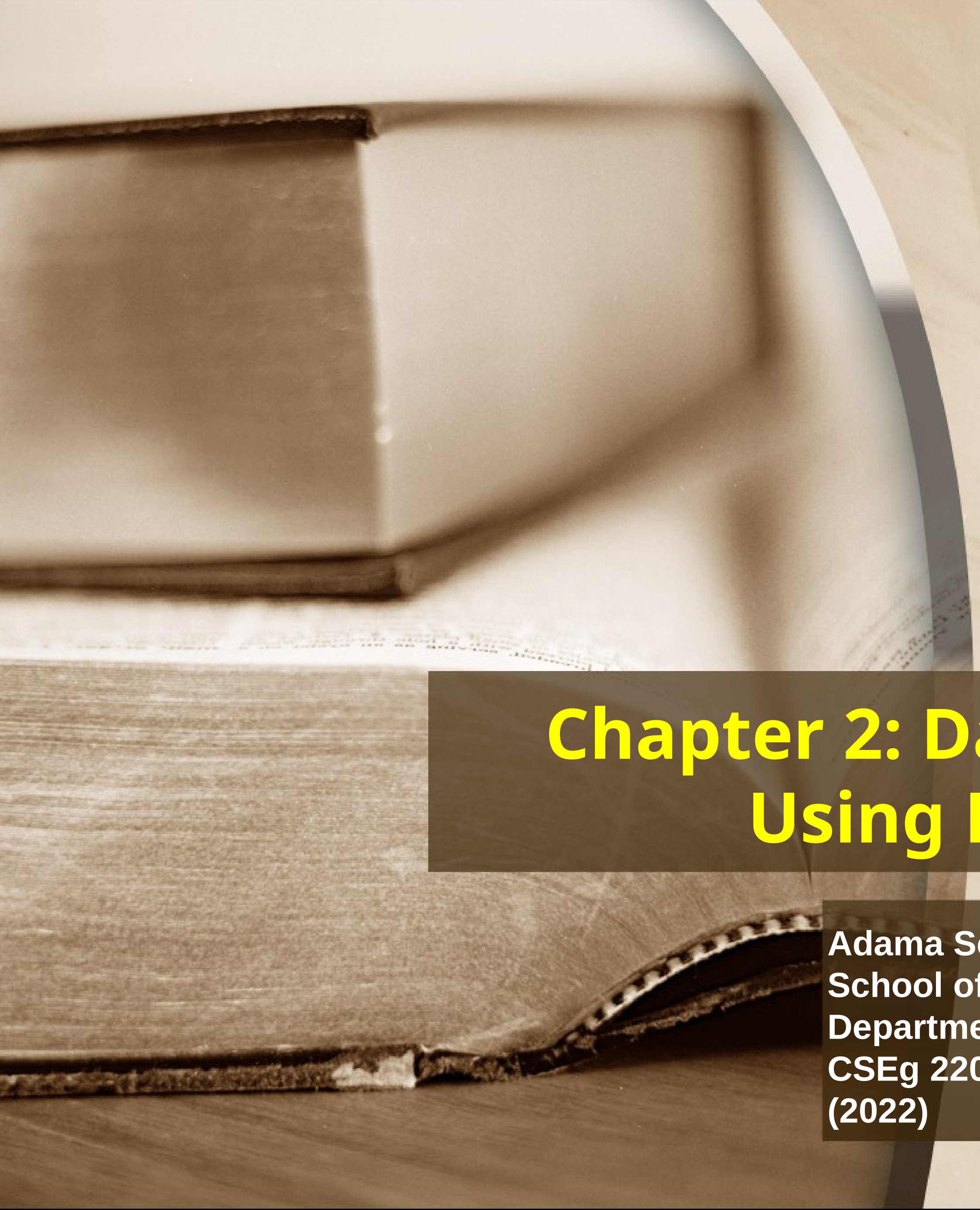


Chapter 2: Data Modeling – Using ER Model

Adama Science and Technology University
School of Electrical Engineering and Computing
Department of CSE

CSEg 2208: Database Systems
(2022)

Outline

- Data Modeling
- Design Processing
- Conceptual Models

Database Model

- One of the most vexing *problems of database design* is that *designers*, *programmers*, and *end users* see data in different ways.
- Different views of *same data lead to designs* that do not reflect organization's actual operation, thus failing to meet *end-user needs* and *data efficiency requirement*.
- To avoid such failures, database designers must obtain a *precise description of the nature of the data* and of *the many uses of that data* within the organization.
- Data modeling reduces complexities of database design using *various degrees of data abstraction* that help to *reconcile varying* views of same data.

Database Model

- Database design focuses on how the *database structure* will be used to *store* and *manage* end-user data.
- Data modeling is the *first step in the database design* journey, serving as a *bridge between real-world objects* and the *database* that resides in the computer.
- Data models: Relatively *simple representations* of the complex real-world data structures. Usually it is *graphical*.
- Model: an abstraction of a *real-world object* or *event*.
  - Useful in *understanding complexities* of the real-world environment.
- Data modeling is *iterative* and *progressive*.

Importance of Data Models

- It facilitate *interaction* among the designer, the applications programmer, and the end user. (communication tool)
- End users have different *views* and *needs* for data.
- Data model *organizes data* for various users.

- Data model is an *abstraction*, you cannot *draw the required data* out of the data model.
- Just as you are not likely to build a *good house* without a *blueprint*, you are equally unlikely to create a *good database* without first creating an *appropriate data model*.

Data Model Basic Building Blocks

- The basic building blocks of all data models are *entities*, *attributes*, *relationships*, and *constraints*.
- *Entity*: anything (a person, a place, a thing, or an event) about which data are to be collected and stored.
- *Attribute*: a *characteristic* of an entity.
- *Relationship*: describes an *association* among entities:
  - 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 the data.

Business Rules

- How do you properly identify *entities*, *attributes*, *relationships*, and *constraints*?
  - The first step is to clearly identify the *business rules* for the *problem domain* you are modeling.

- Business rules are descriptions of *policies*, *procedures*, or *principles* within a specific organization.
  - Description of *operations* to create/enforce actions within an organization's environment.
  - Must be in *writing* and kept *up to date*.
  - Must be *easy to understand* and *widely disseminated*.
  - Describe characteristics of data as *viewed by the company*.

Discovering Business Rules

- The main sources of *business rules* are:
  - Company managers,
  - Policy makers,
  - Department managers,
  - Written documentation such as:
    - Procedures,
    - Standards,
    - Operations manuals.
  - Direct interviews with end users.
- Of course, *not all business rules* can be modeled. Some business rules (constraints) can be enforced by *application software*.

Discovering Business Rules

- The process of *identifying* and *documenting business rules* is essential to database design for several reasons:
  - They help to *standardize the company’s view* of data.
  - They can be a *communications tool* between users and designers.
  - They allow the *designer to understand the nature*, *role*, and *scope* of the data.
  - They allow the designer to *understand business processes*.
  - They allow the designer to develop appropriate relationship *participation rules* and *constraints* and to create an accurate data model.

Translating Business Rules into Data Model Components

- Generally, *nouns* translate into *entities*.
- *Verbs* translate into *relationships* among entities.
- Relationships are *bidirectional*.

- Two questions to identify the *relationship type*:
  - How many instances of B are related to one instance of A?
  - How many instances of A are related to one instance of B?

