Database Management Systems (DBMS)
Introduction

Course Objectives

- Understand basic system concepts of a DBMS
- Learn theory and practice of designing, implementing and using a DBMS
- More system aspects than a specific product focus
- Learn concepts of:
  - Data models,
  - Database design,
  - Query Language,
  - Query processing,
  - Query Optimization,
  - Consistency and Integrity control,
  - Transaction management,
  - Database recovery and fault tolerance
  - Storage structure, and
  - Database security.

Syllabus Review – Course Organization

Part 1
Model & SQL

- ERD Models
- Relational Data Model
- Relational Algebra
- SQL

Part 2
Optimization

- Normalization & Database Design
- Query Performance & Optimization
- Performance Indexing

Part 3
Transactions

- Transaction Management
- Concurrency Control
- Fault Tolerance & Recovery

Terminology

- ***Database System = Database + DBMS + Applications***
  - ***What is a Database?***
    - Collection of related data items
    - Each item has a name (it is addressable) and **may** have a value
  - ***What is a DBMS?***
    - A Database Management System (DBMS) is the software system that manages one or more databases
  - ***What is an Application?***
    - A program that we write to work with the database using a DBMS
    - It is usually written in a high level language like VB, Java, etc.
    - Our focus in this class is on OLTP applications not OLAP applications

Terminology

- Data Modeling Language
  - Defines the general rules for the specification of the ***structures*** of data and the ***operations*** allowed on the data.
- Conceptual Data Model (CDM)
  - The application of the modeling rules to a particular environment.
- Logical Data Model
  - Is the representation of a data model for a specific database type. For example, in Relational the model is shown as the definitions of tables, keys, etc.
- Physical Data Model (PDM)
  - The logical model Shown as a set of Data Definition Language Statements (DDL) for a specific DBMS such as Oracle or SQL Server. This is what we call a database schema.
- Database
  - A database is a collection of data items, structured according to the PDM .

Why Use A DBMS?

- Database management systems (DBMSs) form a layer of software between the users and the data stored on the disk
- They provide for the following:
  - Ease of use
  - Hide the physical structure of the database from us
  - Provide a high level view of the data so that it is easier to work with the database
  - Provide for security of the contents of the database
  - Provide for facilities to backup and to recover data when there are problems
  - Etc.

What is in a Database? Physical View

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John		Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin		Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia		Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer		Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh		Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce		English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad		Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James		Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
		Houston
		Stafford
		Bellaire
		Sugarland

