02. Data Modeling

Definition

- Data Model = a collection of concepts that can be used to describe the structure of a database
  - Structure = data types, relationships, constraints that should hold for that data (sometimes + a set of basic operations)
- Data Models:
  - High-level/Conceptual Data Models
  - Low-level/Physical Data Models
  - Representational/Implementational Data Models
  - Object Data Models

Data Modeling Importance

- To facilitate interaction between designer, applications programmer, end-user … even facilitate better understanding of the organization:
  - Standardize the organization's view of data
  - Allow designers to understand nature, role and scope of data
- Fact: data is viewed differently by different people e.g.
  - Clerk's view vs Manager's view.
  - Manager 1's view vs Manager 2's view
  - Applications Programmer's view vs End-User's view
  - Reflect the 'blind people and the elephant analogy'
- Important that a good blueprint be available … so it won't matter that the different people have different views

Example ABC Company Database

Requirements: (simplified)

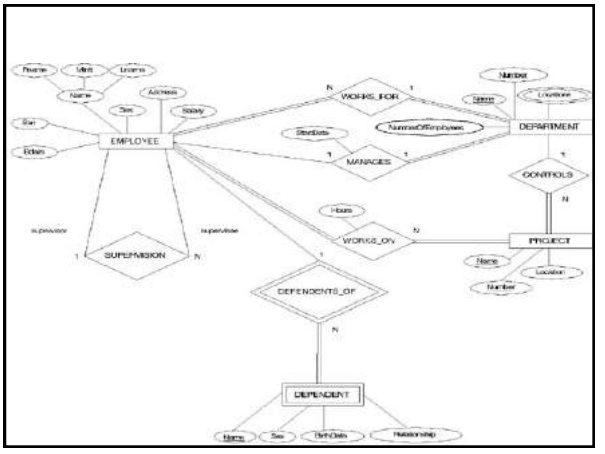
- ABC Company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager.
- Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.

Example ABC Company Database

- For each EMPLOYEE we store:
  - PIN (or SSN)
  - Address
  - Salary
  - Gender
  - Birthdate.
- Each employee *works for* one department but may *work on* several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the *direct supervisor* of each employee.

Example ABC Company Database

- Each employee may *have* a number of DEPENDENTs. For each dependent, we keep track of
  - Name
  - Gender
  - Birthdate
  - Relationship to employee.



Business Rules

- A set of brief, precise and unambiguous descriptions of policies, procedures or principle within a specific organization
- When properly designed they form an important source of entities, attributes, relationships and constraints
- How to discover them?
  - Company Managers, Policy Makers, Departmental Managers, Non managerial End-Users
  - Written Documentation: Company Procedures, Standards, Operations Manuals

Translating Business Rules into Data Models

- **Nouns** will translate to an entity in the model
- Verbs associate the nouns … represent a relationships
  - E.g. A **customer** may generate many **invoices**
- Relationships are bidirectional. To properly identify relationship type ask:
  - How many instances of B are related to one of A?
  - How many instances of A are related to one of B?

The Entity Relationship Model (ERM)

- 1st Proposed by Peter Chen in 1976
- A graphical representation of entities, attributes and their relationships in a database structure that complemented relational data model concepts

Main Phases of Database Design

1. Requirements Collection and Analysis
   - Interviews, Document Review, Observation etc. are used to determine users' data requirements + functional requirements
2. Create a conceptual schema
   - Concise description of entity types, relationships and constraints
3. Implementation of the DB using a commercial DBMS (logical design / data model mapping)
   - Transform the high-level data model into the implementation data model
4. Physical Design:
   - Specify the internal storage structures, indexes, access paths and file organizations.
   - At this same time application programs are being developed

Basic Building Blocks: Entity

- Anything about which data are to be collected and stored.
  - Physical: Person, Car, House
  - Conceptual: company, job, course, event (e.g. musical concert)
- Things in the mini-world that are represented in the database. E.g. EMPLOYEE John Mwangi, the Research DEPARTMENT, the ProductX PROJECT

