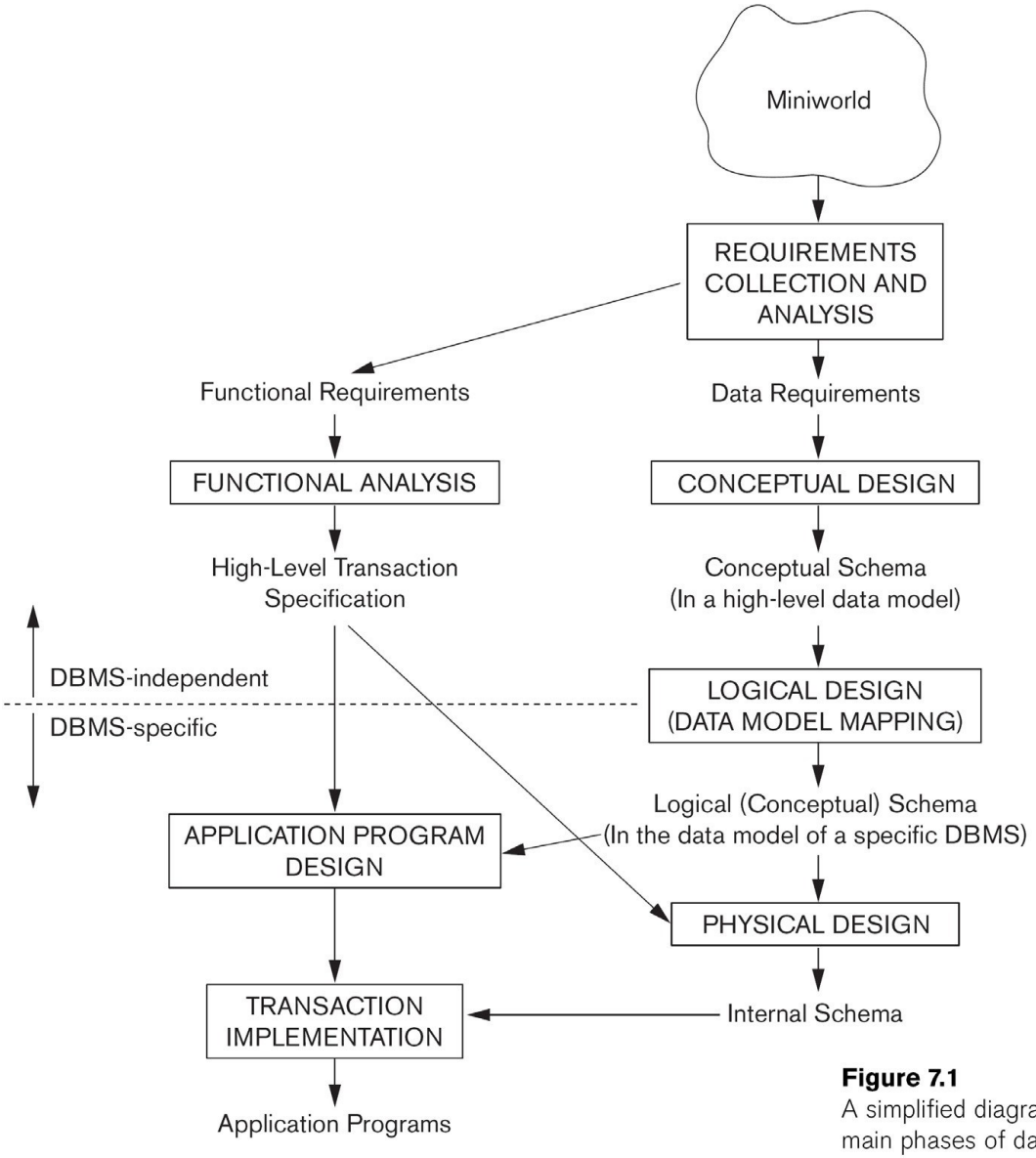


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

Phase 1: Requirements Collection and Analysis

- **Requirements collection and analysis** is the process of collecting and analysing data requirements of the organization so as to provide database solutions that fulfil business needs of the organization

