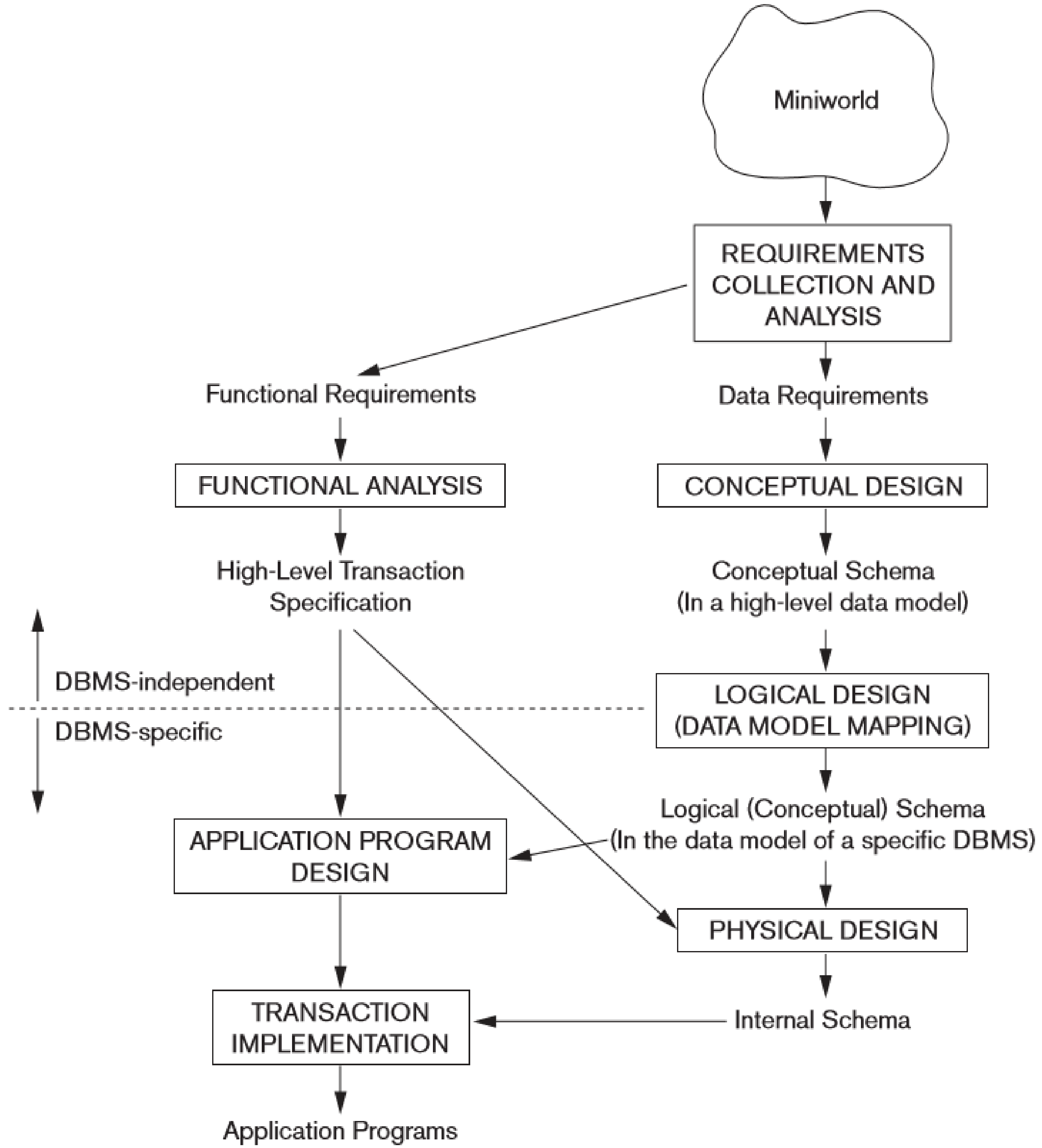
Data Modeling Using the Entity-Relationship (ER) Model

Part 1

Using High-Level Conceptual Data Models for Database Design



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

A Sample Database Application

- 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.