The Evolution of Data Models

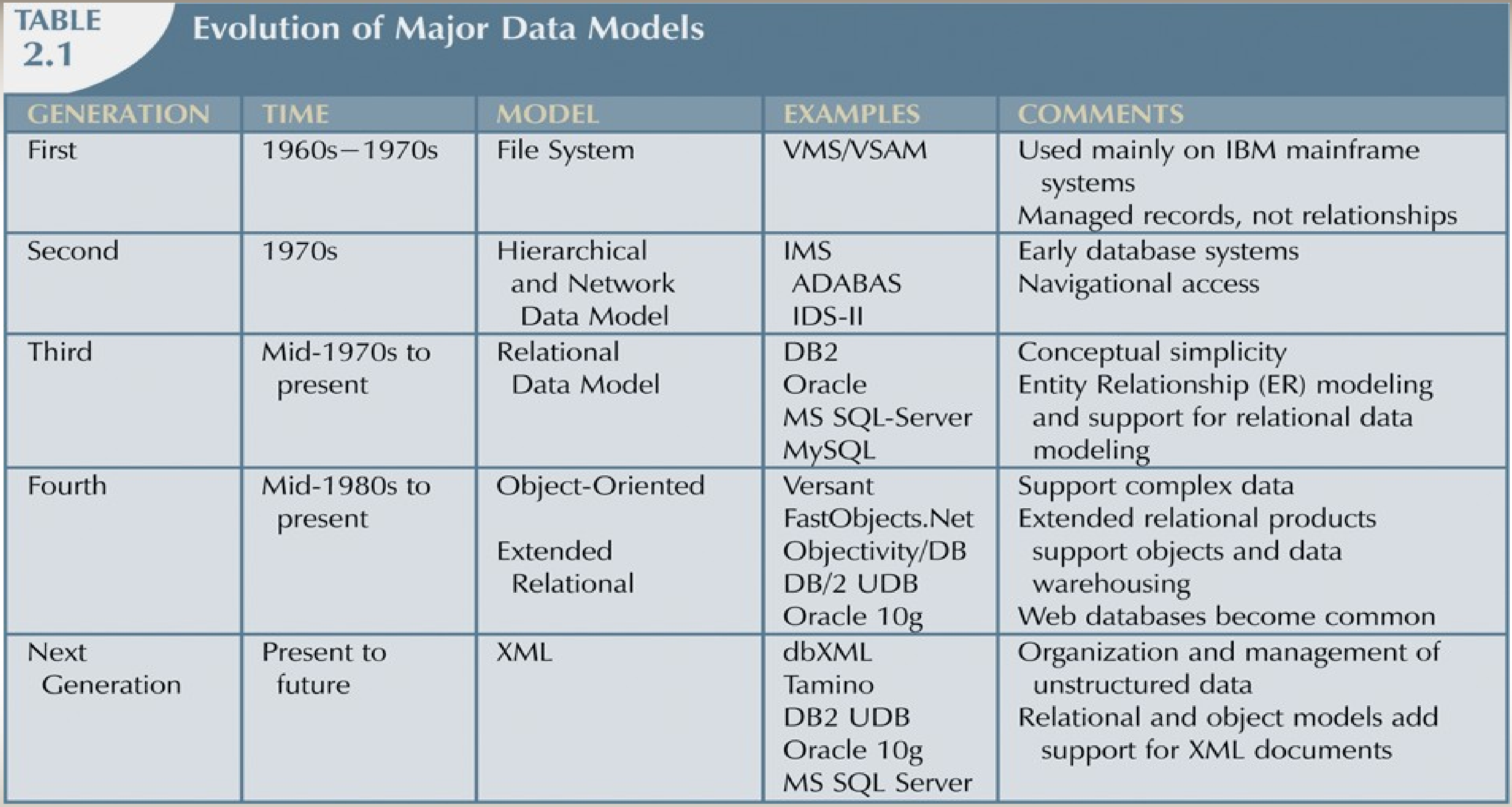


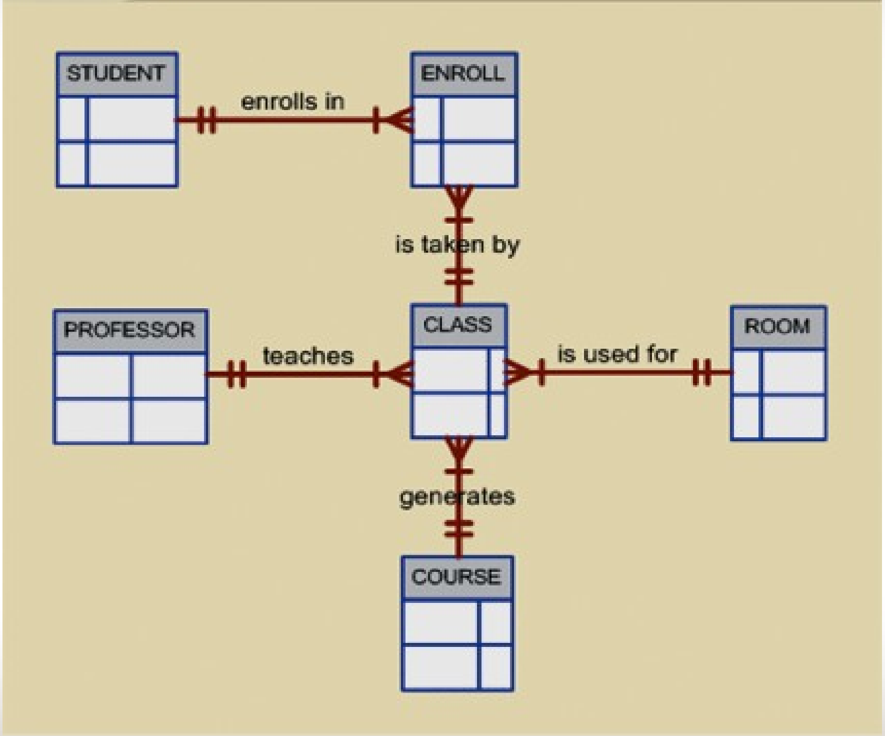
TABLE 2.1 Evolution of Major Data Models

GENERATION	TIME	MODEL	EXAMPLES	COMMENTS
First	1960s–1970s	File System	VMS/VSAM	Used mainly on IBM mainframe systems Managed records, not relationships
Second	1970s	Hierarchical and Network Data Model	IMS ADABAS IDS-II	Early database systems Navigational access
Third	Mid-1970s to present	Relational Data Model	DB2 Oracle MS SQL-Server MySQL	Conceptual simplicity Entity Relationship (ER) modeling and support for relational data modeling
Fourth	Mid-1980s to present	Object-Oriented Extended Relational	Versant FastObjects.Net Objectivity/DB DB/2 UDB Oracle 10g	Support complex data Extended relational products support objects and data warehousing Web databases become common
Next Generation	Present to future	XML	dbXML Tamino DB2 UDB Oracle 10g MS SQL Server	Organization and management of unstructured data Relational and object models add support for XML documents

Conceptual Model

- Represents global view of the *entire database*.
- All *external views integrated* into single global view: conceptual schema.
- ER model is the most widely used *conceptual model*.
- ERD graphically represents the conceptual schema

FIGURE 2.10 Conceptual model for Tiny College



Conceptual Model

- Provides a relatively *easily understood macro level view* of data environment.