- Compilation of data requirements includes:
  - a description of the data to be used or generated
  - details of how data is to be used or generated
  - any additional requirements for database system under development

Phase 2: Conceptual Design

- Conceptual design is the process of constructing a database schema used in an organization, independent of any physical or implementation considerations
  - modelling at a high-level of abstraction
  - simple enough
  - often with graphical representation
  - used to communicate the logical structure of a database with domain experts and potential users

- The data model is built using the input from the requirements specification

- **NB:** The conceptual design is based on the entity-relationship model in this course

Phase 3: Logical Design

- Logical design is the process of constructing a DB schema used in an organization, represented by a particular data manipulation technology, but still independent of physical considerations

- The conceptual DB schema is mapped onto a logical database schema, which can be further refined to meet the data requirements
  - popular data manipulation technologies used in logical design are those provided by the relational or another model (e.g. OO, XML, JSON)
  - in the relational model, the database schema includes a description of all tables with their primary and foreign keys, so that SQL can be applied for data definition, data manipulation, and data querying

- **NB**: logical design is based on the relational model in this course

Phase 4: Physical Design

- Physical design is the process of implementing the logical data model in a DBMS

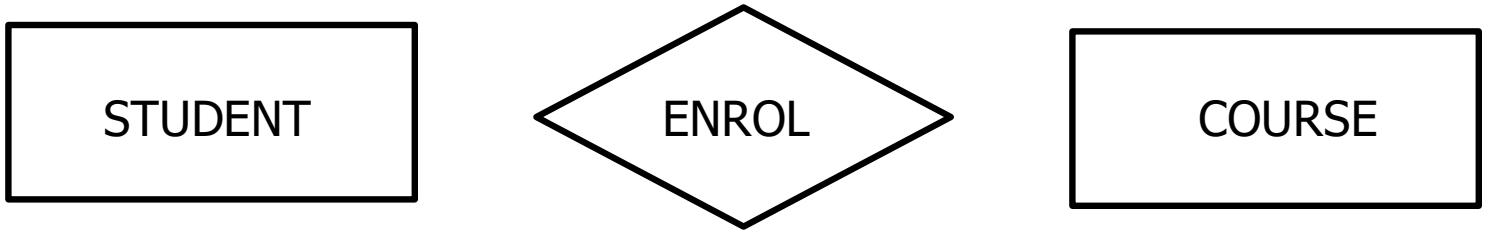
- For a relational model, the physical design must establish relations in the DBMS that involves:
  - selecting the files in which to store the tables
  - deciding which indexes should be used to achieve efficient access to data items
  - describing the integrity constraints and security measures

- Physical design decisions often affect other properties of the DB at run time, such as performance, accessibility, security and user-friendliness

- NB: Physical design details are beyond the scope of this course

Entity-Relationship Model – Terminology

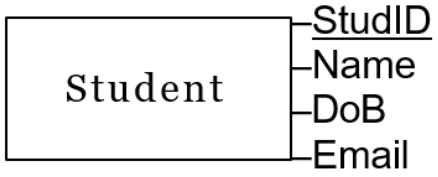
- The Entity-Relationship model (ER) is the most popular conceptual data model (de facto standard)
- Originally proposed by Peter P. Chen
- The target of the database is regarded as consisting of entities and relationships
- Depicted below are two related entities for the TTUPA being STUDENT and COURSE with the relationship being ENROLE.



Entity-Relationship Model – Terminology

- Entities are basic objects that can be identified in the DB
- Attributes are properties that describe entities
- Entities described by the same set of attributes are classified into an Entity set
- An entity set can be abstracted into an entity type
- Entities are best described as nouns. A word used to identify people, places, or things.

