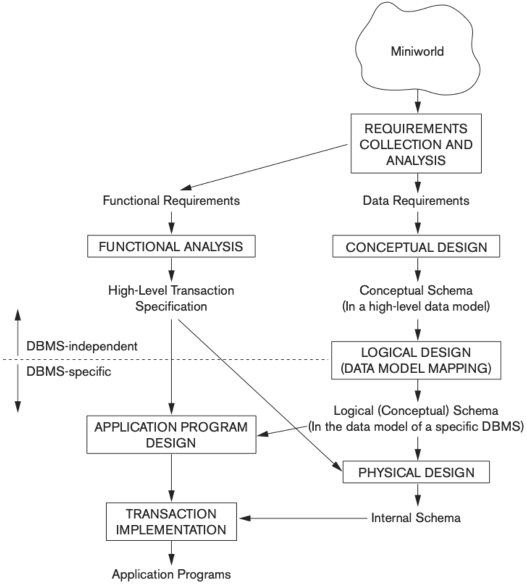
COS221
L03 - Entity-Relationship Conceptual Modelling (Part 1)

(Chapter 7 in Edition 6 and Chapter 3 in Edition 7)

Linda Marshall

24 February 2023

High-level Conceptual Data Models For Database Design - The Phases



Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

High-level Conceptual Data Models For Database Design - The Phases

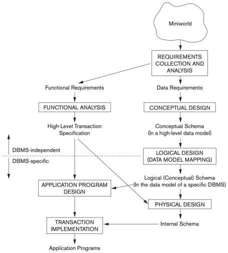

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Requirements Collection and Analysis

- Prospective database users are interviewed by designers.
- The functional and data requirements for the system are documented. These two requirements are refined to produce the the applications and physical database design which the applications will manipulate.

High-level Conceptual Data Models For Database Design - The Phases

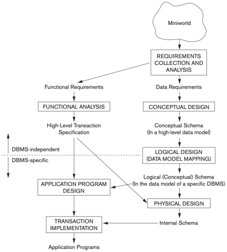

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Functional Requirements

- User-defined operations (or transactions) are used to retrieve and update data in the database
- In COS301, use cases, activity diagrams, data flow diagrams, sequence diagrams, scenarios etc. will be used to capture functional requirements

High-level Conceptual Data Models For Database Design - The Phases

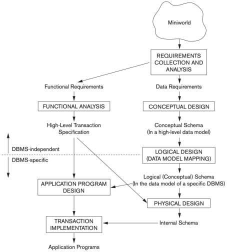
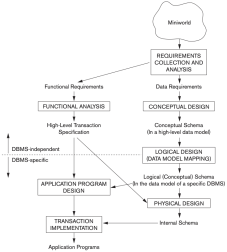

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Conceptual Design

- After collection and analysis of the data requirements, a conceptual schema for the database is created

High-level Conceptual Data Models For Database Design - The Phases



Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Logical Design (Data Model Mapping)

- Translation of the conceptual schema to the specific DBMS model takes place (in our case, the relational model)
- High-level data transformed to the implementation data model

High-level Conceptual Data Models For Database Design - The Phases

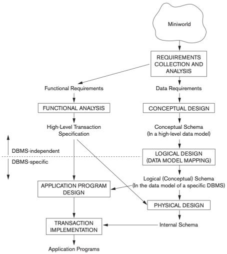

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

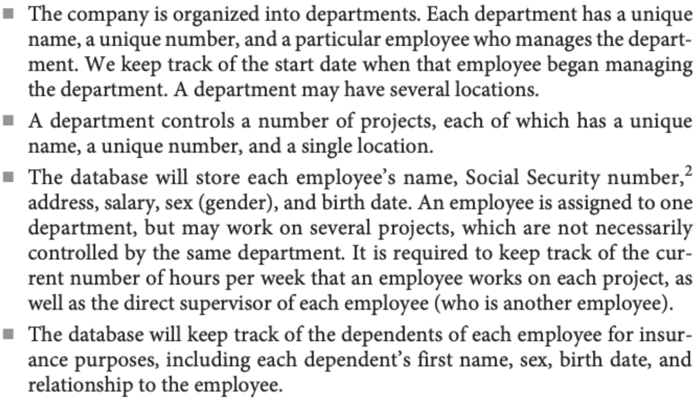
Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Physical Design