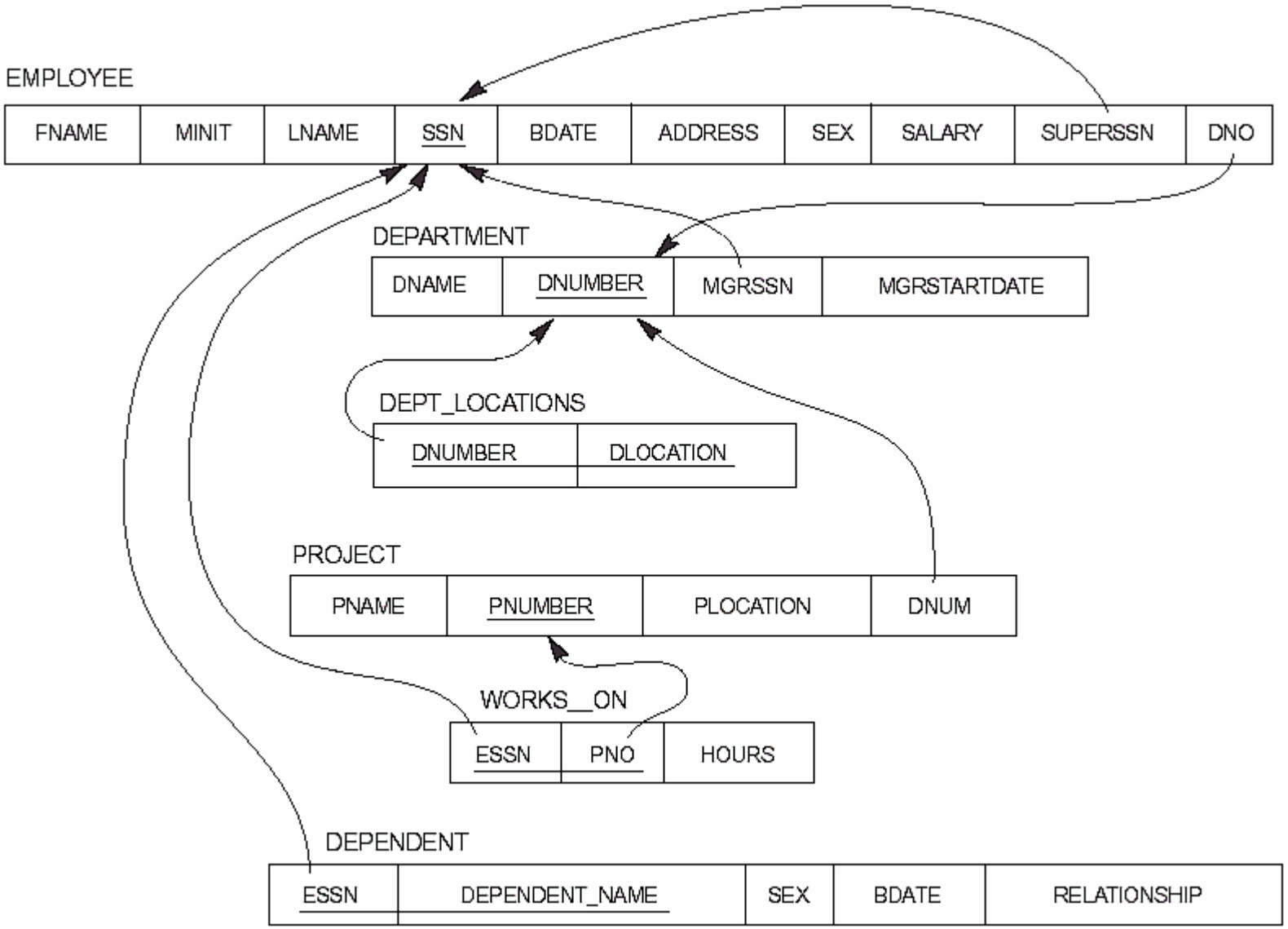
DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