A entity type Student with attribute set {StudID, Name, DoB, Email}, and primary key {StudID}



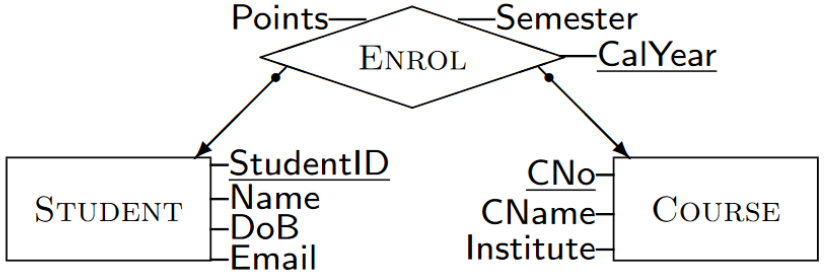
Entity-Relationship Model – Terminology

- Relationships are associations between entities/objects that describe the connections to each other.
- Relationships have entities as their components
- Relationships can have attributes that describe them
- Relationships that are described by the same set of components & attributes form a relationship set
- Below is the relationship **Enrol** Entity with the **Date** Attribute

Entity-Relationship Model – Terminology

- Relationships are associations between entities/objects
- Relationships have entities as their components
- Relationships can have attributes that describe them
- Relationships that are described by the same set of components & attributes form a relationship set
- A relationship set can be abstracted into a relationships type

Example: A relationship type ENROL with component set {STUDENT, COURSE}, attribute set {CalYear, Semester, Points}, and primary key {STUDENT, COURSE, CalYear}

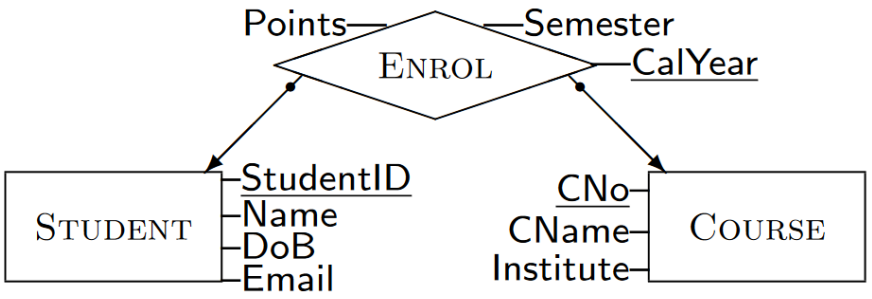


Entity-Relationship Diagram

- For communication between all types of uses, it is helpful to visualise the database schema *graphically*
- A entity-relationship (ER) diagram is a directed graph with a node for every entity or relationship type
- Convention (Chen): draw entity types as rectangles, and draw relationship types as diamonds
- To capture more details of the conceptual design, database designers often show...
  - attributes in the ER diagram: attached to entity/relationship type
  - primary keys in the ER diagram: by underlining key attributes and by putting dots • on key components
  - attribute domains in the ER diagram: added to the attributes

ER Diagram, ER Schema and State – More Definitions

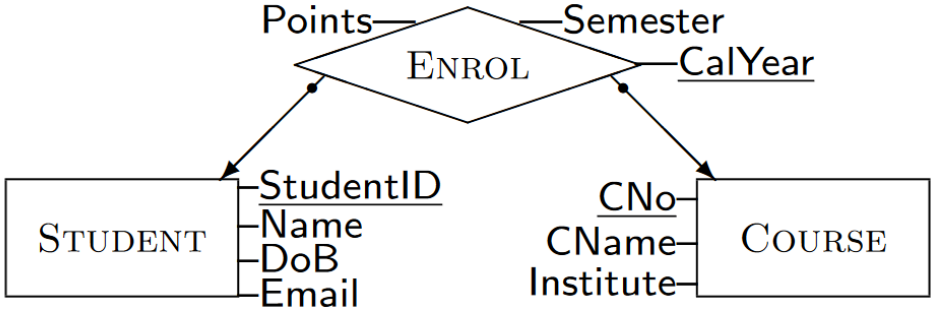
- Our example of an ER diagram:



- Recall: In this ER diagram, we have the entity types Student and Course, and the relationship type Enrol
- All the entity types and relationship types in an ER diagram together form an **ER schema**
- An **ER schema** Is a finite set *S* of entity and relationship types, such that for every relationship type R in *S* all its components belong to *S*, too.