- The internal storage structures, file organisations, indexes, access paths etc. are specified

Sample Database Application - The COMPANY Database

Verbal description



- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee's name, Social Security number,[2] address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

Sample Database Application - The COMPANY Database

So what can we make out from the description?

Sample Database Application - The COMPANY Database

ER Representation

ER diagram describes data as:

- *entities*
- entities have *attributes*
- entities have *relationships* with other entities

In this lecture we will concentrate on the entities and their attributes.

Entities and their Attributes

- An *entity* is the basic object that is represented. It may be an object with:
  - *physical existence* - something particular, such as a person, car etc.
  - *conceptual existence* - for example a University, Company etc.
- Each entity has *attributes*
  - Attributes are properties that describe the entity. Values will be assigned to the attributes to define/identify a particular entity

Entities and their Attributes

Attribute types

- simple (atomic) vs. composite
- single-valued vs. multivalued
- stored vs. derived
- NULL values
- Complex
- Key

Entities and their Attributes

Attribute types

- simple (atomic) vs. composite
  - Simple attributes are not divisible.
  - Composite attributes can be divided into smaller parts. They can be referred to as a unit or their constituent parts can be referenced.

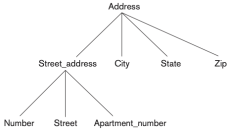


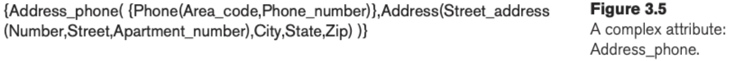
Figure 3.4
A hierarchy of composite attributes.

- single-valued vs. multivalued
  - Most attributes have a single value for a particular entity.
  - Where one value does not suffice, an attribute may be represented by a set of values. For example, a person may have more than one university degree. These attributes may be bounded to constrain the number of values.

Entities and their Attributes

Attribute types, cont.

- stored vs. derived
  - Derived attributes make use of stored attributes to calculate (derive) a value. Examples of derived attributes include *age* which can be derived from the data of birth, the number of students in a class which can be derived from counting the number of students registered in a course, ...
- NULL values
  - An entity without an applicable value is assigned a NULL value
- Complex
  - Consist of composite and/or multivalued attributes. Composite attributes are represented using round braces (...), while multivalued attributes are represented using curly braces { ... }



{Address_phone({Phone(Area_code,Phone_number)},Address(Street_address (Number,Street,Apartment_number),City,State,Zip))}

Figure 3.5
A complex attribute: Address_phone.

A person has multiple addresses. At each address there may be multiple phones. An address is a nested composite and the phones at the addresses are in turn composites.

Entities and their Attributes

Attribute types, cont.

- Key
  - Identifies each entity uniquely.
  - Keys may be composite.
  - If more than one attribute of an entity type is a key attribute, then they are so independently of each other.
  - An entity type with no keys is a *weak entity type*

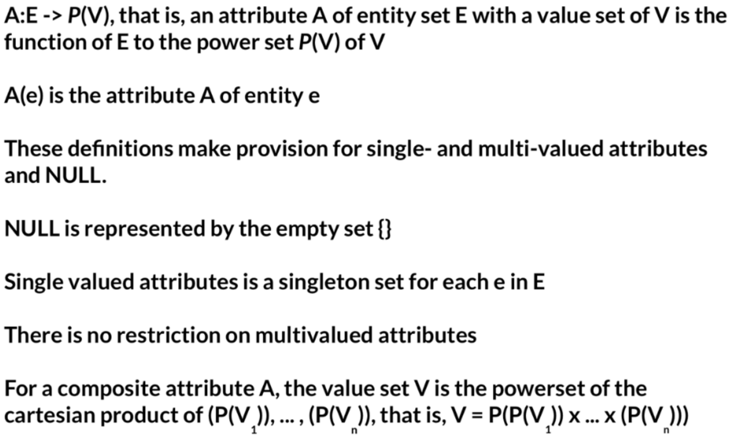
Entities and their Attributes

Attribute value sets

- Attributes have *value sets* (or domains).
- That is a set of all values that may be assigned to that attribute.
- Value sets are typically not represented in ER-diagrams.

