

Johns Hopkins Engineering

Principles of Database Systems

Module #2
Conceptual Database Design I

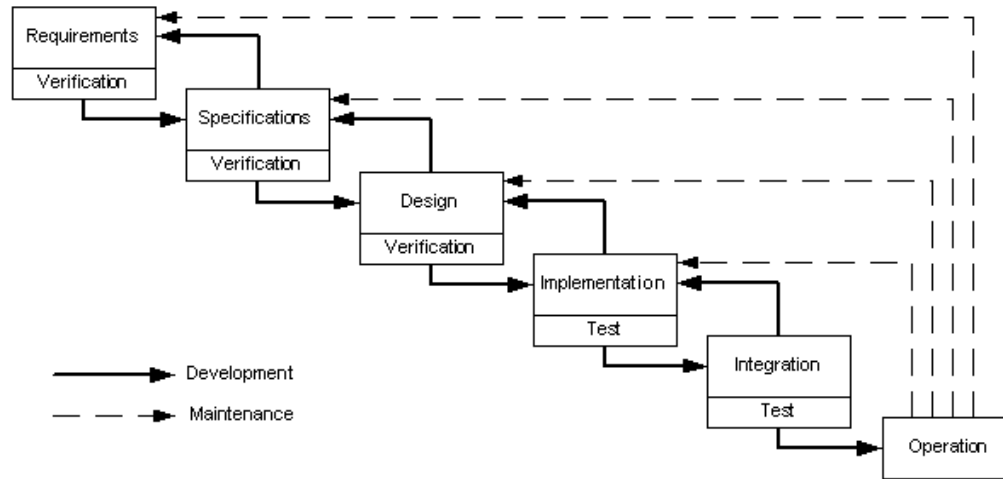
Software Engineering

- SW engineering is a discipline that focuses on production of quality software, delivered on time, within budget, that meets requirements, and satisfies clients' expectations.
- Software lifecycle includes requirements gathering, specification, design, implementation, integration, and maintenance phases.
- A database design and processing approach follows the phases of a software life cycle.

Software Lifecycle Model

- The series of steps through the product progresses
 - Waterfall model was widely used

Waterfall Model



The waterfall model with its feedback loops allows for revisions of all other stages of the process.

Software Lifecycle Model (cont.)

- Waterfall works well when
 - Requirements are stable:
 - Customers know what they want and customers will commit
 - Technology/methodology is well understood
 - Problem may be complex but well-understood
- Other lifecycle models
 - Agile, Rapid Prototyping Model, Incremental Model, Spiral Model, Staged Delivery Model, Design-to-Schedule Model

Project Management

- Project process groups
 - Initiating processes – define and authorize the project
 - Planning processes – define objectives, refine and plan the actions required to attain them
 - Executing processes – integrate resources to carry out the plan
 - Monitoring and controlling processes – measure progress to identify variance, and take corrective action if necessary
 - Closing processes – formally and orderly close the project with formal acceptance

Project Management (cont.)

- There are 9 Knowledge Areas:
 - Project management integration, scope, time, cost, quality, human resource, communication, risk, and procurement.
- Each area has multiple processes
 - Scope planning, definition, creating WBS, verification, and control.
- Project scope defines requirements
- A process has input, tools and techniques, and output.

System Lifecycle Model

- Database development is integrated into a system life cycle.

System Operations Concept Definition	System Requirements Definition	System Design	System Implementation	System Integration and Test	System Installation and Test	System Operations and Maintenance
Data System Concept Definition	Data System Requirements Definition	Data System Design	Database Implementation	Database Integration and Test	Database Installation and Test	Database Operations and Maintenance

Subsystem Requirements Definition	Subsystem Design	Subsystem Implementation	Subsystem Integration and Test
Data System Requirements Definition	Data System Subsystem Design	Subsystem Database Implementation	Subsystem Database Integration and Test

Johns Hopkins Engineering

Principles of Database Systems

Module 2 / Lecture 2
Conceptual Database Design I

Phases of Database Design

- Requirement collection and analysis.
- Conceptual database design to create conceptual schema, which is DBMS independent, and in a high-level data model.
- Logical database design to implement the database into a DBMS. This step is DBMS dependent.
- Physical database design to address the internal storage structure and file organization. This step is also DBMS dependent.

Sources of Unplanned and Planned Downtime

- Understand downtime sources

SOURCES OF UNPLANNED DOWNTIME

Percentile	Source
40%	Application Failure
40%	Operator Error
20%	Environmental factors such as hardware, operating system, power, disasters

SOURCES OF PLANNED DOWNTIME

Percentile	Source
65%	Application and database
13%	Hardware, networks, operating systems, systems software
10%	Batch application processing
10%	Backup and recovery
2%	Physical plant / environmentals

Source: Gartner research

Cost of Fixing or Tuning A Database System

- Two-thirds of the total software costs are devoted to maintenance.
- Cost vs. Time (design, development, production)
- Benefit vs. Time (design, development, production)