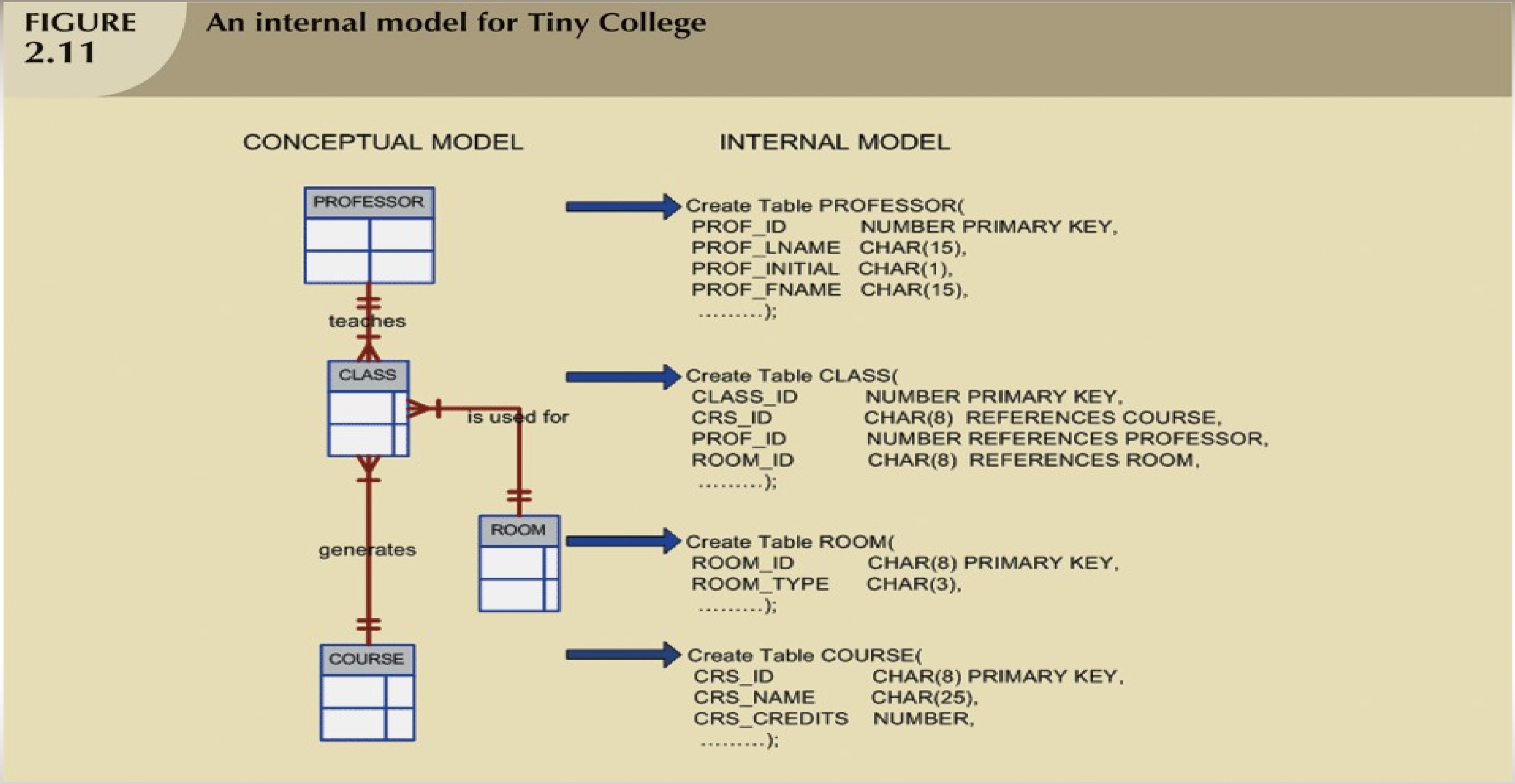
- Independent of both *software* and *hardware*:
  - Does not depend on the *DBMS software* used to implement the model.
  - Does not depend on the *hardware used* in the implementation of the model.
  - Changes in *hardware* or *software* do not affect database design at the conceptual level.

Internal Model

- Representation of the database as “seen” *by the DBMS*.
  - Maps the *conceptual model* to the DBMS.
- Internal schema depicts a specific representation of an internal model.
- Depends on *specific database software*:
  - Change in *DBMS software* requires internal model to be changed.
- *Logical independence*: change internal model without affecting *conceptual model*.

Internal Model

FIGURE 2.11 An internal model for Tiny College



```
Create Table PROFESSOR(
 PROF_ID        NUMBER PRIMARY KEY,
 PROF_LNAME     CHAR(15),
 PROF_INITIAL   CHAR(1),
 PROF_FNAME     CHAR(15),
 .........);

Create Table CLASS(
 CLASS_ID       NUMBER PRIMARY KEY,
 CRS_ID         CHAR(8)  REFERENCES COURSE,
 PROF_ID        NUMBER REFERENCES PROFESSOR,
 ROOM_ID        CHAR(8)  REFERENCES ROOM,
 .........);

Create Table ROOM(
 ROOM_ID        CHAR(8) PRIMARY KEY,
 ROOM_TYPE      CHAR(3),
 .........);

Create Table COURSE(
 CRS_ID         CHAR(8) PRIMARY KEY,
 CRS_NAME       CHAR(25),
 CRS_CREDITS    NUMBER,
 .........);
```

Physical Model

- Operates at lowest level of abstraction.
  - Describes the *way data are saved on storage media* such as disks or tapes.
- Requires the definition of *physical storage* and *data access methods*.

- Relational model aimed at logical level:
  - Does not require *physical-level details*.
- *Physical independence*: changes in physical model do not affect internal model.

Physical Model

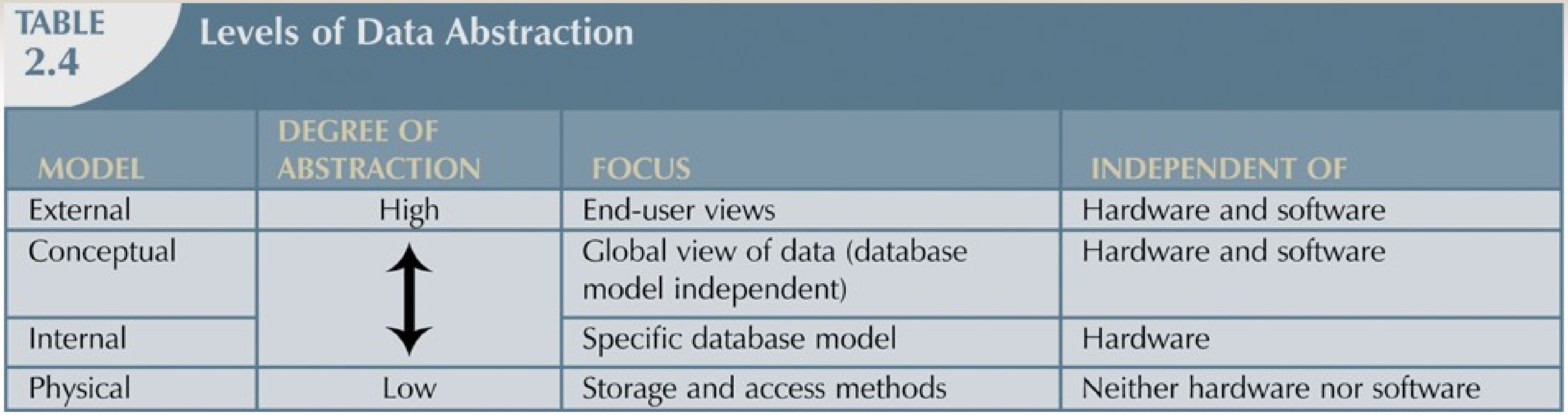


TABLE 2.4 Levels of Data Abstraction

MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual	↕	Global view of data (database model independent)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

Requirements Analysis

- A database is intended to *model a real* world enterprise.
- What data are to be *stored in the database*?
    - Linguistic aside: data is *plural*, datum is *singular*.

- What *applications* are required to work with the database?
- Which are the *most frequent*, and the *most important operations*?

Conceptual Database Design

- Use the *Entity Relationship (ER) model* to develop a high level description of the data.
- Identify the *entities* and *relationships* in the enterprise.
- Identify *what information* about these entities and relationships is to be stored in the database.
- Identify the *integrity constraints* (or business rules) that apply to the entities and relationships.