Basic Building Blocks: Attributes

- Attribute: a descriptive characteristic of an entity,
  - For a customer: name, phone, address, credit limit
  - Each Entity has a value for its attributes
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Mwangi', PIN='123456789', Address ='P O Box 12345, Nyahururu 00909', Gender='M', BirthDate='09-JAN-75'
- Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, subrange, enumerated type, …

Basic Building Blocks

- Relationship: An association among entities
  - E.g. Agent serves customer
  - Types:
    - One-to-Many (1:M) Relationship
    - Many-to-Many (M:N or M:M) Relationship
    - One-to-One (1:1) Relationship
- Constraint: A restriction placed on data. Usually expressed in form of rules e.g.
  - Employee's salary to have values between 6,000 and 350,000
  - Each class must have one and only one teacher

Types of Attributes

- Simple
  - Each entity has a single atomic value for the attribute. For example, SSN or Gender. Not divisible.
- Composite
  - The attribute may be composed of several components. E.g.
    - Address (PO Box, Town, PostCode, Country). Or
    - Name (FirstName, MiddleName, LastName)
  - May form a hierarchy where some components are themselves composite.
    - e.g. StreetAddress (part of address) may consist of Number, StreetName and Apartment Number

Types of Attributes

- Single-Valued vs. Multivalued
  - Single Valued: Very common e.g. Age
  - Multi-valued: Entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.
- Stored vs. Derived Attributes
  - e.g. relationship between BirthDate and Age
- Null Values: Occur when values for the attribute are not:
  - Not Applicable
  - Not Available or
  - Not Known
- Complex Attributes:
  - Composite and multivalued attributes e.g. address (with its component parts), phone number(if a residence has multiple phones)

Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type. For example, the EMPLOYEE entity type or the PROJECT entity type.
- Entity Set= the collection of all entities of a particular entity type
- Important to have an attribute that provides Uniqueness to the entity: **The Key Attribute.**
- Examples in real life: PIN (or SSN) of EMPLOYEE, National ID No, Student Reg No. , company name etc.

Entity Types and Key Attributes

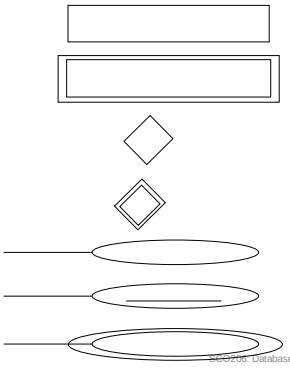
- A key attribute may be composite → the combination of attribute values provide the uniqueness of an entity. It must be minimal (least number of attributes rqrd to give uniqueness) e.g. student regno
- An entity type may have more than one key. For example, the CAR entity type may have two keys:
  - Chasis Number and
  - License_Plate Number.
- Entity may have no key → a weak entity type

ENTITY SET corresponding to the ENTITY TYPE CAR

CAR
License_Plate(Series, Number), Chasis Number, Make, Model, Year, (Color)

car_1
((KZZ, 999Z), AE90123456, Toyota Corolla, Saloon, 1987, (red, black))
car_2
((KAB, 123A), WP9879872, Nissan 300ZX, 2-door, 2002, (blue))
car_3
((KZZ, 222Z), TD-9786729, Ford Capri, 4-door, 1988, (white, blue))
.
.
.