Sound database design is the foundation for a successful database project

The Main Phases Of Database Design

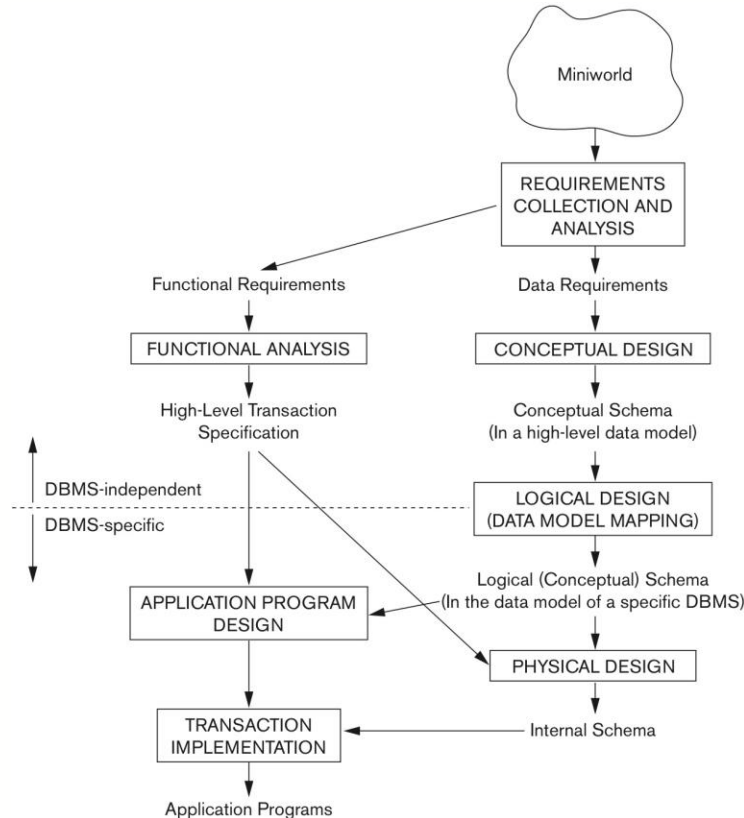


Figure 3.1 A simplified diagram to illustrate the main phases of database design.

Requirements

- Contains everything from business processes and functions
- Provides basis for creating database design and database application development
- Can be Word or Excel documents
- May include use cases:
 - Scenarios of WHO and What without concept of time

Requirements (cont.)

- May include data flow diagrams
 - Data elements are structured, but not normalized
 - May be used for data element or entity discovery
 - Lack of timing
- May include business process models (control flow)
 - Close to the business
 - Easy to understand by various stakeholders
 - Shows use of data with respect to timing
 - Describe what or who is working on a particular subset of a process

Johns Hopkins Engineering

Principles of Database Systems

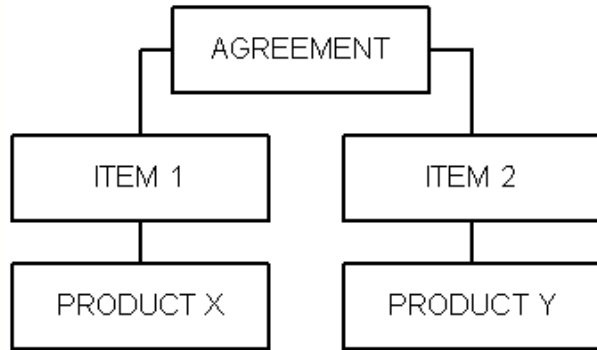
Module 2 / Lecture 3
Conceptual Database Design I

Entity-Relationship Model (ERM)

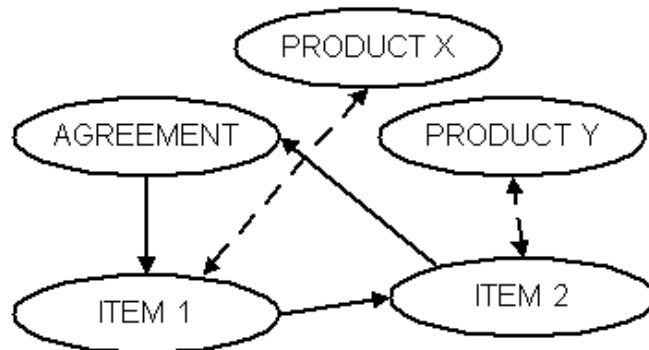
- Entity-Relationship Model or ER Model is a conceptual representation of data.
- Conceptual modeling is independent of the hardware or software to be used for implementation.
- An ER model can be mapped to a hierarchical, network, or relational database.
 - For Example: AGREEMENT to PRODUCT



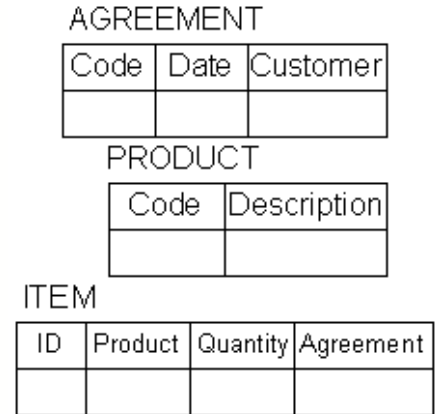
Entity-Relationship Model (ERM) (cont.)



Hierarchical database



Network database



Relational database

Conceptual Data Modeling

- A business process is a sequences of steps, tasks, or activities that converts inputs to outputs among business entities.
- Data modeling is to logically organize the governed data and its relationships, based on the business rules identified through analyzing the business process.
- The goal is to develop an entity-relationship (E-R) model that represents the information requirements of the business.

ER Model Components

■ Entity

- Important information needs to be held in the user environment within an organization.
- It can be an object(s) of interest to the business.
- It is a class or category of things such as a person, place, object, and event.
 - Person – EMPLOYEE, STUDENT
 - Place – STORE, STATE, HOUSE, DEPARTMENT
 - Object – VEHICLE, PROJECT, ORDER
- Entities can be thought of as nouns. Each entity has attributes to describe its properties.