- We store each employee’s name, Social Security number, 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.
- We keep track of the current number of hours per week that an employee works on each project.
- We also keep track of the direct supervisor of each employee (who is another employee).

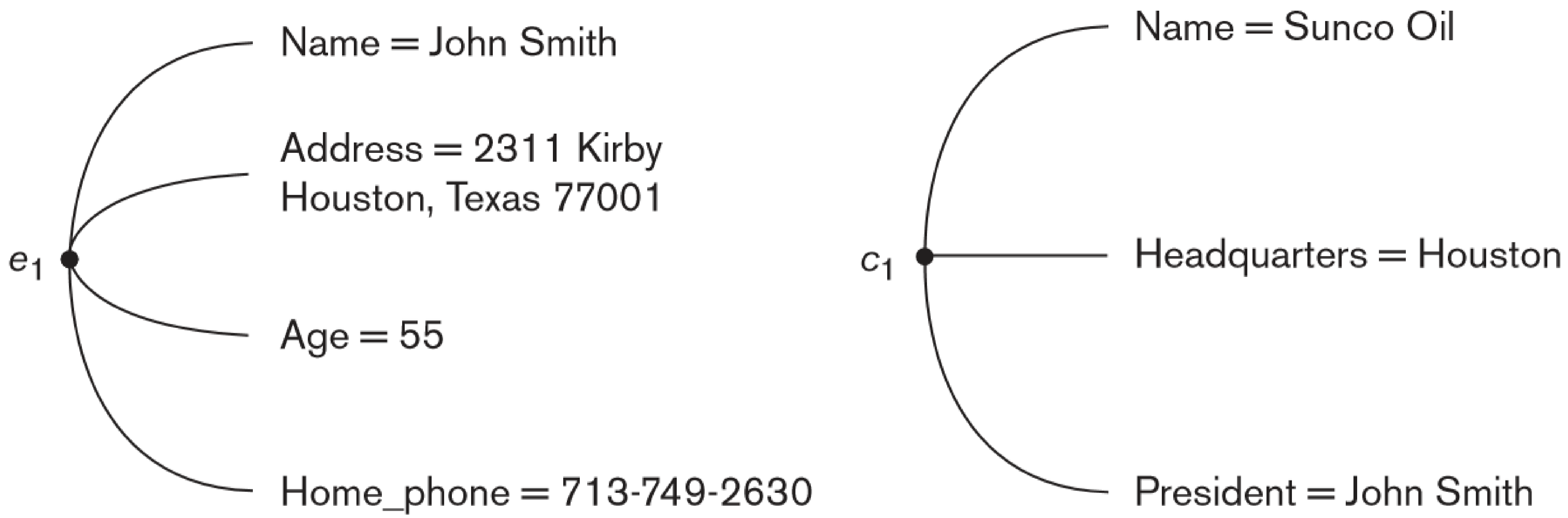
- We want to keep track of the dependents of each employee for insurance purposes.
- We keep each dependent’s first name, sex, birth date, and relationship to the employee.