Summarized Notation for ER Schema



Entity

Weak Entity

Relationship

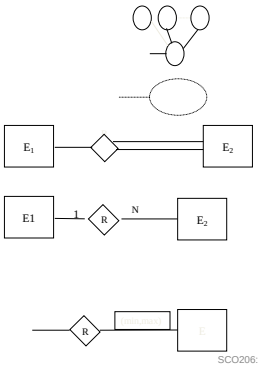
Identifying Relationship

Attribute

Key Attribute

Multivalued Attribute

Summarized Notation for ER Schema



Composite Attribute

Derived Attribute

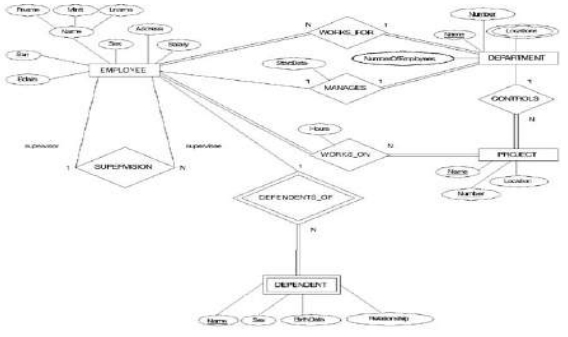
Total Participation of E_2 In R

Cardinality Ratio 1:N for E_1: E_2 In R

Structural Constraint (min,max) on Participation of E in R

ER DIAGRAM

Entity Types are EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT



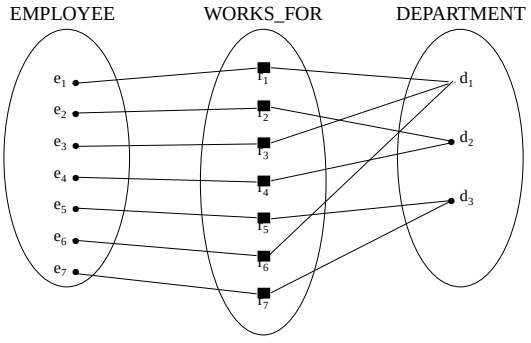
Relationships and Relationship Types

- A **relation type** R among *n* entity types $E_1, E_2, \ldots E_n$ defines a set of associations or a **relationship set** among entities from these entity types
- A relationship relates two or more distinct entities with a specific meaning. For example, EMPLOYEE John Mwangi works on the ProductX PROJECT or EMPLOYEE Joseph Oduor manages the Research DEPARTMENT.

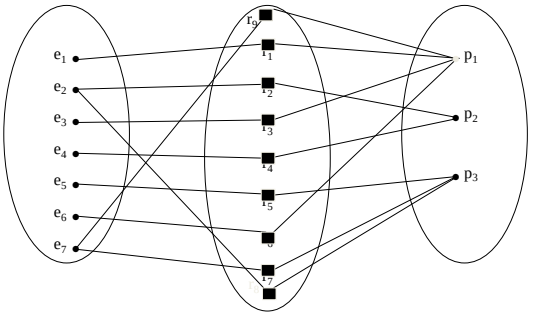
Relationships and Relationship Types (1)

- Relationships of the same type are grouped or typed into a relationship type. For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Hence WORKS_ON relationship has a degree of 2 (binary relationship).

- Example Relationship instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT



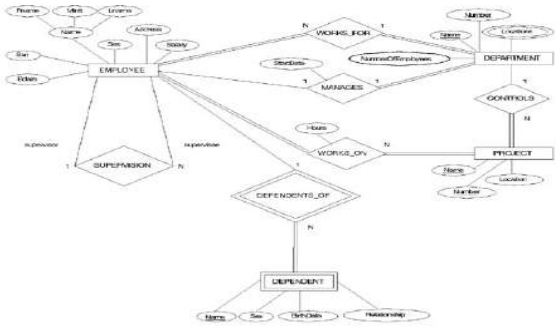
- Example relationship instances of the WORKS_ON relationship between EMPLOYEE and PROJECT



Relationships and Relationship Types

- More than one relationship type can exist with the same participating entity types. For example, MANAGES and WORKS_FOR are distinct relationships between EMPLOYEE and DEPARTMENT, but with different meanings and different relationship instances.

ER DIAGRAM – Relationship Types are:
WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF



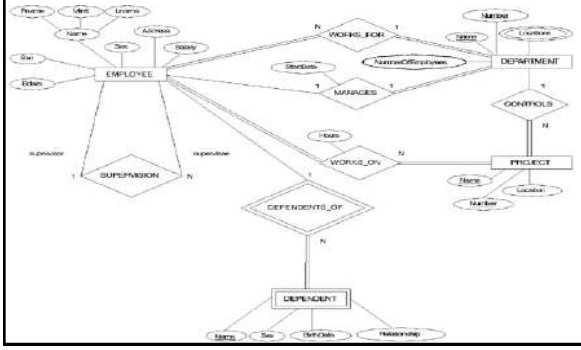
Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