Entity-Relationship Model

- The most common abstract *data model*. The major components of the ER Model are:
- *Entities* and *Attributes:*
  - *Entities* are specific objects or things in the mini-world that are represented in the database.
    - For example : EMPLOYEE,DEPARTMENT, PROJECT
  - *Attributes* are properties used to describe an entity. For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate
  - A *specific entity* will have a value for each of its attributes.
    - For example a specific employee entity may have Name=lema tefera', SSN='123456789', Address ='731, Sex='M', BirthDate='09-JAN-55‘

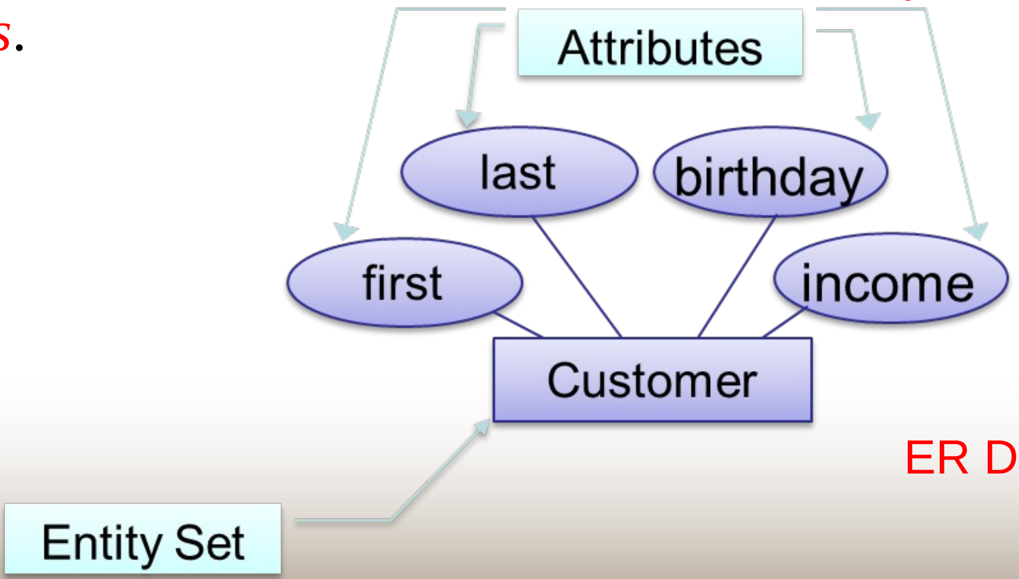
Entity-Relationship Model

- Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, subrange, enumerated type, …

- Entity-relationship diagrams show the *structure of a database* graphically:
- *Relationships* that connect the *entities*, e.g. customer owns account, student takes course.
- *Simple symbols*: rectangles, diamonds, ovals and lines represent the components of the ER model.
- They are *straightforward* and *easy* to explain to users.

Entity-Relationship Model

- Entity Sets: An entity set is a *collection of entities* of the same type.
  - e.g. all of the customers.
- An entity is described by a *set of attributes*.
  - Each attribute has a domain with a *set of all possible attribute values*.



ER Diagram Entity Sets

Types of Attributes

- Simple:
  - Each entity has a *single atomic value* for the attribute.
    - For example, SSN or Sex.
- Composite:
  - The attribute may be *composed of several components*.
    - For example: Address (Apt#, House#, Street, City, region,Country) or
    - Name (FirstName, MiddleName, LastName)..
- Multi-valued:
  - An entity may *have multiple values* for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

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.

- An attribute of an entity type for which each entity must have a unique value is called a *key attribute of the entity type*.
  - For example, SSN of EMPLOYEE.

Entity Types and Key Attributes

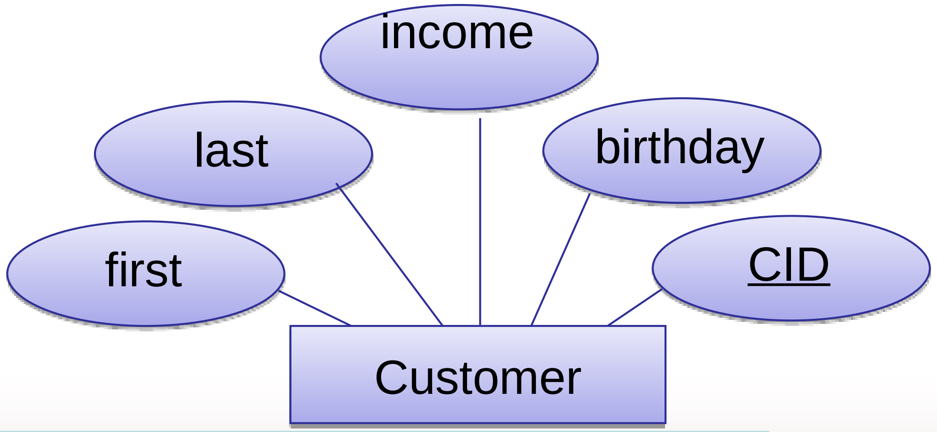
- A key attribute may be *composite*.
  - For example, VehicleTagNumber is a key of the CAR entity type with components (*Number*, *Region*).

- An entity type may have *more than one key*. For example, the CAR entity type may have *two keys*:
  - VehicleIdentificationNumber (popularly called VIN) and
  - VehicleTagNumber (Number, Region), also known as license_plate number.

Entity Types and Key Attributes

- Primary Key- is a set of attributes whose values *uniquely identify an entity* in an entity set;
  - The primary key should be chosen so that its attributes never (or very rarely) *change*.
  - Primary key attributes are *underlined* in an ERD.

*assuming (unrealistically) that there are no two people with the same first name, last name and birthday