Entity Types, Entity Sets, Attributes, and Keys

Entities and Attributes

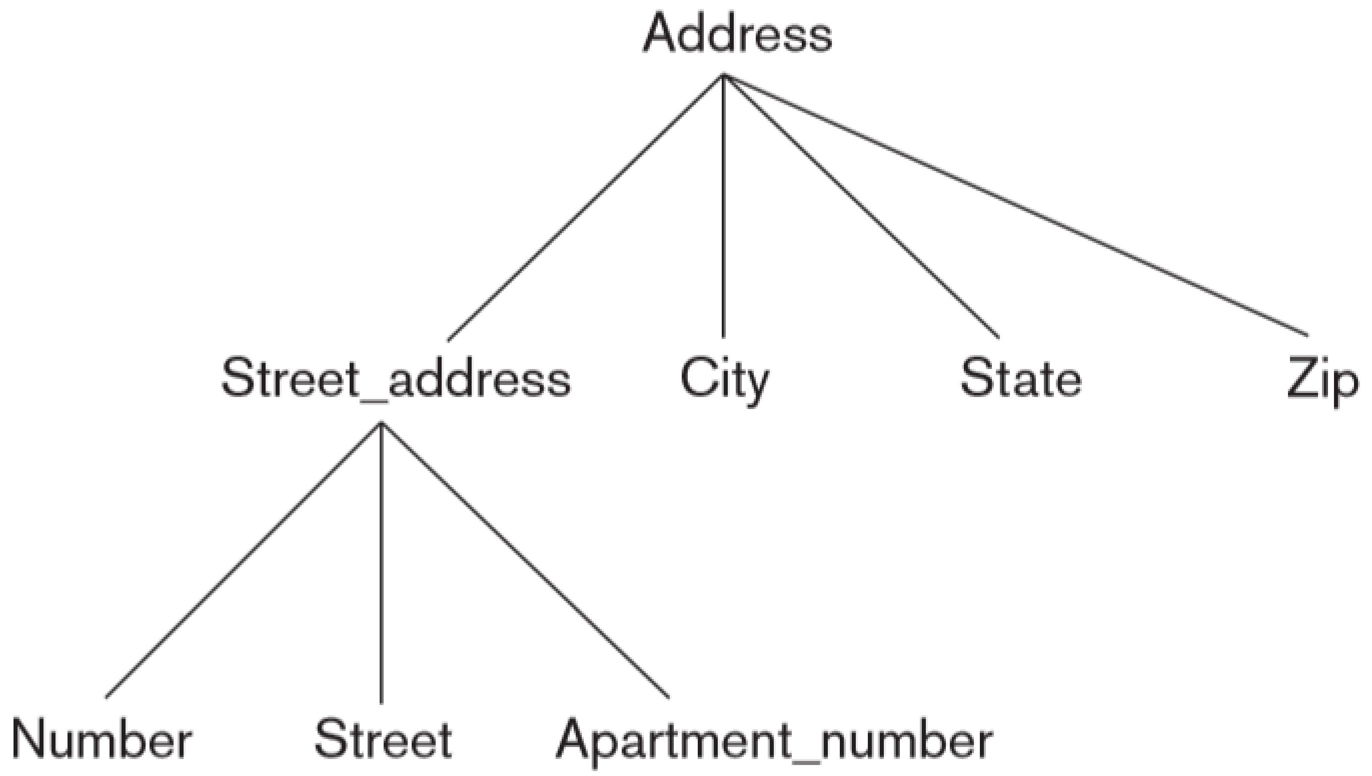
- The basic object that the ER model represents is an entity, which is a *thing* in the real world with an independent existence.
- An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).

- Each entity has attributes—the particular properties that describe it.
- For example, an `EMPLOYEE` entity may be described by the employee’s name, age, address, salary, and job.


e_1
Name = John Smith
Address = 2311 Kirby
Houston, Texas 77001
Age = 55
Home_phone = 713-749-2630
c_1
Name = Sunco Oil
Headquarters = Houston
President = John Smith

- Several types of attributes occur in the ER model: *simple* versus *composite*, *single-valued* versus *multivalued*, and *stored* versus *derived*.

- **Composite attributes** can be divided into smaller subparts, which represent more basic attributes with independent meanings.
- Attributes that are not divisible are called **simple** or **atomic attributes**.


Address
Street_address
City
State
Zip
Number
Street
Apartment_number

- Composite attributes are useful to model situations in which a user sometimes refers to the composite attribute as a unit but at other times refers specifically to its components.
- If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes.

- Most attributes have a single value for a particular entity; such attributes are called **single-valued**.
- Some attributes may have multiple values; such attributes are called **multivalued**.
- A multivalued attribute may have lower and upper bounds to constrain the *number of values* allowed for each individual entity.