Example:

Suppose that a DEPENDENT entity is identified by the dependent's first name and birthdate, *and* the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

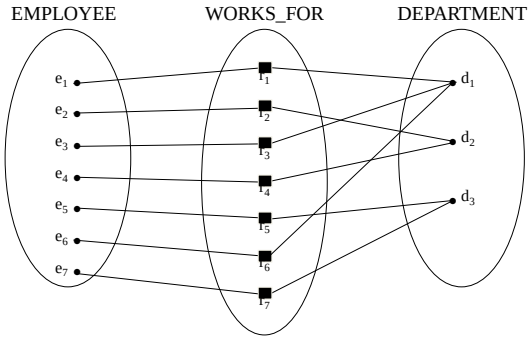
Weak Entity Type is: DEPENDENT
Identifying Relationship is: DEPENDENTS_OF



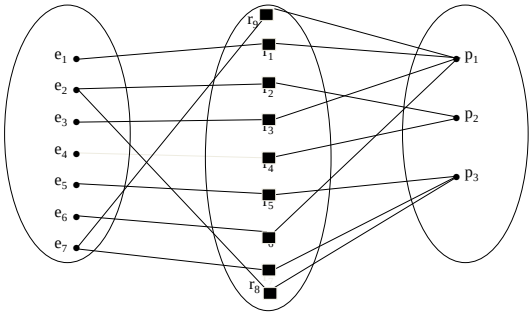
Constraints on Relationships

- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Maximum Cardinality
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many
  - Minimum Cardinality (also called participation constraint or existence dependency constraints)
    - Zero (optional participation, not existence-dependent)
    - One or more (mandatory, existence-dependent)

One-to-Many (1:N) RELATIONSHIP



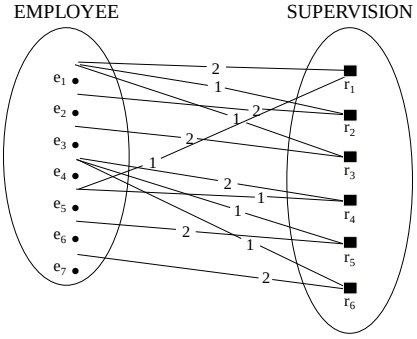
Many-to-Many (M:N) RELATIONSHIP



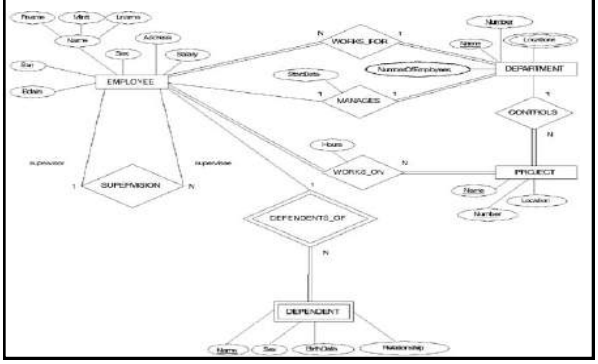
Relationships and Relationship Types

- We can also have a **recursive** relationship type.
- Both participations are same entity type in different roles.
- For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
- In following figure, first role participation labeled with 1 and second role participation labeled with 2.
- In ER diagram, need to display role names to distinguish participations.

A RECURSIVE RELATIONSHIP: SUPERVISION



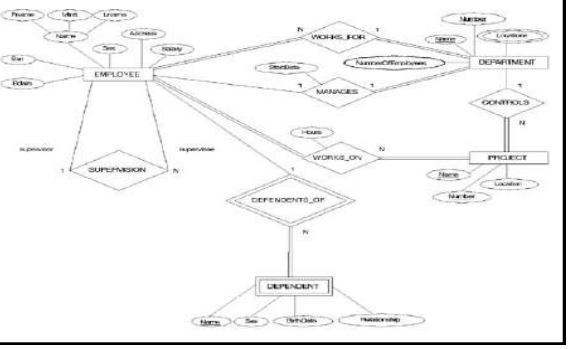
Recursive Relationship Type is: SUPERVISION
(participation role names are shown)



Attributes of Relation Types

- A relationship type can have attributes; for example, HoursPerWeek of WORKS_ON; its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.