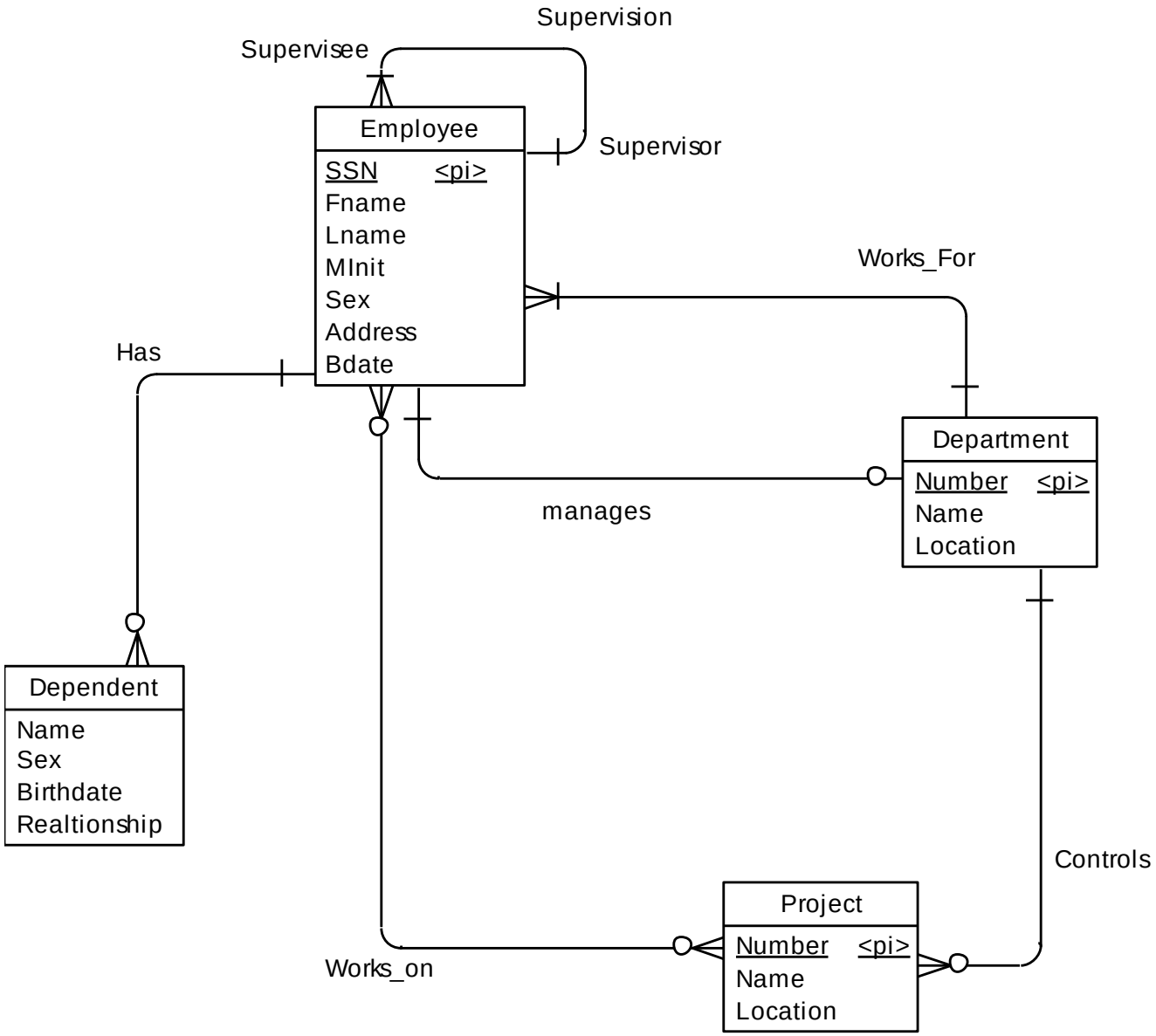
PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

What is in a Database? Logical View



What is in a Database? Conceptual View



Database Life Cycle

- ***Database Life Cycle***
  - **Step 1: Develop the Blueprint (CDM)**
    - **Define and Model the Environment**
      - *Objects* – What is important to us
      - *Constraints & Relationships* – Rules & connections between objects
  - **Step 2: Generate the Database**
    - **Create the Database Schema**
      - Generate the PDM and the DDL
      - Build the database tables, indices, etc. using the DDL
  - **Step 3: Load (Database)**
    - **Data Population - Load the data**
      - Insert, Migrate, or do a mass initial load
  - **Step 4: Use**
    - **Data Manipulation - Retrieve and manage**
      - Retrieve, insert, update, and delete data using Data Manipulation Language (DML).

Entity Relationship (ER) Modeling Concepts

ER Model

Conceptual Data Model

Objects

Relationships between Objects

Entity Relationship Diagram (models your business)

ER applies to this layer

Logical Data Model

Relational Schemas

Constraints

Dependencies

Relational Model (a logical model of your business)

Physical Data Model

Physical Database Objects

Create Table, Index, etc

Physical Model (Database Specific)

Entity Relationship Model

- Entities:
  - Perception of basic objects of the real world
- Attributes:
  - characteristics of an object of the real world
- Relationships:
  - associations of real world objects with each other.