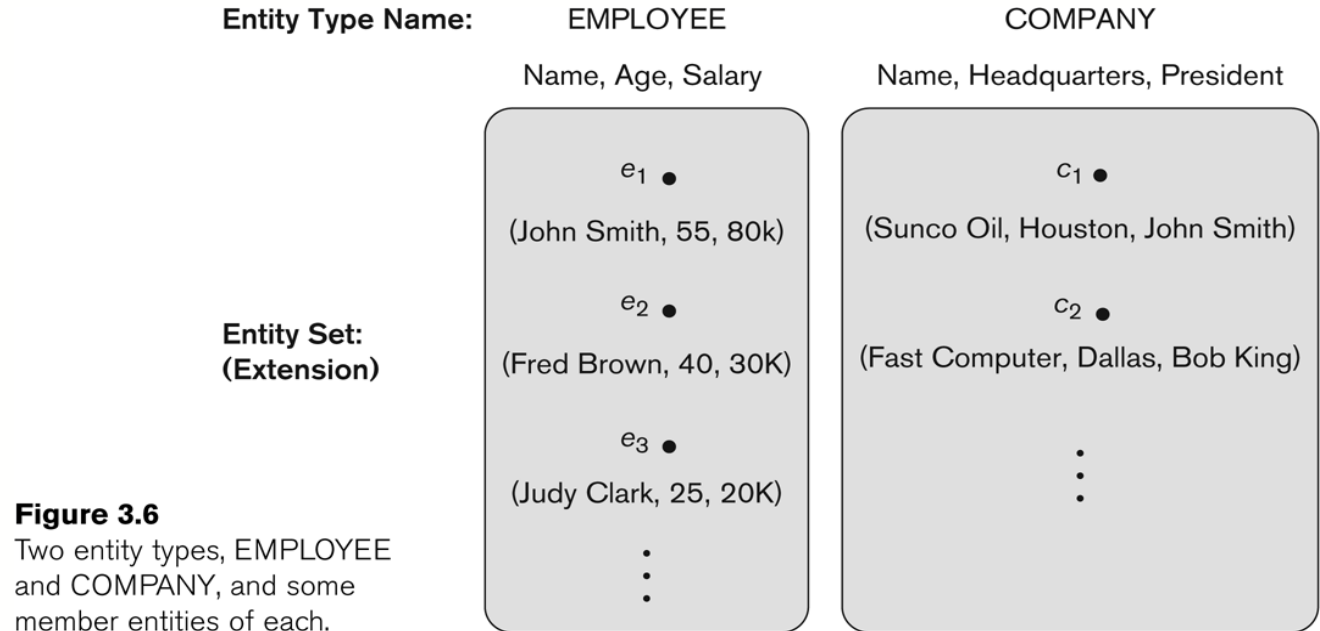
ER Model Components (cont.)

- Entity Type
 - A collection of entity instances that share common characteristics or properties
 - Name of an entity type is singular
 - An attribute, or set of attributes that uniquely identifies an entity is called a Unique Identifier (UID)

- An Entity Instance
 - Is a single occurrence of an entity type
 - Is a specific thing
 - Each instance must be uniquely identifiable from other instances of the same entity type.
 - For example: an employee John Smith with SSN or an employee ID, a vehicle with VIN as '1FALP52S6TA110981'

ER Model Components (cont.)

- Entity: An object or concept with attributes, that is important to the business
- Entity Type: A collection of entities that share common characteristics or properties
- Entity Instance: A single occurrence of an entity type



ER Model Components (cont.)

- Attribute
 - A property or characteristic of an entity type
 - The specific information that needs to be held. They describe an entity by qualifying, identifying, classifying, quantifying, or expressing the state of an entity instance.
 - For example, an employee's NAME, SSN, DOB, and SALARY.
 - An attribute represents a type of description or detail, not an instance.
 - For example, Mary is the first name of an EMPLOYEE instance, ZIP Code is a part of ADDRESS.

ER Model Components (cont.)

- Simple Attribute
 - An attribute that cannot be broken down into smaller components
 - For example, VIN, weight, or horsepower of a vehicle
- Composite Attribute
 - An attribute that can be broken down into component parts
 - For example, an address of a CUSTOMER entity type

Street Line 1 = street # + street name + suite/apartment # or PO box #

Street Line 2 = building # or name + other address qualifier

City, State, and Zip code

ER Model Components (cont.)

- Multivalued Attribute
 - An attribute that may have more than one value for a given entity instance
 - For example, skill of an employee, color of a vehicle
- Derived Attribute
 - An attribute whose value can be calculated from related attribute values
 - For example, a grand total of a transaction, GPA for a student

Johns Hopkins Engineering

Principles of Database Systems

Module 2 / Lecture 4
Conceptual Database Design I

Common Practices For Creating Attributes

- Always break attributes down into their lowest meaningful components:
 - For example, the name of PERSON can be broken down into last_name, first_name
- Break down aggregate attributes, composite attributes, and embedded code fields into simple attributes:
 - Address can be decomposed into multiple fields: street, apartment/suite, city, state, and zip code
 - In general, dates, times, SSN, and zip code are not decomposed further

Common Practices For Creating Attributes (cont.)

- Verify that an attribute is not derived or calculated from the existing values of other attributes, e.g. age, total
- Artificial attributes are used often for UIDs. An artificial code is defined when the business does not have a natural attribute that uniquely identifies an entity.
 - For example: first name, last name, address, and customer_id on a CUSTOMER entity type

Strong vs. Weak Entity Types

- Strong or Independent Entity Type:
 - An entity type that exists independently of other entity types.
 - An entity type whose instances can be uniquely identified without determining its *relationship* to another entity.
- Weak or Dependent Entity Type:
 - An entity type whose existence depends on other entity types.
 - An entity type whose instances cannot be uniquely identified without determining its *relationship* to another entity.