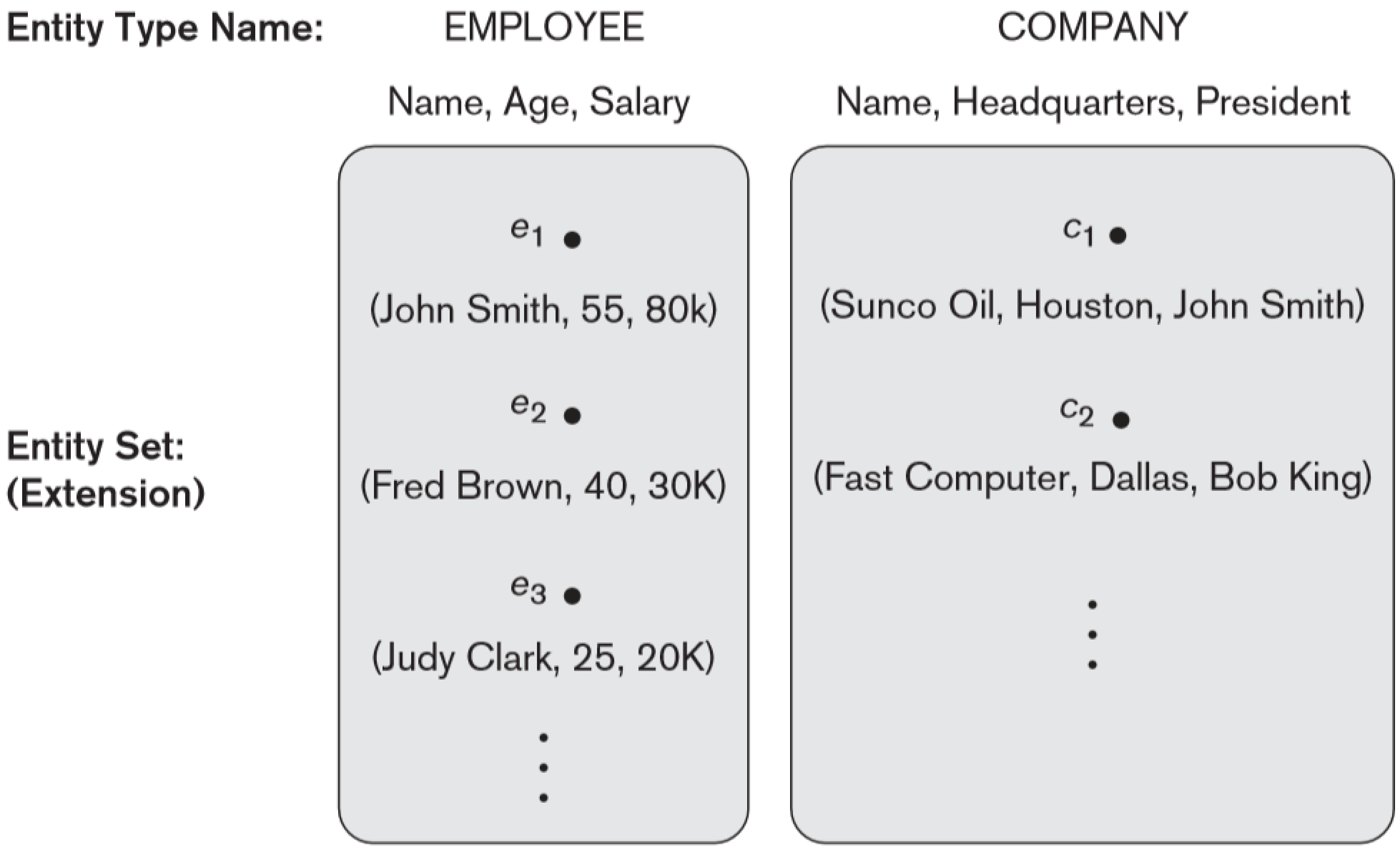
- In some cases, two (or more) attribute values are related—for example, the `Age` and `Birth_date` attributes of a person.
- For a particular person entity, the value of `Age` can be determined from the current (today’s) date and the value of that person’s `Birth_date`.
- The `Age` attribute is hence called a **derived attribute** and is said to be **derivable from** the `Birth_date` attribute, which is called a **stored attribute**.
- Some attribute values can be derived from related entities; for example, an attribute `Number_of_employees` of a `DEPARTMENT` entity can be derived by counting the number of employees related to (working for) that department.