ER Modeling

- An Entity is a perception of an object in the real world
  - Each entity type is represented by a set of attribute types
  - Each attribute type represents one characteristic of the real world object
- Example:
  - Employee(Name, Address, SSN, EmpID, Sal, DoB) is an entity type that talks about all employees
    - Instance: (Name: John Doe, SSN: 890-90-1234) talks about one specific employee
    - In order to be able to specify one specific employee, we need to have a way of **identifying** every entity of a given type (SSN or EmpID in our example).

Entity Relationship Model

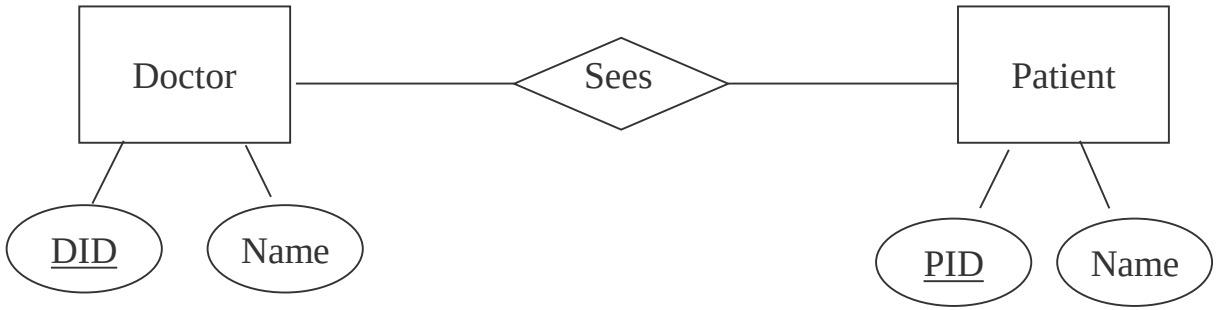
- Examples of entity types:
  - Employee
  - Customer
  - Project
  - Person
  - Automobile
  - Holiday
  - Pet
- Some of these entity types are physical (Employee) and some are logical (Project)

Entity Relationship Model

- Relationship Types
  - All relationships that have the same characteristics belong to a given "Relationship Type"
- Degrees of Relationships
  - Binary relationships: associate two entity types
  - Ternary relationships: Associate three entity types
  - Nary relationships: associate N entity types
  - Examples:
    - Binary Relationships:
      - *Employee* works for C*ompany*
      - *Person* Has *Pet*
      - *Person* Marries *Person*
    - Relationship of Degree Three: *Employee* uses *Tool* to work on *Project*
    - Relationship of Degree Five: A *Priest* marries a *Man* and a *Woman* at a *Church* in presence of a *witness*