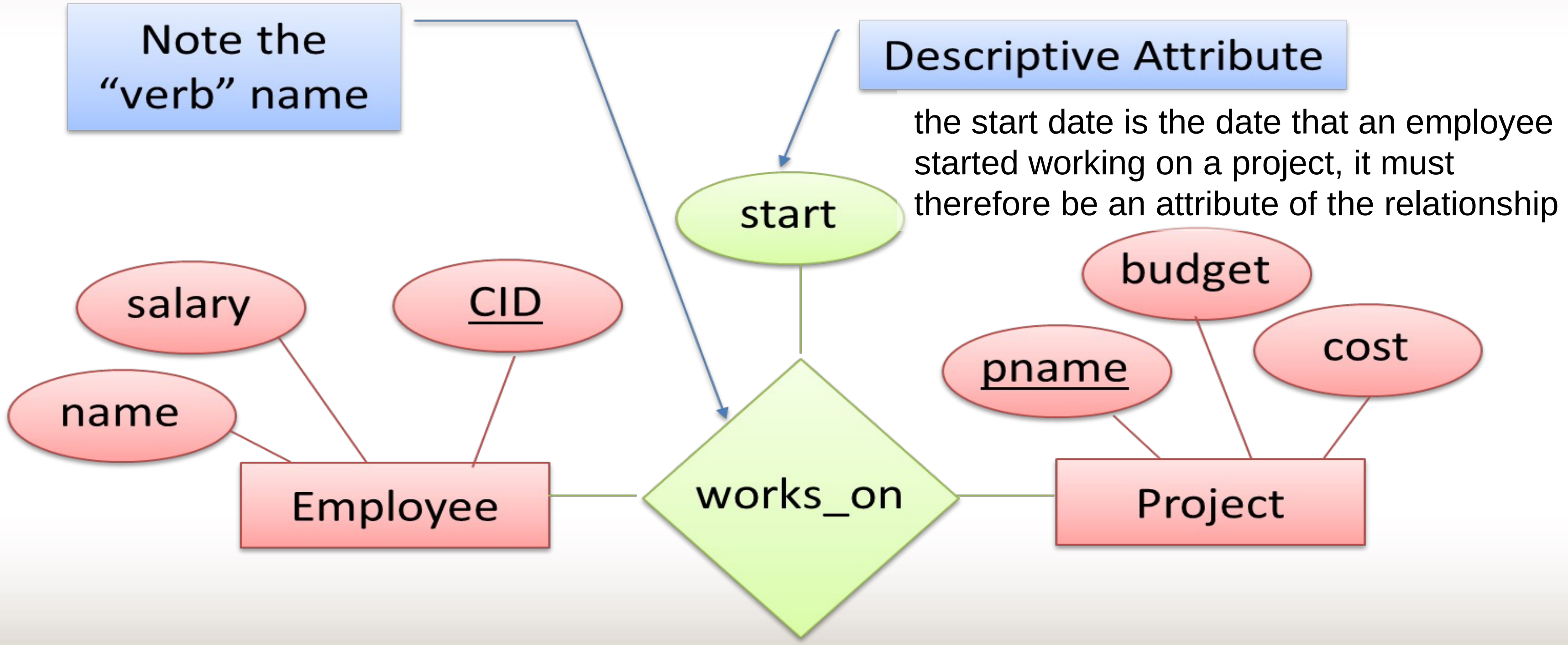


Candidate Key 1: {first, last, birthday}*

Primary Key: {CID}

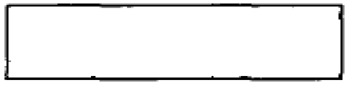
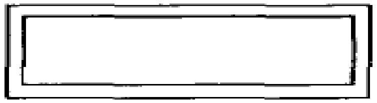
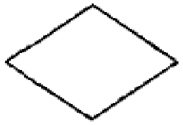
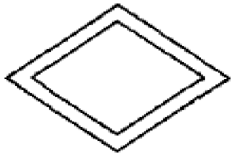
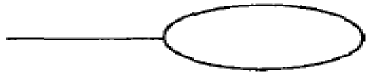
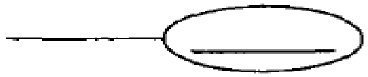
Relationship Sets

- Relationship is an *association* between two or more entities
- Relationship sets may have *descriptive attributes*.



the start date is the date that an employee started working on a project, it must therefore be an attribute of the relationship

ER-Diagram Notation For ER Schemas

Symbol	Meaning
	ENTITY
	WEAK ENTITY
	RELATIONSHIP
	IDENTIFYING RELATIONSHIP
	ATTRIBUTE
	KEY ATTRIBUTE

Example: COMPANY Database

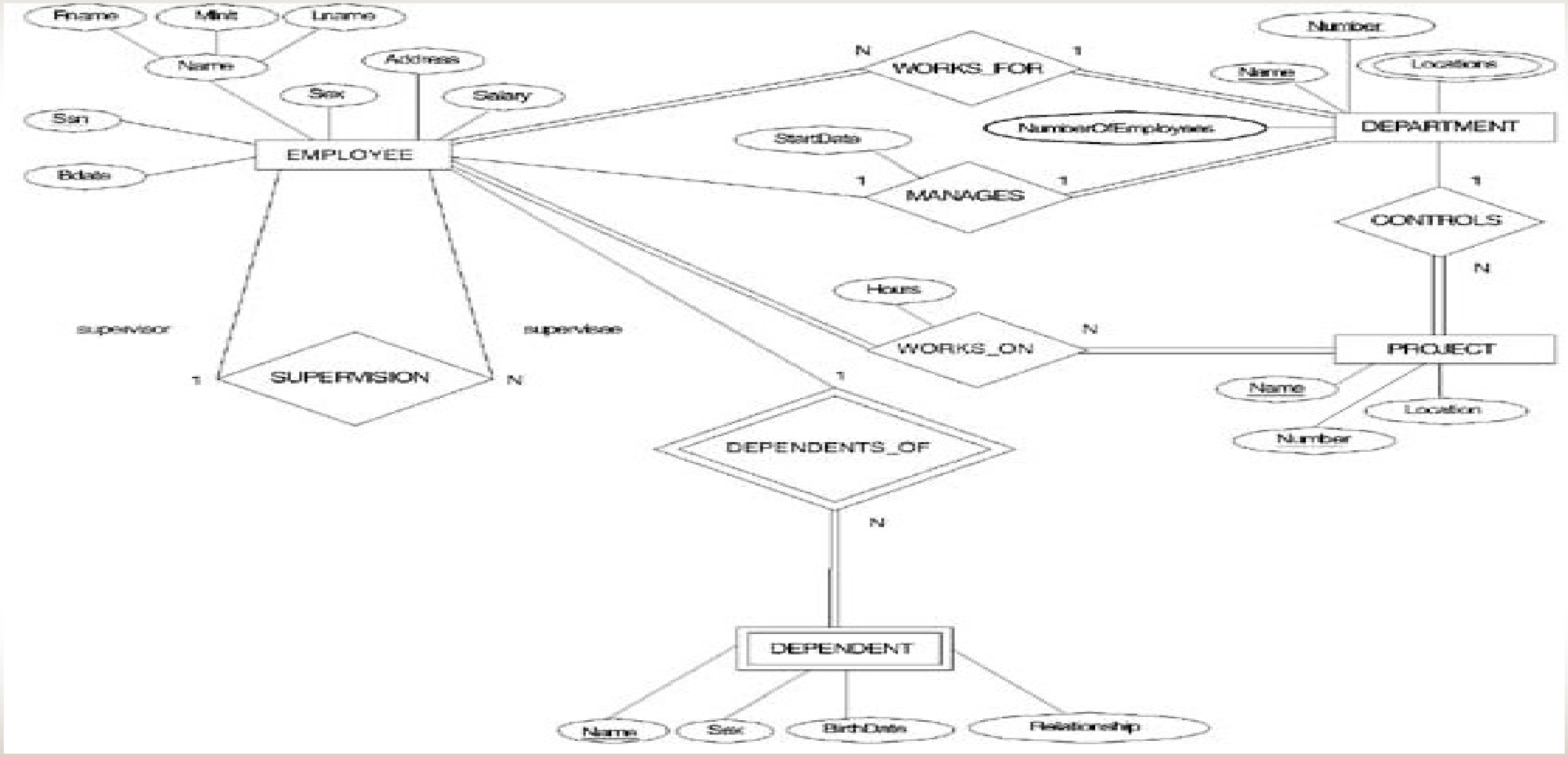
- Requirements of the Company: The company is organized into *DEPARTMENTs*.
  - Each department has a *name*, *number* and an *employee* who *manages* the department.
    - We keep track of the *start date* of the department manager.
  - Each department controls a number of *PROJECTs*.
    - Each project has a *name*, *number* and is *located* at a single location.
  - We store each *EMPLOYEE's social security number*, *address*, *salary*, *sex*, and *birthdate*.

Example: COMPANY Database

- Requirements of the Company: The company is organized into *DEPARTMENTs*.
  - 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.
  - Each employee may have a number of *DEPENDENTs*.
    - For each dependent, we keep track of their *name*, *sex*, *birthdate*, and *relationship* to employee.

Example: COMPANY Database

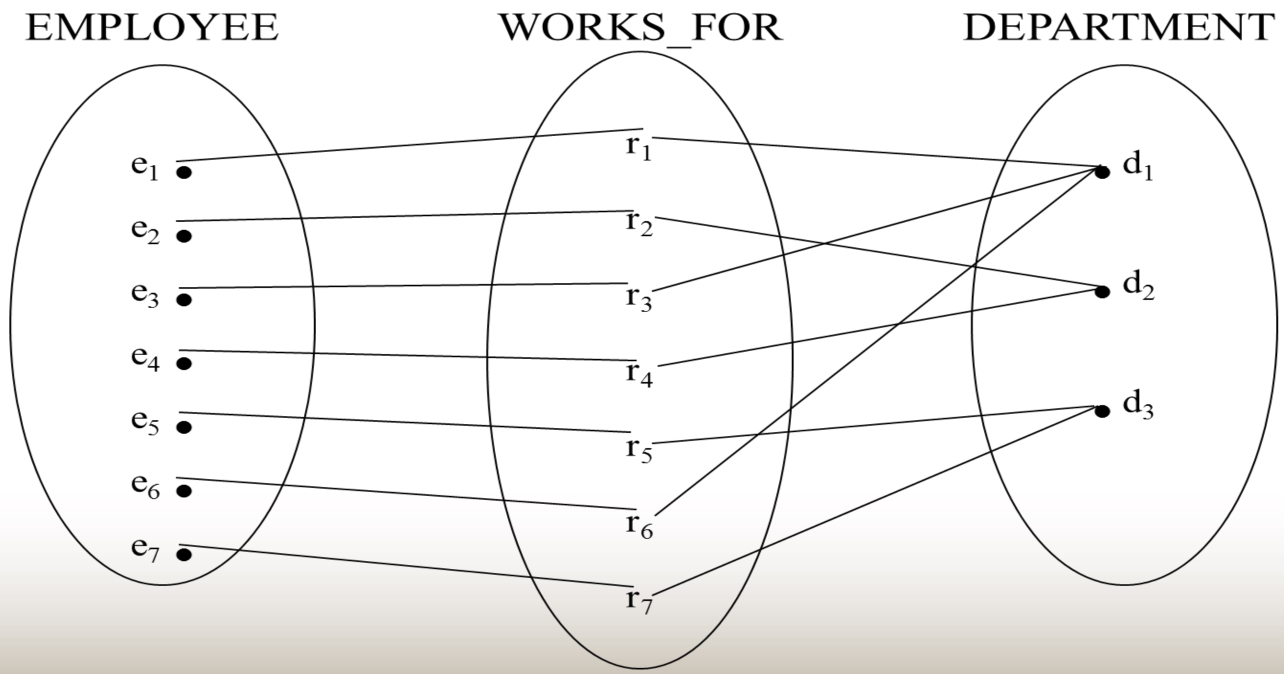
❖ ER Diagram: - the entity types are *Employee*, *Department*, *Project* and *Dependent*.