Entities and their Attributes

Attribute value sets - mathematically



A:E -> *P*(V), that is, an attribute A of entity set E with a value set of V is the function of E to the power set *P*(V) of V

A(e) is the attribute A of entity e

These definitions make provision for single- and multi-valued attributes and NULL.

NULL is represented by the empty set {}

Single valued attributes is a singleton set for each e in E

There is no restriction on multivalued attributes

For a composite attribute A, the value set V is the powerset of the cartesian product of (P(V_1)), ... , (P(V_n)), that is, V = P(P(V_1)) x ... x (P(V_n)))

Entity types and Entity sets

- A database comprises of groups of similar entities
- An *entity type* defines a set/collection of entities with the same attributes. It is described by an entity name and its attributes.
- An *entity set* is a set of all entities belonging to an entity type.

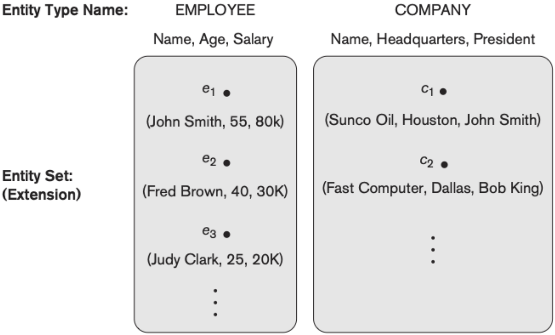
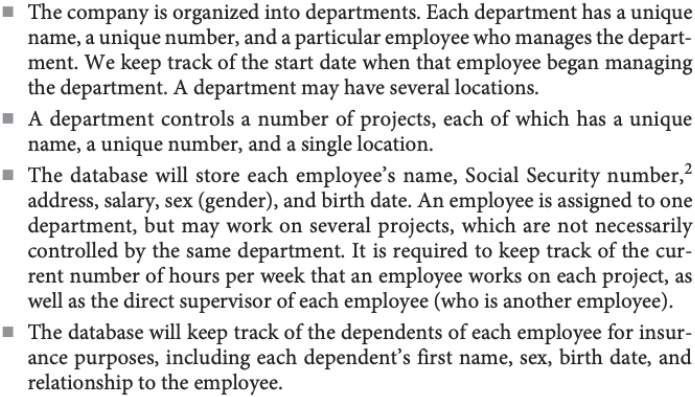


Figure 3.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

Sample Database Application - The COMPANY Database

Verbal description revisited



- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee's name, Social Security number,[2] address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

Sample Database Application - The COMPANY Database

Preliminary Entities and attributes

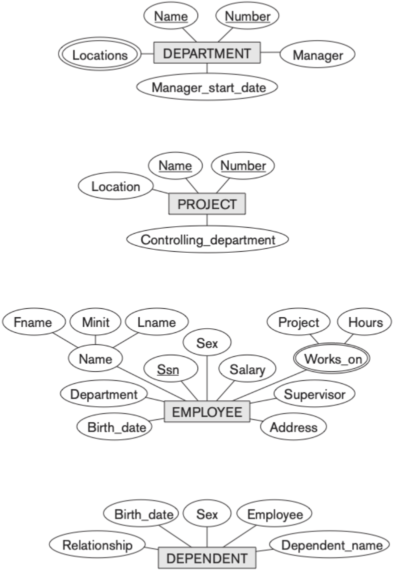


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

In the next lecture we will consider the relationships between entities and model those.