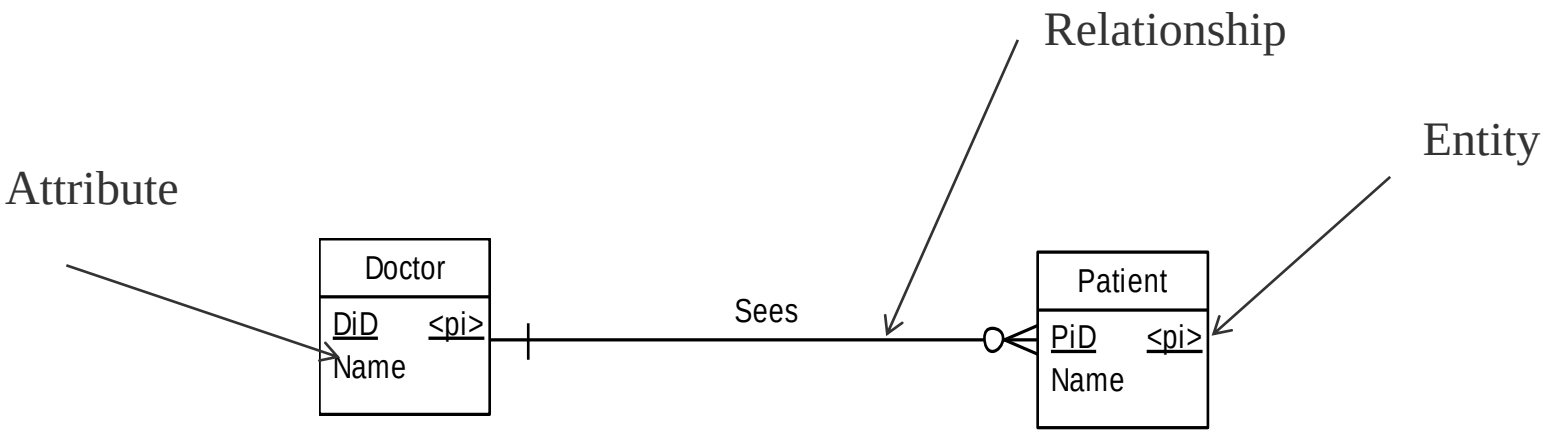
Entity Relationship Model

- Notation
  - Your book uses the notation introduced by Peter Chen (the inventor of the ER modeling)
  - In his notation:
    - Entities types are shown as ***squares***
    - Attributes types are shows as ***circles***
    - Relationships types are shown as ***Diamonds***
    - Example:



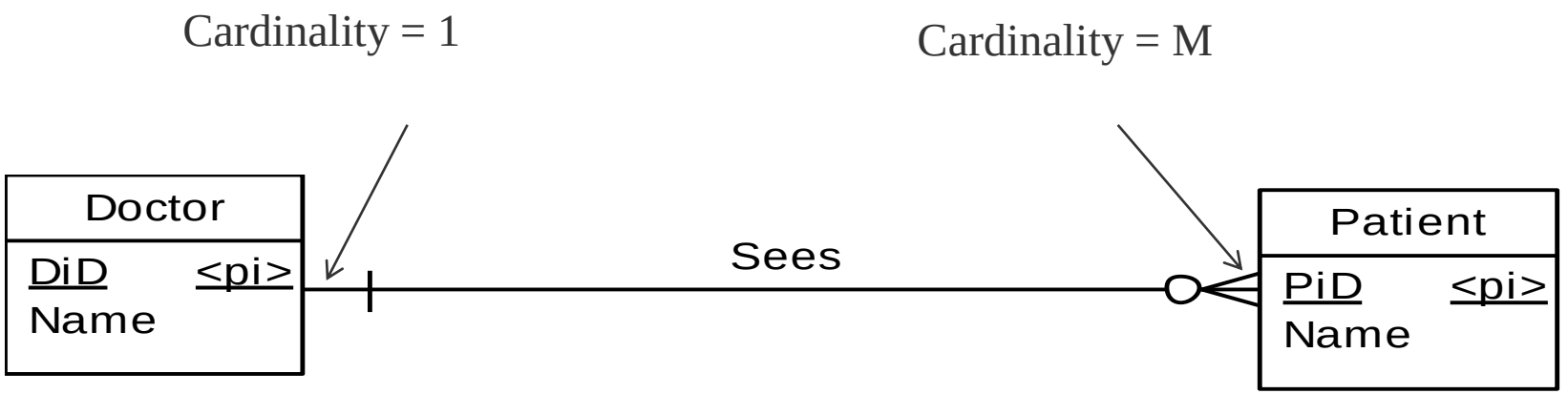
Entity Relationship Model

- Notation
  - We will use the Martin notation
    - Commonly used by data modeling software products today
  - In Martin's notation:
    - Entities types are shown as ***squares***
      - ***The name of the entity is on top***
    - Attributes names are shown under the name of the entity
    - Relationships types are shown as ***Lines***
    - Example:



Entity Relationship Model

- Mapping Cardinality
  - Mapping cardinality is a constraint that specifies the number of entities to which a given entity can be associated with.
    - For example:
      - A patient only sees one doctor. Cardinality = 1
      - A doctor can see many patients. Cardinality = M