Relationship

- How the entities are related
- It is a two-directional, significant association between two entities, or between two entity types
- Relationship can be either - *must be* (mandatory) or *may be* (optional)
 - For example, the relationship between DEPARTMENT and EMPLOYEE: Explain a reasonable relationship(s)

Identifying vs. Non-identifying Relationship

- Identifying Relationship:
 - The relationship is between a weak entity type and its owner.
 - An instance of a weak (child) entity type fully depends on an instance of a strong (parent) entity type.
- Non-identifying Relationship:
 - The relationship may be between two strong entity types.
 - An instance of a strong entity type can exist without depending on other entity type.

Relationship Cardinality

- Relationships have a property called cardinality. Cardinality statements designate "how many" instances of the parent entity are connected to "how many" instances of the child.
- Most relationships are "one to something".
- Any many-to-many relationship must be broken down into a pair of one-to-many relationships.

Relationship Cardinality (cont.)

- Cardinality specifies that an entity instance can connect to how many other entity instances in a relationship.
 - A cardinality 0 represents a may be relationship.
 - Any many-to-many relationship must be broken down into a pair of one-to-many relationships.
- Relationship Types
 - One to Many (1:M);
 - Many to Many (M:M) or (M:N);
 - One to One (1:1)
- All relationships should represent the information requirements and rules of the business.

Chen's Notation for ER Modeling

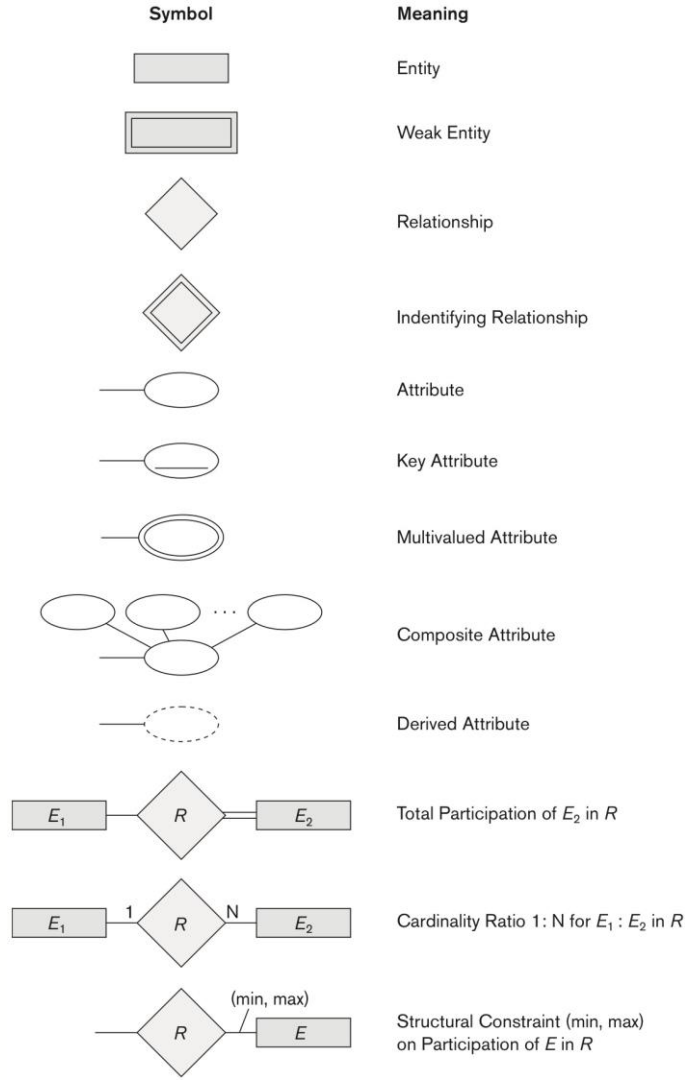


Figure 3.14
Summary of the
notation for ER
diagrams.

Johns Hopkins Engineering

Principles of Database Systems

Module 2 / Lecture 5
Conceptual Database Design I

Total Participation (Existence Dependence)

- Every entity in an entity type *must be related* to another entity type via a relationship.
- The relationship for total participation is displayed as *double lines* connecting the participating entity type to the relationship.

Figure 3.2 – An ER schema diagram for COMPANY database.

EMPLOYEE and WORK_FOR (An employee must work for a department)

DEPARTMENT and WORK_FOR (A department must have an employee(s))

DEPARTMENT and MANAGES (A department must have one manager)