Example: COMPANY Database

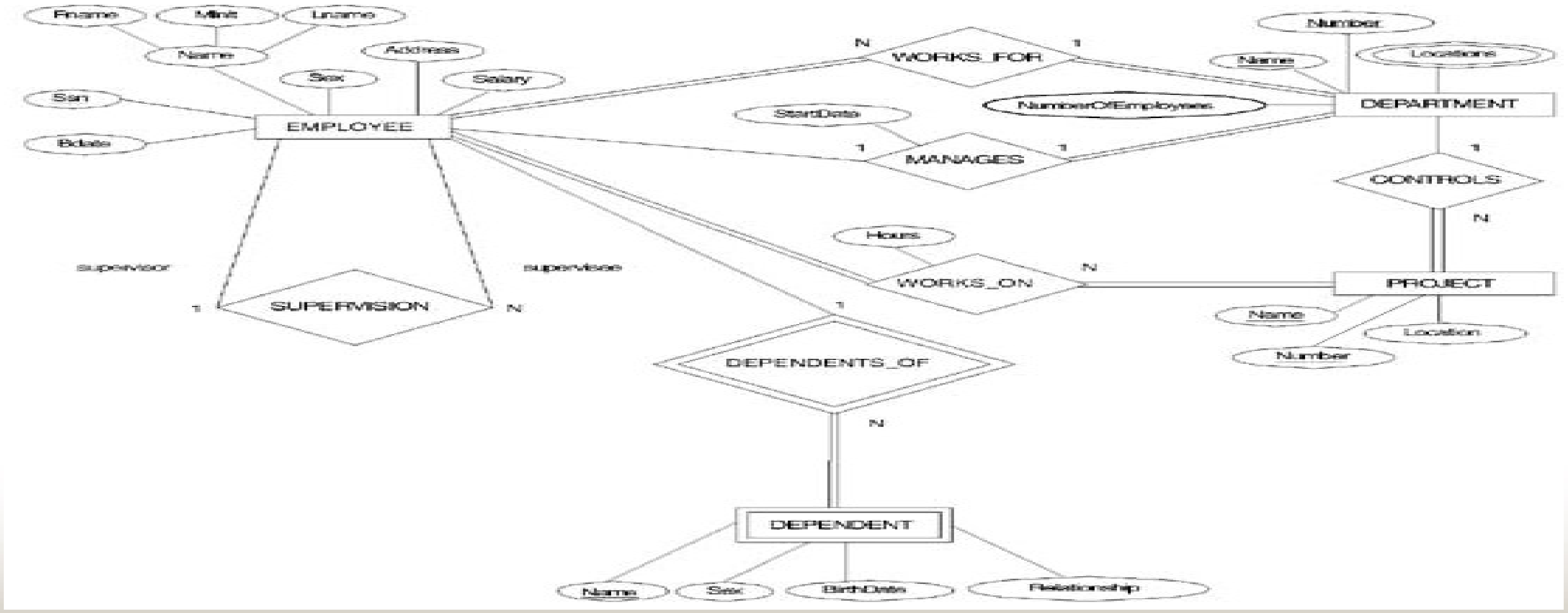
❖ ER Diagram: - the entity types are *Employee*, *Department*, *Project* and *Dependent*.

Example relationship instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT



Example: COMPANY Database

- 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.

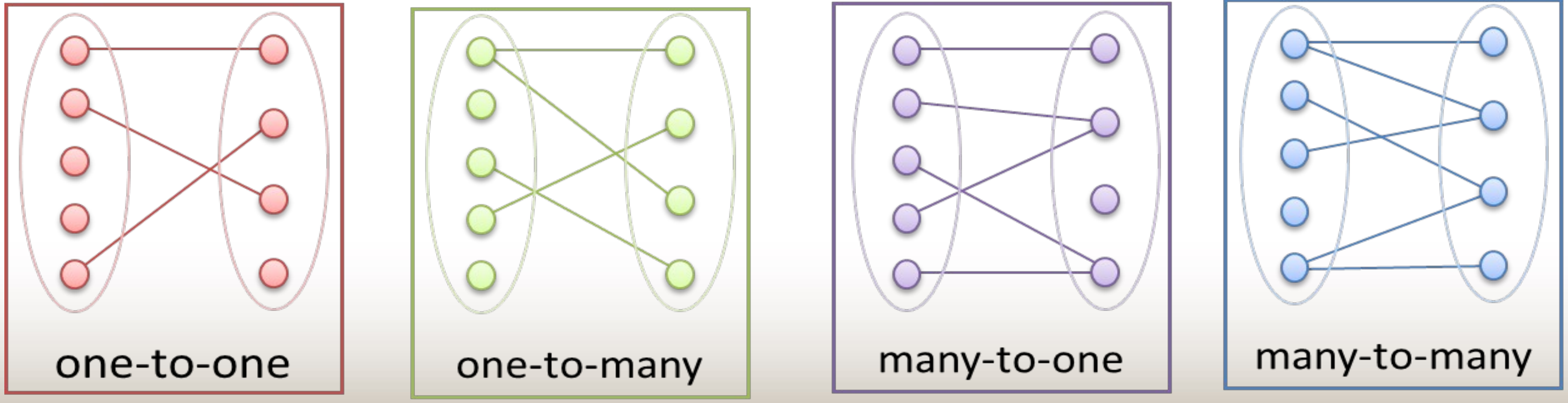


Degree of a relationship

- Degree of a relationship : deal with *how many entities participate* in it or the *number of entities participating* in a relationship:
- Among the Degrees of relationship, the following are the basic:
  - *UNARY/RECURSIVE RELATIONSHIP*: Tuples/records of a Single entity are related withy each other. Both participations are same entity type in different roles.
    - Example, *SUPERVISION* relationships between *EMPLOYEE* (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
  - *BINARY RELATIONSHIPS*: Tuples/records of two entities are associated in a relationship.
  - *TERNARY RELATIONSHIP*: Tuples/records of three different entities are associated
  - And a generalized one: *N-ARY RELATIONSHIP*: Tuples from arbitrary number of entity sets are participating in a relationship.

Cardinality of a relationship

- The number of *instances/tuples* that can be associated with a single instance from one entity in a single relationship.
- There are different types of *binary relationships*:
  - In terms of the number of relationships that the participating entities may be involved in.
- Mapping cardinalities are specified in an *ER diagram by directed* lines (arrows).