- In general, composite and multivalued attributes can be nested arbitrarily.
- We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces { }.
- Such attributes are called **complex attributes**.

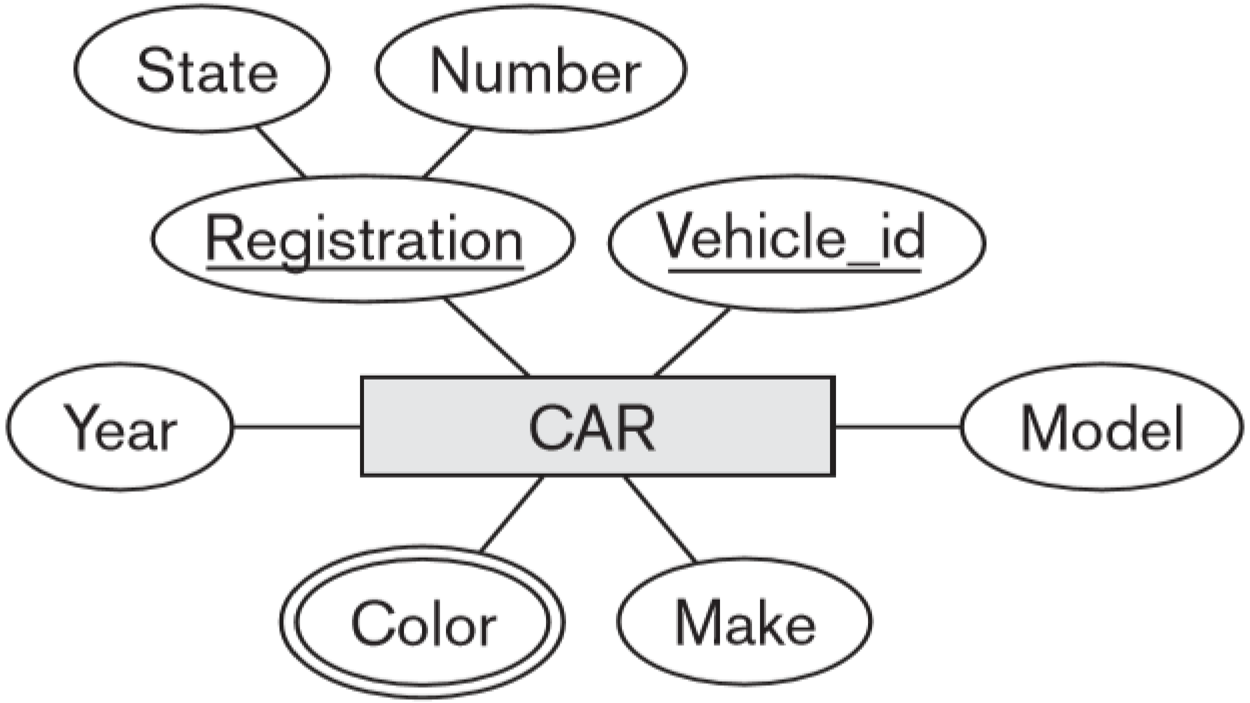
{Address_phone({Phone(Area_code,Phone_number)},Address(Street_address (Number,Street,Apartment_number),City,State,Zip))}