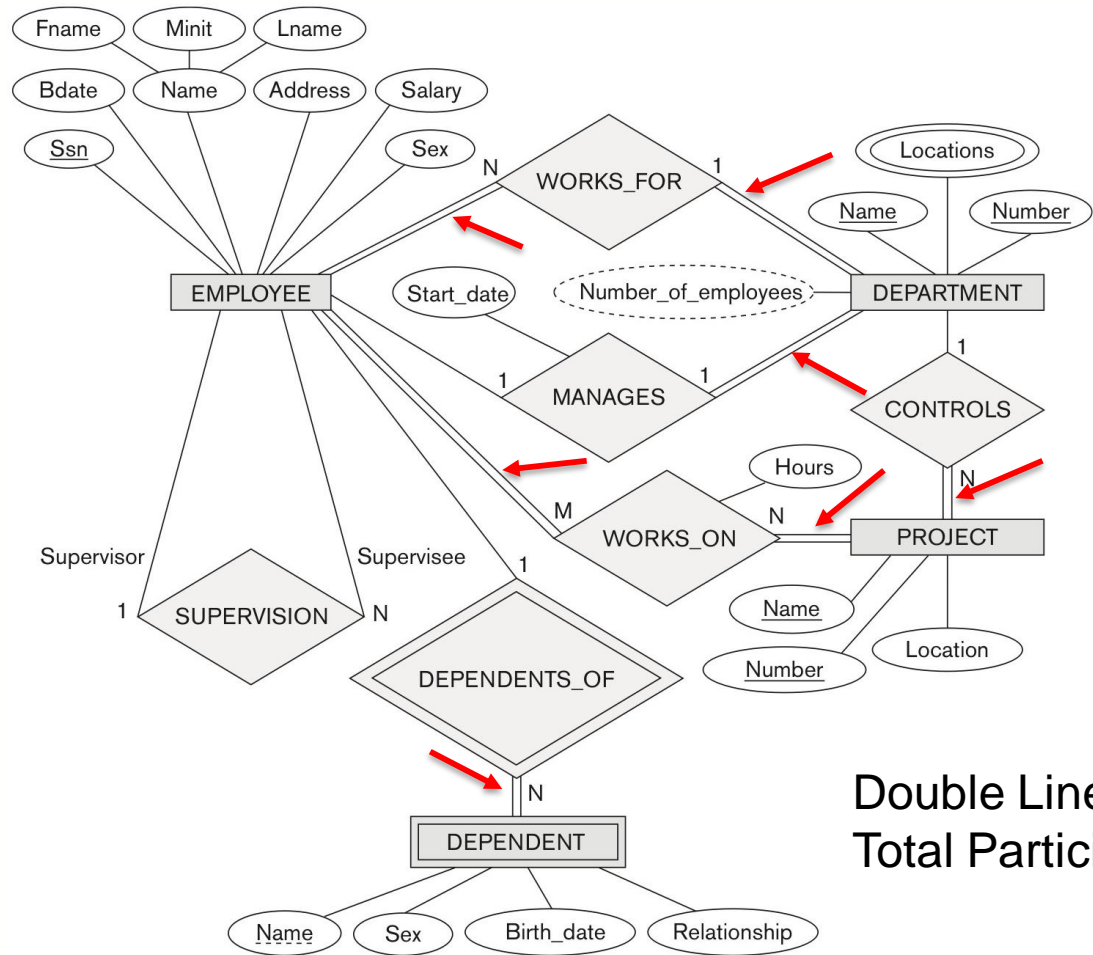
EMPLOYEE and WORK_ON (An employee must work on a project)

PROJECT and WORK_ON (A project has to be assigned an employee to work)

DEPENDENT and DEPENDENTS_OF (A dependent must link to an employee)

Total Participation (cont.)

Figure 3.2 An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.