ER Diagram, ER Schema and State – More Definitions

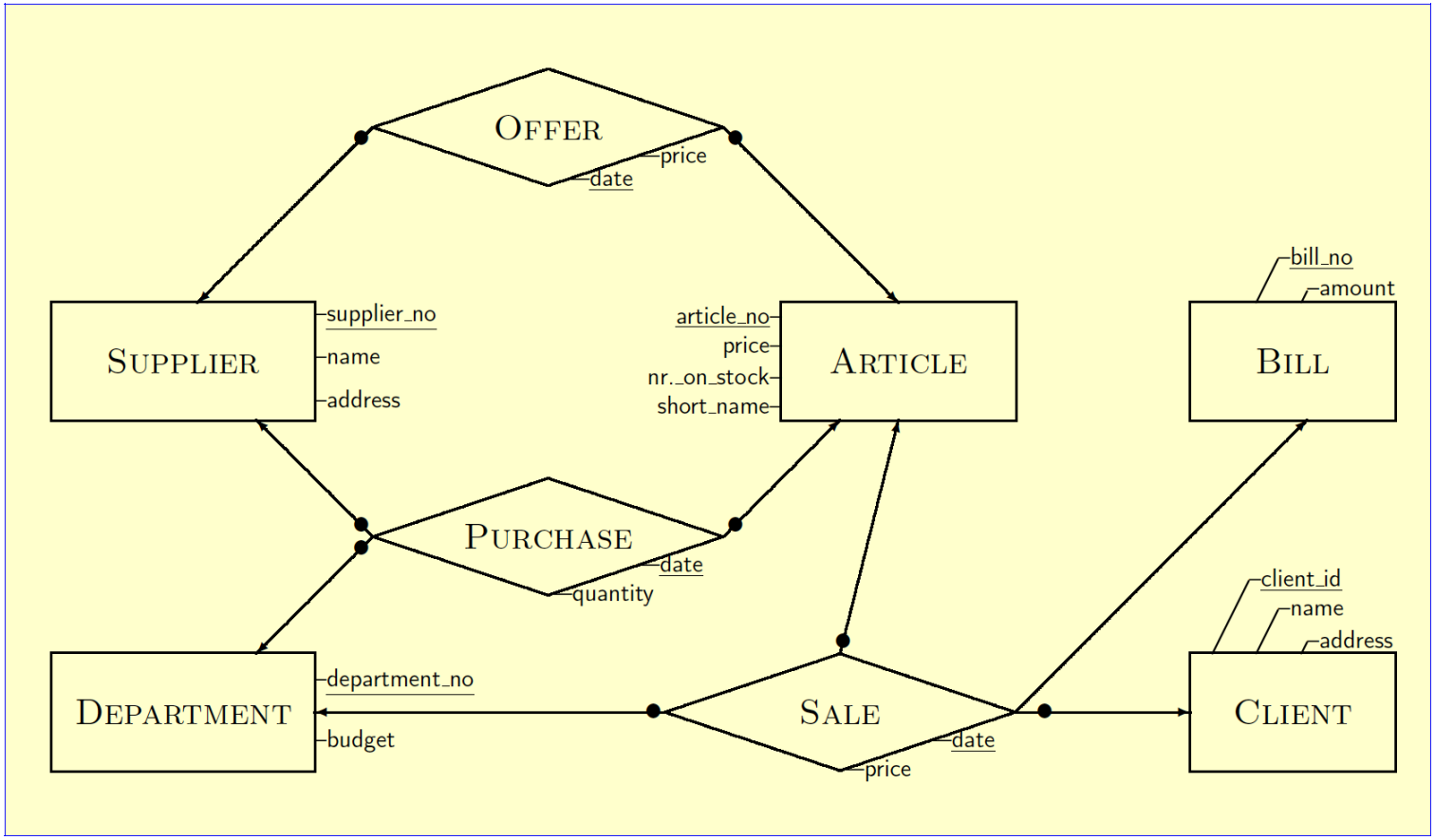
- Our example of an ER diagram:



- A **state** or **instance** S^t of an ER schema S assigns:
  - each entity type E an entity set E^t
  - each relationship type R a relationship set R^t
  - such that for each relationship r in the relationship set R^t and for every component E of R the entity $r(E)$ is an element of the entity set E^t.