Entity Types, Entity Sets, Keys, and Value Sets

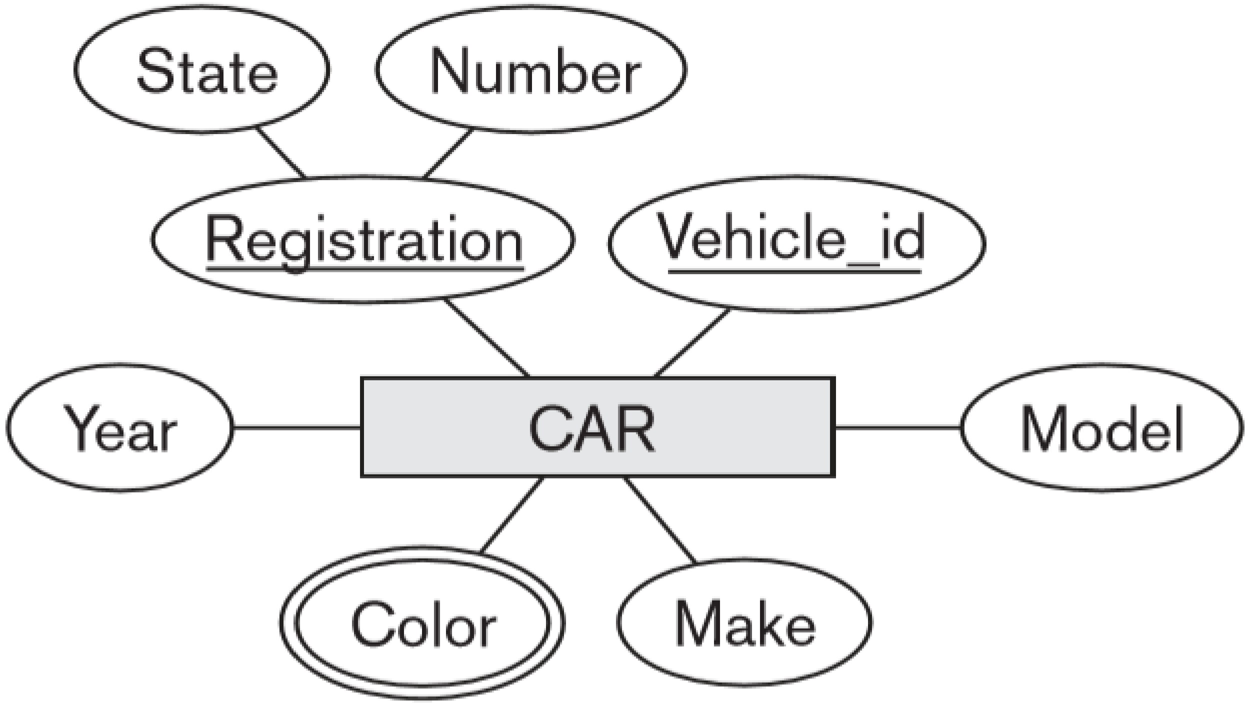
- An **entity type** defines a *collection* (or *set*) of entities that have the same attributes.
- Each entity type in the database is described by its name and attributes.
- The collection of all entities of a particular entity type in the database at any point in time is called an **entity set**; the entity set is usually referred to using the same name as the entity type.


Entity Type Name:
EMPLOYEE
COMPANY
Name, Age, Salary
Name, Headquarters, President
Entity Set:
(Extension)
e_1
(John Smith, 55, 80k)
e_2
(Fred Brown, 40, 30K)
e_3
(Judy Clark, 25, 20K)
c_1
(Sunco Oil, Houston, John Smith)
c_2
(Fast Computer, Dallas, Bob King)

- An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.
- Attribute names are enclosed in ovals and are attached to their entity type by straight lines.
- Composite attributes are attached to their component attributes by straight lines.
- Multivalued attributes are displayed in double ovals.


State
Number
Registration
Vehicle_id
Year
CAR
Model
Color
Make

- An important constraint on the entities of an entity type is the **key** or **uniqueness constraint** on attributes.
- An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set.
- Such an attribute is called a **key attribute**, and its values can be used to identify each entity uniquely.
- Sometimes several attributes together form a key, meaning that the *combination* of the attribute values must be distinct for each entity.
- If a set of attributes possesses this property, the proper way to represent this in the ER model that we describe here is to define a *composite attribute* and designate it as a key attribute of the entity type.
- Notice that such a composite key must be *minimal*; that is, all component attributes must be included in the composite attribute to have the uniqueness property.
- In ER diagrammatic notation, each key attribute has its name **underlined** inside the oval.