Double Lines –
Total Participation

Total Participation Interpretation

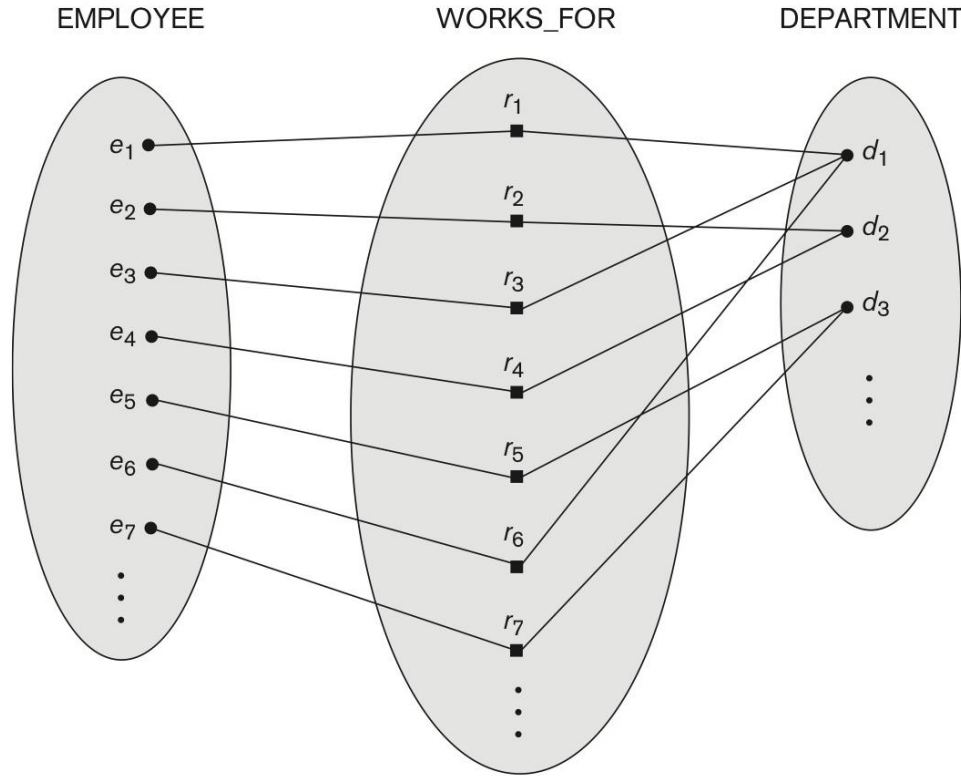


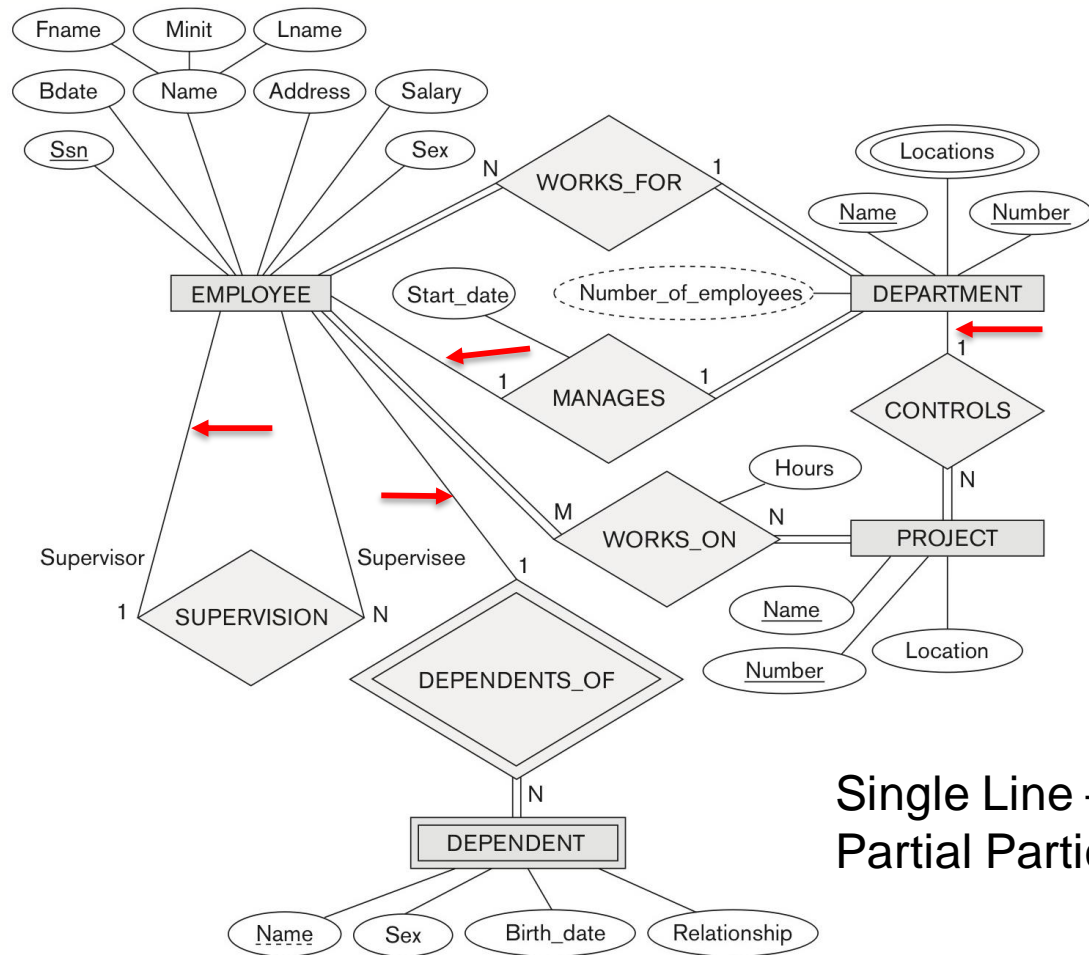
Figure 3.9 Some instances in the `WORKS_FOR` relationship set, which represents a relationship type `WORKS_FOR` between `EMPLOYEE` and `DEPARTMENT`.

Partial Participation

- If not a total participation, then relationship is a partial participation with *a single-line* connection.
 - Examples
 - EMPLOYEE and MANAGES (An employee may manage a department.)
 - EMPLOYEE and DEPENDENTS_OF (An employee may have a dependent).

Partial Participation (cont.)

Figure 3.2 An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 3.14.



Single Line –
Partial Participation

Partial Participation (cont.)

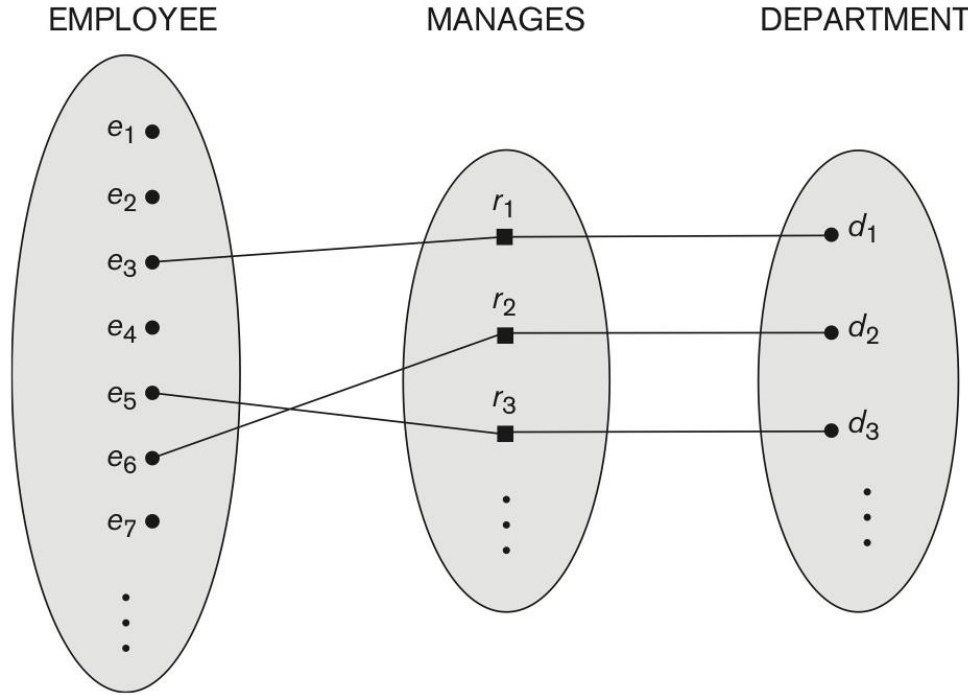
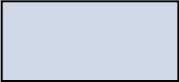



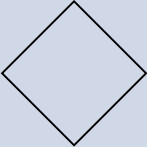

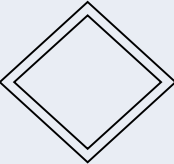




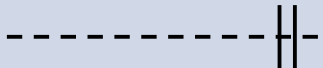
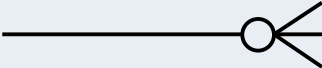



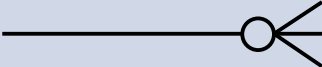





Figure 3.12 A 1:1 relationship, MANAGES, with partial participation of EMPLOYEE and total participation of DEPARTMENT.