How to read an Entity-Relationship Diagram?

- **Which entity types and relationship types can be found in this example?**



Entity Types in an ER Diagram – Example

In our ER diagram example we have the following entity types:

- Supplier
  - *attributes:* supplier no, name, address
  - *primary key:* supplier no
- Article
  - *attributes:* article no, price, nr on stock, short name
  - *primary key:* article no
- Bill
  - *attributes:* bill no, amount
  - *primary key:* bill no
- Department
  - *attributes:* department no, budget
  - *primary key:* department no
- Client
  - *attributes:* client id, name, address
  - *primary key:* client id

Relationship Types in an ER Diagram – Example

In our ER diagram we have the following relationship types:

- Offer
  - *components:* Supplier, Article
  - *attributes:* date, price
  - *primary key:* Supplier, Article, date
- Purchase
  - *components:* Supplier, Article, Department
  - *attributes:* date, quantity
  - *primary key:* Supplier, Article, Department, date
- Sale
  - *components:* Department, Article, Client, Bill
  - *attributes:* date, price
  - *primary key:* Department, Article, Client, date

Keys in an ER Diagram – Example

- **Primary keys** ensure that the entities in an entity set (or the relationships in a relationship set) can be **uniquely identified**

- In our example the entity types have primary keys as follows:
  - clients are uniquely identified by their client id
  - departments are uniquely identified by their department no
  - bills are uniquely identified by their bill no
  - articles are uniquely identified by their article no
  - suppliers are uniquely identified by their supplier no

- In our example, the relationship types have primary keys as follows:
  - offers are uniquely identified by their supplier, article and date
  - purchases are uniquely identified by their department, supplier, article, date
  - sales are uniquely identified by their department, article, client and date

Components of Relationship Types

- An entity type E is described by its attribute $attr(E)$ and its primary key $key(E)$
- Note: an entity type has no components

- Relationship *R* is described by its components $comp(R)$, attributes $attr(R)$ and primary key $key(R)$

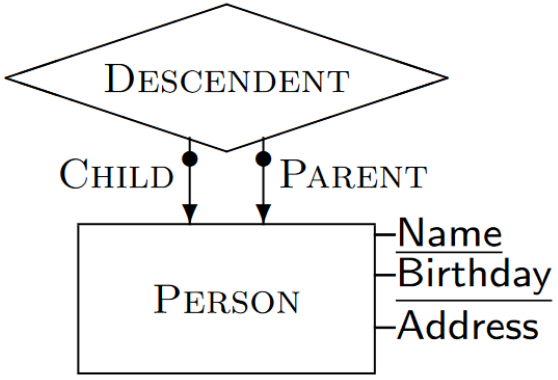
- A relationship type with n components is called *n*-ary:
  - Unary relationship type has 1 component
  - Binary has 2
  - Ternary has 3, …