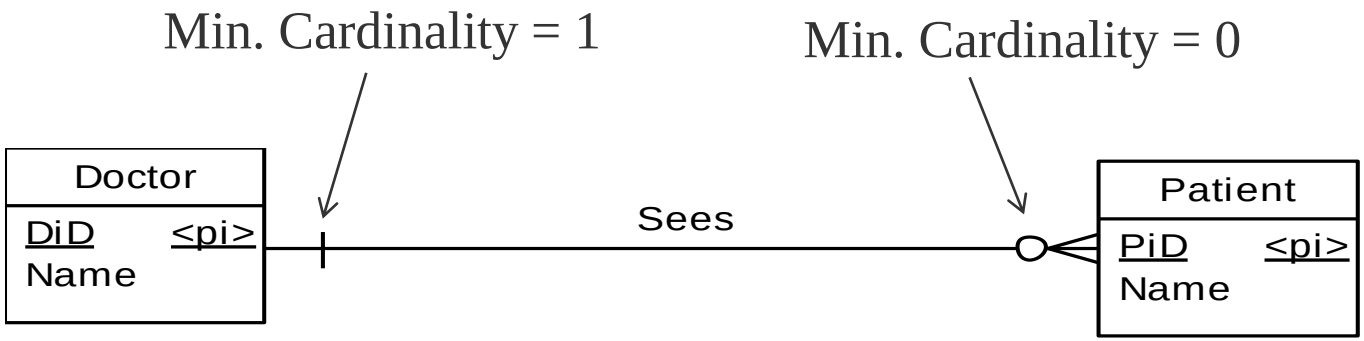


Entity Relationship Model

- Mapping Cardinality
  - 1 and many cardinalities constraints how entities participate in a relationship
  - Minimum and Maximum cardinalities further constrain the relationship
    - Maximum cardinality provides an upper bound for the cardinality
    - Minimum cardinality provides a lower bound for the cardinality

Entity Relationship Model

- Mapping Optionality
  - Example:
    - A patient only sees one and only one doctor in any visit.
      - » Cardinality = 1
      - » Minimum Cardinality = 1
      - » Maximum Cardinality = 1
    - A doctor can see many patients, but may see no patients
      - » Cardinality = M
      - » Minimum Cardinality = 0
      - » Maximum Cardinality = M



Entity Relationship Model

- Let’s suppose you want to model a bank
  - Identify the entity types for the bank.
    - Customer
    - Account
  - The attributes of each one of the entity types
    - Account: account number, balance, type
    - Customer: Customer ID, SSN, Name, Address

Entity Relationship Model

- Cardinality - For the bank model you created, suppose:
  - A customer must have at least one account but can have more than one account.
  - An account must have at least one owner and at most two owners.
  - Show this information on the model

Entity Relationship Model

- Entity Identification – **Keys**
  - Each object of the real world is uniquely identifiable. Therefore, objects have a key.
  - Entities that represent these objects must also be uniquely identifiable by the values of one or more of their attributes
    - We call the smallest collection of the attributes that uniquely identify an entity, a **Candidate Key**
    - Amongst the candidate keys, we choose one as what we call the **Primary Key**

Entity Relationship Model

- Keys – Example:
  - Entity Employee (SSN, EmpID, Name, Address)
    - Name cannot be a key
    - EmpID can be a key
    - SSN can be a key
    - SSN and Name can also be a key
      - But they form a super key since a subset of the attributes is a key
    - SSN and Name and Address can also be a super key
    - Employee has two candidate keys
      - SSN and EmpID
    - We choose
      - EmpID as the Pkey
      - SSN as a candidate key
    - Why?