State
Number
Registration
Vehicle_id
Year
CAR
Model
Color
Make

- An entity type may also have *no key*, in which case it is called a *weak entity type*.

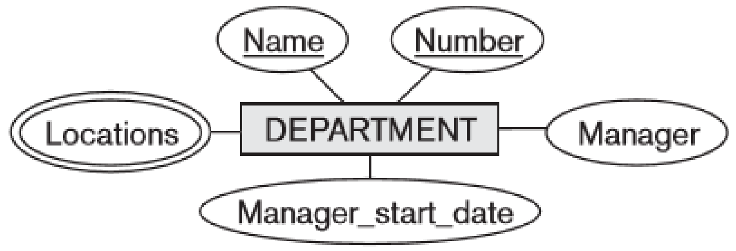
- In our diagrammatic notation, if two attributes are underlined separately, then *each is a key on its own*.
- Unlike the relational model, there is no concept of primary key in the ER model that we present here; the primary key will be chosen during mapping to a relational schema.

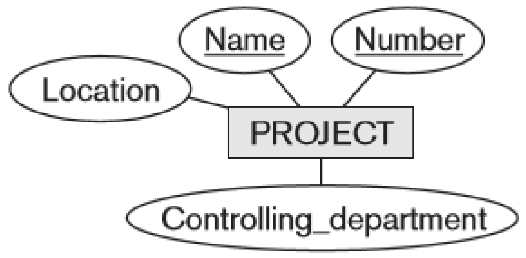
- Each simple attribute of an entity type is associated with a **value set** (or **domain** of values), which specifies the set of values that may be assigned to that attribute for each individual entity.
- Value sets are not displayed in ER diagrams.

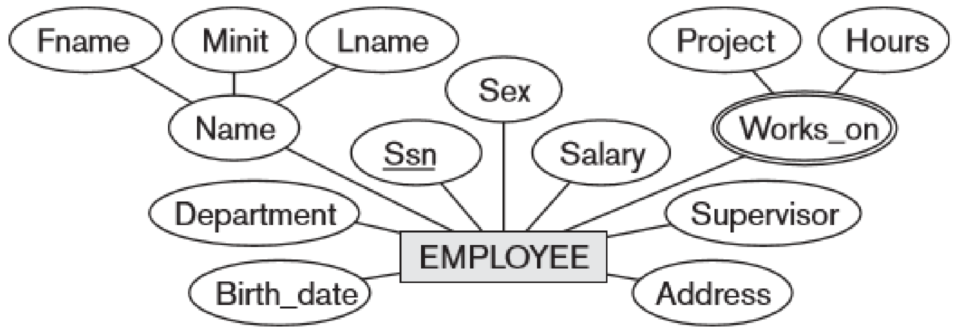
- Mathematically, an attribute A of entity set E whose value set is V can be defined as a **function** from E to the power set $P(V)$ of V:

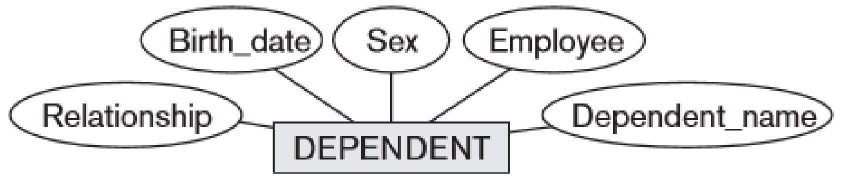
$$A : E \rightarrow P(V)$$

Initial Conceptual Design of the COMPANY Database


Name
Number
Locations
DEPARTMENT
Manager
Manager_start_date


Name
Number
Location
PROJECT
Controlling_department


Fname
Minit
Lname
Project
Hours
Sex
Name
Works_on
Ssn
Salary
Department
Supervisor
EMPLOYEE
Birth_date
Address


Birth_date
Sex
Employee
Relationship
Dependent_name
DEPENDENT