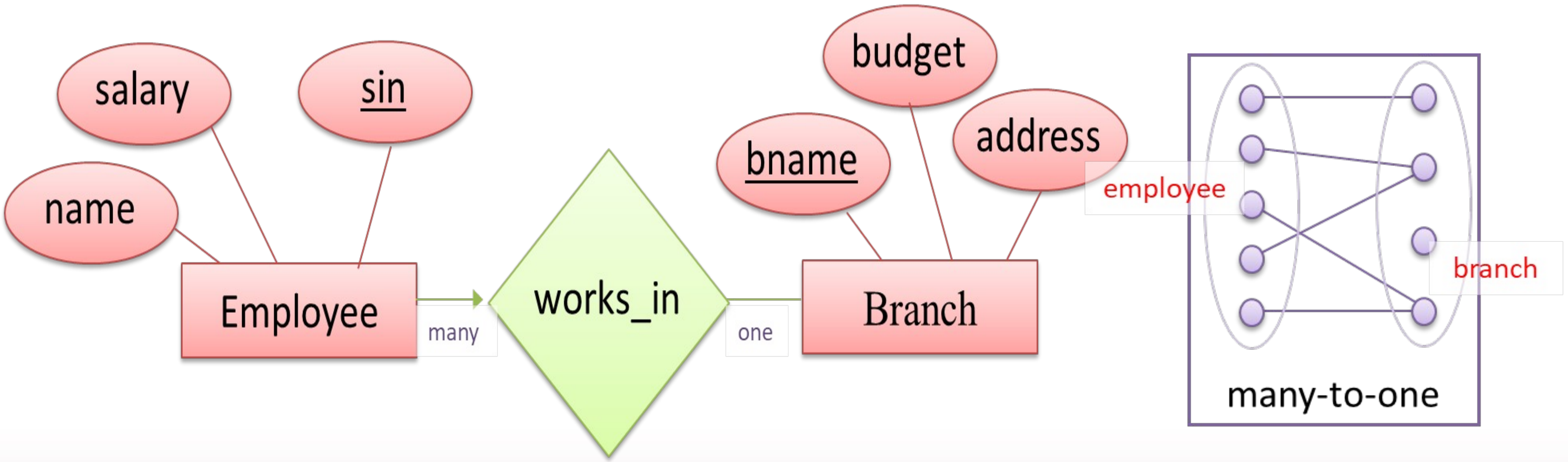


Cardinality of a relationship

- Example: Assume entity sets A and B
- One-to-one:
  - An entity in A associates with *at most one entity* in B, an entity in B associates with *at most one* entity in A.
- One-to-many (A to B):
  - An entity in A associates *with any number of entities* in B, an entity in B associates with *at most one entity* in A,
- Many-to-many:
  - An entity in A associates *with any number of entities* in B, an entity in B associates *with any number of entities* in A,

Mapping Cardinalities Example

- An employee can work *in only one branch*.
- A branch can have *many employees working in it*.



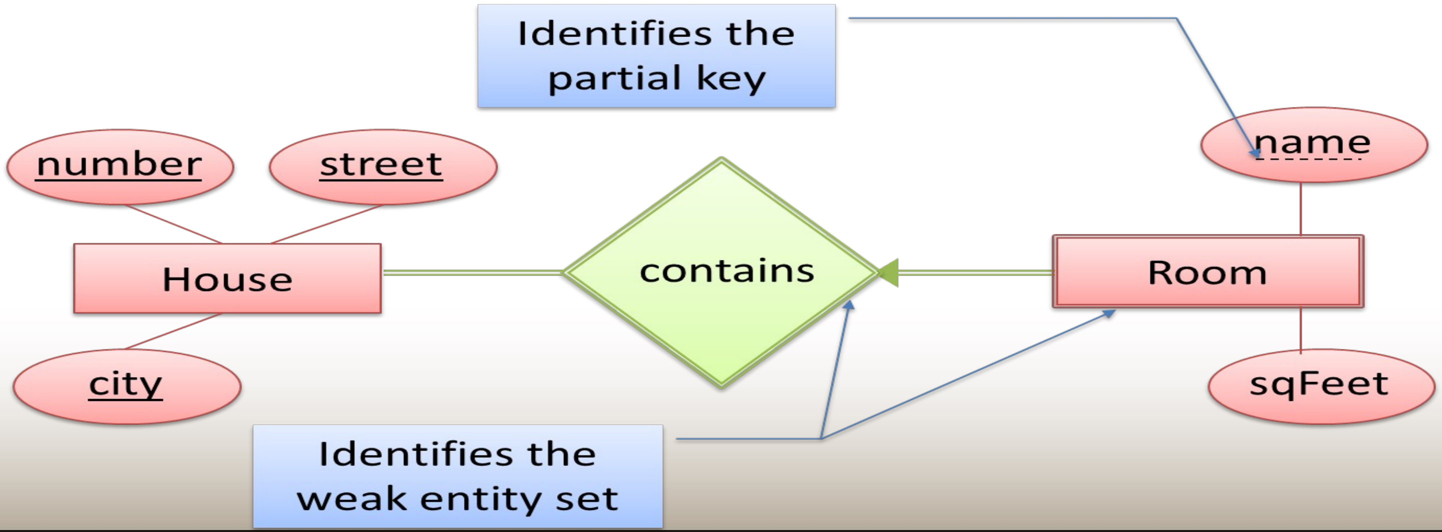
A branch can have many employees but an employee can work in only one branch

Weak Entity Types

- An entity that can not exist without the entity with which it has a relationship–it is indicated by a *double rectangle*.
- 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*.

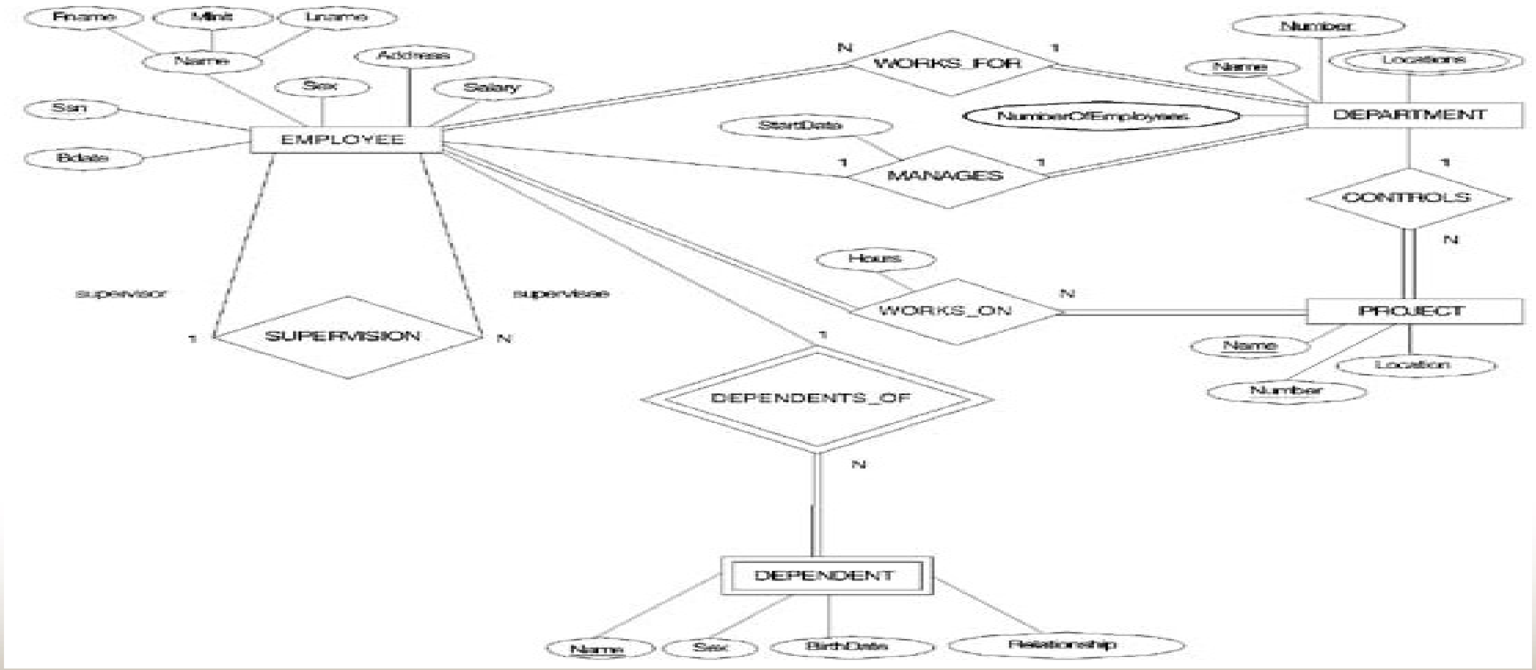
Weak Entity Types

- Weak entity sets are permitted only when:
- The *owner* and *weak entity set* participate in a one-to-many identifying relationship set and,
- The *weak entity set* must have total participation in the identifying relationship.
- Weak Entity Example:



Weak Entity Types

- *Example*: Suppose that a DEPENDENT entity is identified by the dependent’s first name and birhtdate, 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*.



Extended ER Model

- It may be useful to classify the entities in an entity set into *subclasses*:
  - Similar to subclassing in OOP.
  - Each entity in a *subclass* is also an entity in the *superclass*.

- The *attributes* of the *superclass entity* are inherited by the subclass entities:
  - The subclass entity may also have *additional attributes*.

- Class hierarchies can have *multiple level*.

Extended ER Model

- Designing Subclasses:
- The *subclass relationship* is sometimes referred to as an “ is a” relationship:
  - A “is a” B (A is the *subclass*, B the *superclass*)
  - A *specializes* B or B *generalizes* A.

- *Specialization* is the process of identifying subsets with additional attributes from an existing entity set.
- *Generalization* is the process of identifying common characteristics (attributes) of entity sets.
  - And creating a new parent entity set with those attributes.

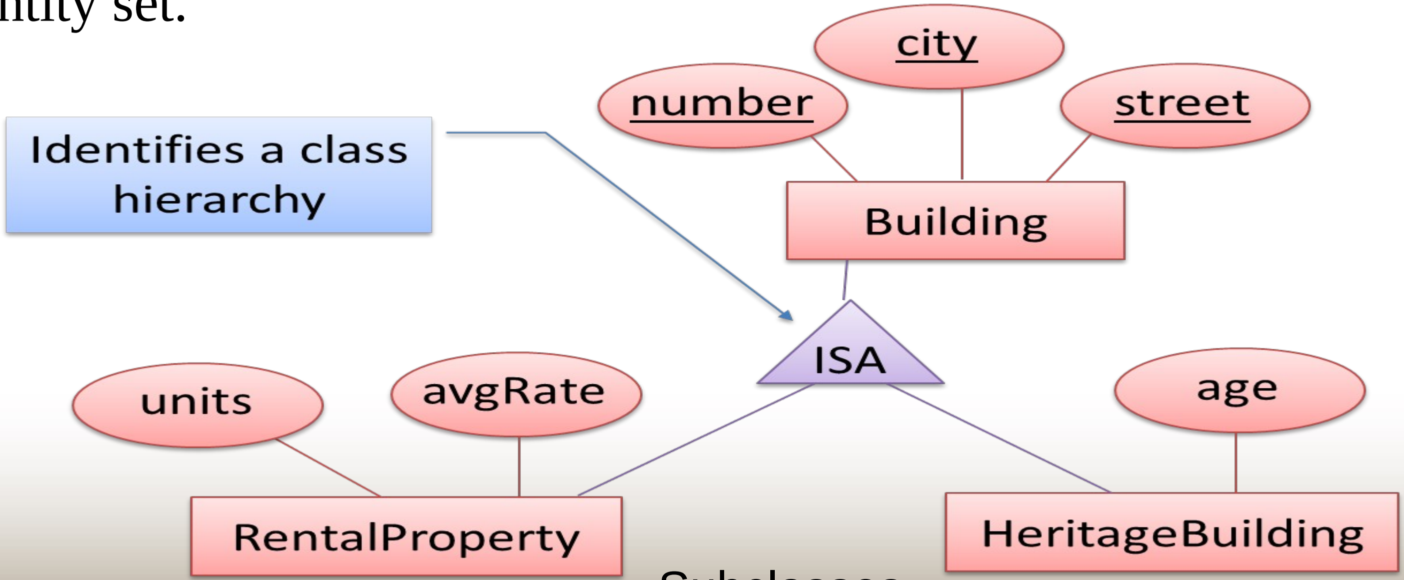
Extended ER Model

- Inheritance:
- The *attribute* of *parent entity sets* are inherited by their subclasses.
    - A subclass entity set is therefore described by *its attributes* and the *attributes of its superclass(es)*.

- Subclass entity sets *also inherit* participation in *relationship sets*.
    - Subclass entity sets can participate in *any relationships* that their *superclasses participate in*.

Why Use Class Hierarchies?

- Attributes in common don't have to be re-defined for each subclass, so that *additional descriptive attributes* can be added to subclasses.
- To identify the *set of entities that participate* in a particular relationship.
  - i.e. subclasses can be created to *identify a relationship* with another entity set.



Exercise One

- *Exercise one*: Nahom Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of 600Birr/day).
  - Each *musician* that records at Notown has an SSN, a name, an address, and a phone number.
  - Each *instrument* that is used in songs recorded at Nahom has a name (e.g., guitar, synthesizer, flute)
  - Each *album* that is recorded on the Nahom label has a title, a copyright date, a format(e.g., CD , VCD, DVD), and an album identier.
  - Each song recorded at Nahom has a title and an author.

Exercise One

- Each musician may play several instruments, and a given instrument may be played by several musicians.
- Each album has a number of songs on it, but no song may appear on more than one album.
- Each song is performed by one or more musicians, and a musician may perform a number of songs.
- Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

- *Design a conceptual schema for Nahom and draw an ER diagram for your schema*. Be sure to indicate all key and cardinality constraints and any assumptions that you make. Identify any constraints that you are unable to capture in the ER diagram and briefly explain why you could not express them.

Exercise Two

- *Exercise two*: The Prescriptions-R-X chain of pharmacies has offered to give you a free lifetime supply of medicines if you design its database. Here's the information that you gather:
  - Patients are identied by an SID, and their names, addresses, and ages must be recorded.
  - Doctors are identied by an EID. For each doctor, the name, specialty, and years of experience must be recorded.
  - Each pharmaceutical company is identied by name and has a phone number.
  - For each drug, the trade name and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the trade name identies a drug uniquely from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer.
  - Each pharmacy has a name, address, and phone number.
  - Every patient has a primary physician. Every doctor has at least one patient.

Exercise Two

- Each pharmacy sells several drugs and has a price for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to another.
- Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, Each prescription has a date and a quantity associated with it.
- Pharmaceutical companies have long-term contracts with pharmacies. A pharmaceutical company can contract with several pharmacies, and a pharmacy can contract with several pharmaceutical companies. For each contract, you have to store a start date, an end date, and the text of the contract.
- Pharmacies appoint a supervisor for each contract. There must always be a supervisor for each contract, but the contract supervisor can change over the lifetime of the contract.

- *Draw an ER diagram* that captures the above information.

Exercise Three

- Consider the following *relations for a database* that keeps track of student enrollment in courses and the books adopted for each course:
  - STUDENT(SSN, Name, Major, Bdate)
  - COURSE(Course#, Cname, Dept)
  - ENROLL(SSN, Course#, Quarter, Grade)
  - BOOK_ADOPTION(Course#, Quarter, Book_ISBN)
  - TEXT(Book_ISBN, Book_Title, Publisher, Author)
- Draw a *relational schema diagram* specifying the foreign keys for this schema.

Question & Answer

Thanks !!!