Chen's and IE/IDEF1X Notations

	Chen	IE
Strong Entity		
Weak Entity		
Non-identifying Relationship		
Identifying Relationship		

Chen's and IE/IDEF1X Notations (cont.)

	Chen	IE
Single Cardinality	<u>1</u>	 or   or 
Multiple Cardinality	<u>N</u>	 or   or 
A Pair of Cardinality	<u>(0, N)</u>	 or 
A Pair of Cardinality	<u>(x, N)</u>	 or 

Understanding Cardinality Constraints

- Cardinality constraint:
 - Specifies the number of instances of one entity type that can (or must) be associated with each instance of another entity type.
- Minimum cardinality:
 - The minimum number of instances of one entity type that may be associated with each instance of another entity type.
- Maximum cardinality:
 - The maximum number of instances of one entity type that may be associated with each instance of another entity type.

Johns Hopkins Engineering

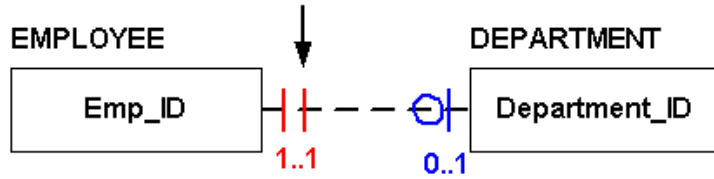
Principles of Database Systems

Module 2 / Lecture 6
Conceptual Database Design I

Interpreting Relationships in IE (Crow's Foot)

1:1 Relationship

Each department is managed by one employee.



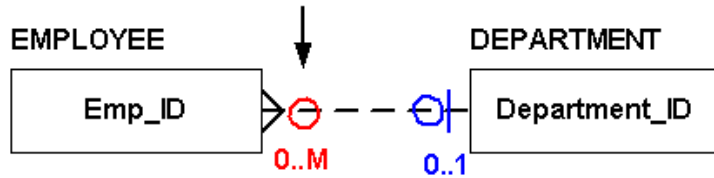
A pair of (1, 1)
associated with a
Parent EMPLOYEE

A pair of (0, 1) associated
with a Child DEPARTMENT

An employee manages zero or one department.

1:M Relationship

Each department has zero to many employees.



A pair of (1, M)
associated with a
Child EMPLOYEE

A pair of (0, 1)
associated with a Parent
DEPARTMENT

An employee works for zero or one department.

Interpreting Cardinality Ratios with Chen's Notation And IE Notation

- Typical parent-child entities as a 1:N relationship with “**Entity 1**” (a parent) and “**Entity 2**” (a child).



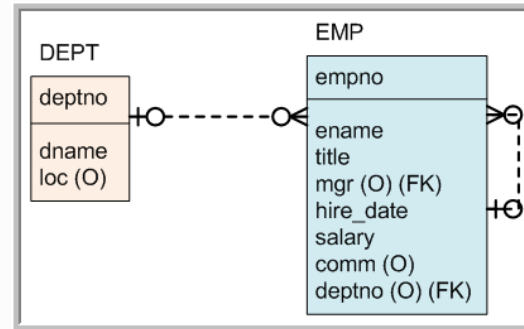
- Chen's and IE notations may not show a pair of (min, max).
- IE notation includes a pair of (min, max) cardinality ratios. How to identify/interpret (min, max) for both entities with total and partial participations:



An ERD using IE Notation - Example 1

- Entities: EMP, DEPT
- Relationship 1:
 - One department may have zero or many employees
 - For instance, a new department may not have any employees in the beginning
 - One employee may be assigned to zero or one department
 - For instance, a new employee may be not assigned to a department yet
- Relationship 2:
 - An employee may report to another employee

- A relationship connecting from DEPT (parent) to EMP (child) shows a primary key (PK) migration to a child entity as a foreign key (FK).
- A FK value in a record of EMP links to a parent record – an employee assigned to a department.