Attribute of a Relationship Type is:
Hours of WORKS_ON



Structural Constraints – one way to express semantics of relationships

Structural constraints on relationships:

- **Cardinality ratio** (of a binary relationship): 1:1, 1:N, N:1, or M:N

  SHOWN BY PLACING APPROPRIATE NUMBER ON THE LINK.

- **Participation constraint** (on each participating entity type): total (called *existence dependency*) or partial.

  SHOWN BY DOUBLE LINING THE LINK

NOTE: These are easy to specify for Binary Relationship Types.

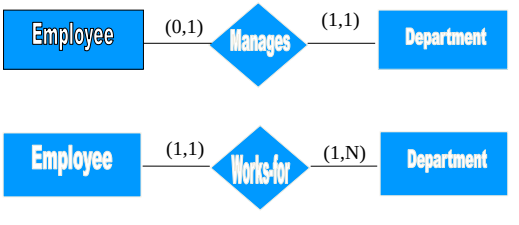
Alternative (min, max) notation for relationship structural constraints:

- Specified on *each participation* of an entity type E in a relationship type R
- Specifies that each entity e in E participates in *at least* min and *at most* max relationship instances in R
- Default(no constraint): min=0, max=n
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints

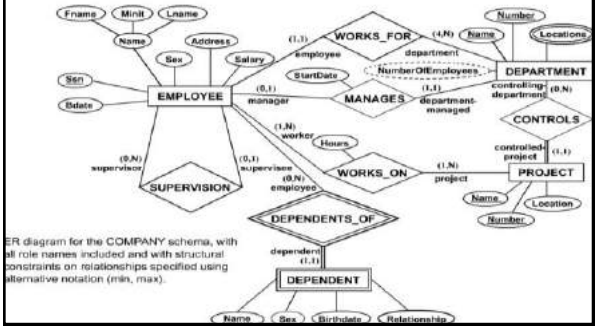
Examples:

- A department has *exactly one* manager and an employee can manage *at most one* department.
  - Specify (0,1) for participation of EMPLOYEE in MANAGES
  - Specify (1,1) for participation of DEPARTMENT in MANAGES
- An employee can work for *exactly one* department but a department can have *any number of employees*.
  - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
  - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

The (min,max) notation relationship constraints



ABC Company ER Schema Diagram using (min, max) notation

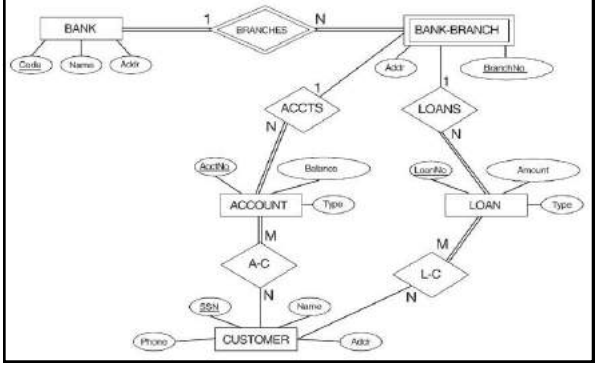


ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

Relationships of Higher Degree

- Relationship types of degree 2 are called **binary**
- Relationship types of degree 3 are called **ternary** and of degree n are called **n-ary**
- In general, an n-ary relationship *is not* equivalent to n binary relationships

ER DIAGRAM FOR A BANK DATABASE



Consider the ER diagram in previous slide:

a) List the (non-weak) entity types in the ER diagram
b) Is there a weak entity type? If so, give its name, partial key and identifying relationship
c) What constraints do the partial key and the identifying relationship of the weak entity type specify in the diagram?
d) List the names of all relationship types and specify the (min,max) constraint on each participation of an entity type in a relation type. Justify your choices.
e) List concisely the user requirements that led to this ER schema design.
f) Suppose that every customer must hae at least one account but is restricted to at most two loans at a timem and that a bank branch cannot have more than 1000 loans. How does this show up on the (min,max) constraints?