Recursive Relationship Types and Roles

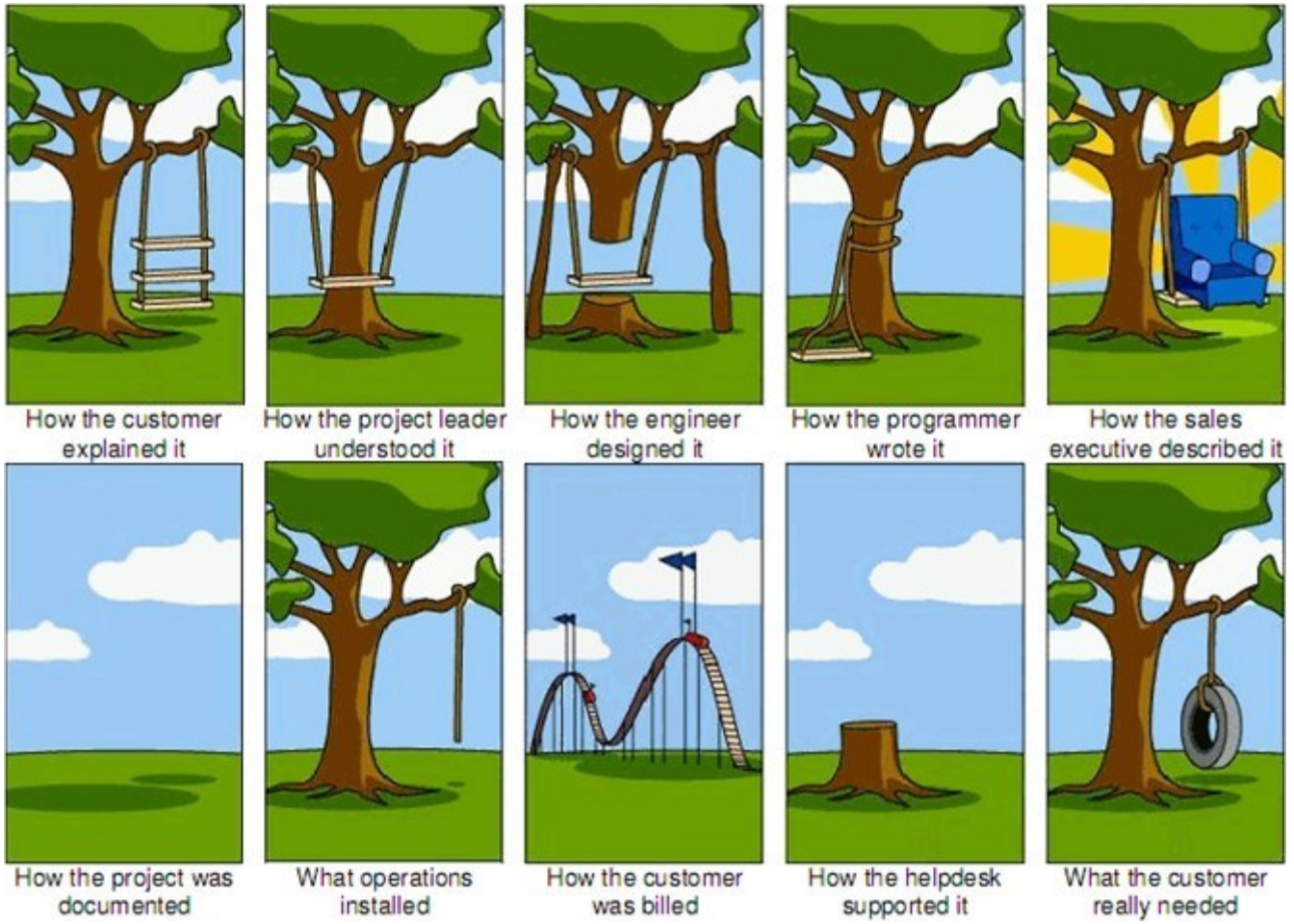
- A relationship type is **recursive** if it contains the same entity type more than once as a component

- **E.g.** suppose we want to model relationships between a child and a parent
- Use a relationship type Descendent that contains the entity type Person twice as a component

- **Roles** are used to distinguish the different occurrences of the same entity type in a recursive relationship type



Why is Database Design a Challenge?



Database Requirements Complexity

- Understanding Requirements - a thorough understanding of the organization's requirements, including data needs, business rules, and processes.

- Gathering and interpreting the requirements accurately can be challenging, especially when dealing with stakeholders with diverse perspectives and priorities.

Database Design Complexity

- Database design is a multi-disciplinary task that requires **consideration between the trade-offs,** and constraints matching technical expertise, domain knowledge, and effective communication skills.

- Data Modelling Complexity – **translating real-world entities** and relationships accurately into a structured database schema that makes design decisions can be complex.

Database Security Complexity

- Security and Privacy - the **security** and **privacy of data** is critical in database design. Design decisions must consider factors such as access control, encryption, and compliance with regulatory requirements i.e. balancing **security requirements** with **usability** and **performance** can be challenging.