Tool: Visio

Chen's Notation with (Min, Max) Pairs of Cardinality Ratios

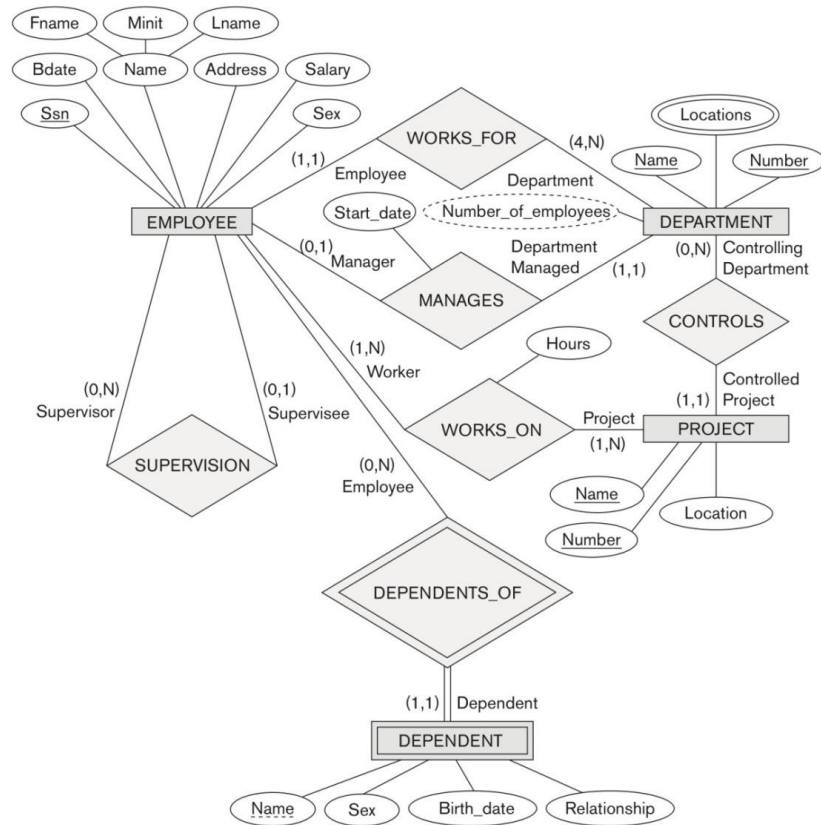
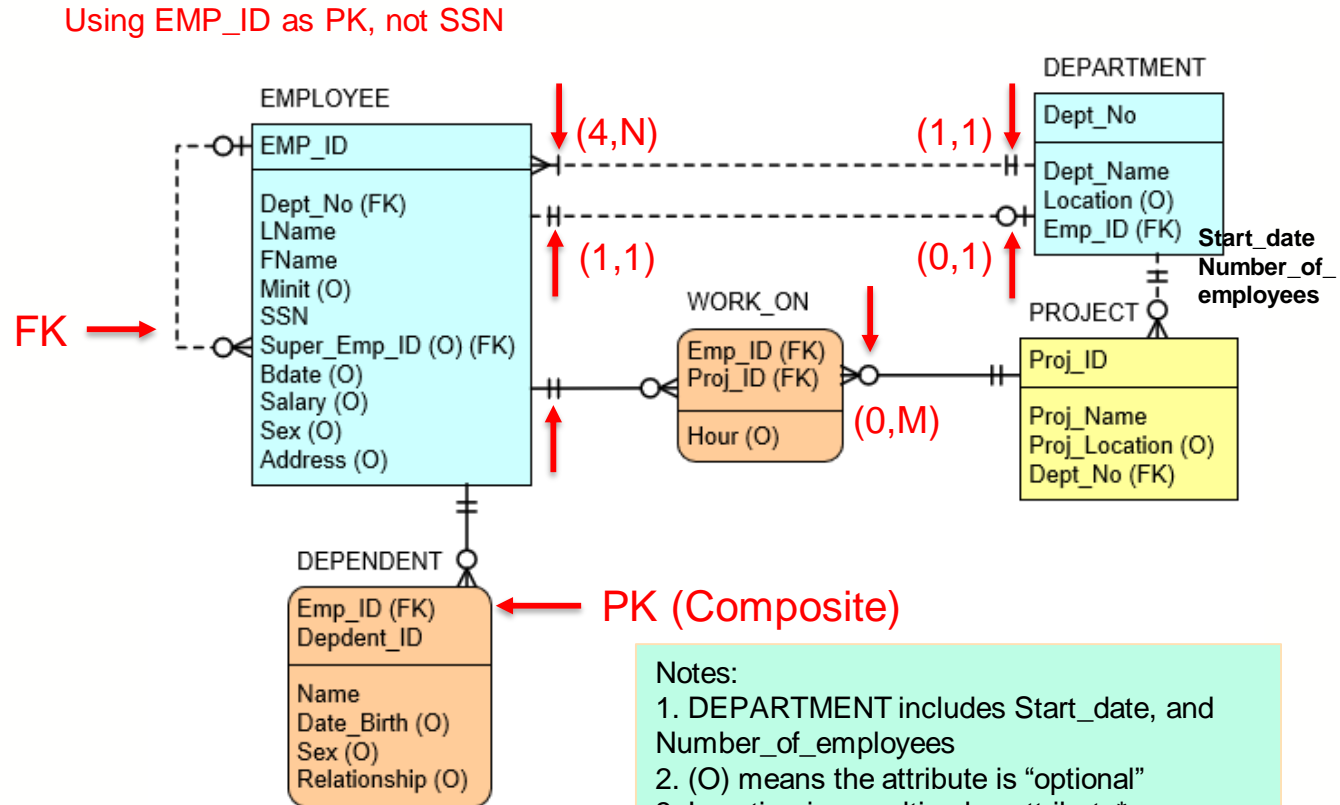


Figure 3.15 ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Company ERD Using Visio with IE Notation



Please compare and contrast the Company ERD using IE notation and the Company ERD using Chen's notation shown in Figure 3.

Chen's Notation vs. IE Notation

■ Chen's Notation

- May be easy to learn for a beginner.
- May be easier to understand relationships among entities (removing attributes) at a high level.
- Can show derived attribute and multivalued attributes.
- An ERD begins to feel cluttered when the data model begins to scale.
- Can draw an ERD with a graphical tool. As a result, an ERD only supports conceptual database design. Most database design tools don't support Chen's notation.

■ IE Notation

- May be is intuitive to a beginner.
- Lead to a compact and precise ERDs that are closer to actual implementations.
- Supports foreign key migration automatically that helps a designer and other stakeholders better understand the design and implementation.
- Database design tools support IE. They can easily support a conceptual design to logical and physical designs and further database application development.