Entity Relationship Model

- Strong Entities
  - Can exist on its own.
  - Strong entities have a Pkey
    - Employee is a strong entity

- Weak Entities
  - Depends on the presence of another entity
  - Weak entities do not have a Pkey (by themselves)
    - They depend on a strong entity for their identification/existence

Entity Relationship Model

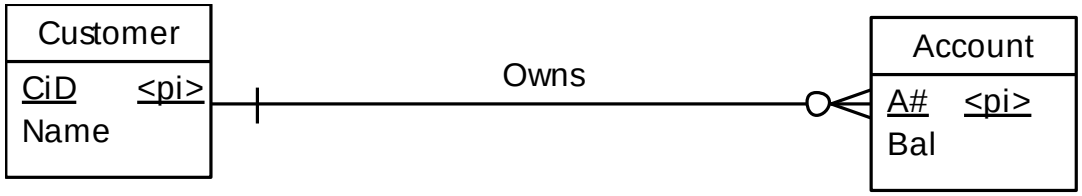
- Dependencies
  - Identification dependency (Example)
    - On earth, country is a **strong** entity
    - Within a country, State is a **strong** entity
      - States names are unique in the United State
    - City is a **weak** entity
      - To uniquely specify any city in a state, we need to combine city_name and state_name
    - To globally identify any city, we need to combine country_name, State_name, city_name
  - Context is important in identification

Entity Relationship Model

- Dependencies
  - Identification dependency (Example)
    - Database name is unique only within a DBMS instance
    - Within a database, table names are unique
    - Columns are unique only within a table
    - To identify a column, we need:
      - database_name, table_name, column_name

Entity Relationship Model

- Dependencies
  - In addition to identification dependency, there is another type of dependency called, **Existence Dependency**
    - Example:
      - Customer is a **strong** entity
      - Account is also a **strong** entity



      - But existence of an account depends on the existence of a customer who owns the account
      - In this case,
        - The customer entity is called the **dominant** entity
        - The account entity is called the **dependent** entity

Entity Relationship Model

- Data Types
  - The values that can assign to the attributes of an ER model are constraint by the data types that we associate with them.
  - There are two data types:
    - System Defined Data Type - Data types that the DBMS system supports automatically
      - Date,
      - Integer,
      - Real or Decimal(precision, Scale),
      - Char(length),
      - Varchar(length),
      - Boolean,
      - Etc.
      - Examples:
        - » IS_KEY Boolean,
        - » Name Varchar(30),
        - » Salary Decimal (10, 2),
        - » EID Number(8).

Entity Relationship Model

- Data Types
  - User Defined Data type (Domains)
    - Data types that a user defines based on the system defined data types
      - We may define Money_paid having decimal (10,2)
      - Then we can associate Money_paid as the data type to Salary and Commission and price as
      - Salary Money_paid
      - Commission Money_paid
      - Price Money_paid
    - A user defined data type is sometimes called a Domain

Entity Relationship Model

- Attribute Types
  - Simple attributes
    - Attributes that are atomic – **cannot** be decomposed
      - SSN, Weight, Height, Color, Sex
  - Composite attributes
    - Attributes that **can be** decomposed into other component attributes
      - Address is composed of (Street, City, Zip, State)
      - Name is composed of (First, MI, Last)
      - DOB is composed of (Month, Day, Year)

Entity Relationship Model

- Attribute Types
  - Single valued attributes
    - Attributes that can only have one value at any point in time
      - Age
      - Sex
      - Height
  - Multi-valued attributes
    - Attributes that can have multiple values at any point in time
      - Degree (High School Diploma, BS, MS, PhD)
      - Address (Home address, work address)
      - Telephone number (Home phone, Work Phone, Cell Phone)

Entity Relationship Model

- Attribute Types
  - Base attribute
    - An attribute whose value is stored in the database
  - Derived (Calculated) attribute
    - An attribute whose value can be derived either from other attribute(s) of the entity or the entity's relationship with other entity(ies).
  - Example
    - DOB is a base attribute since it must be stored
    - Age is a derived attribute since it can be calculated from DOB as:
      - Age = Today().year – DOB.year
    - No-of-Emps of a Department can be calculated by counting the number of Employees who are related to the department

Entity Relationship Model

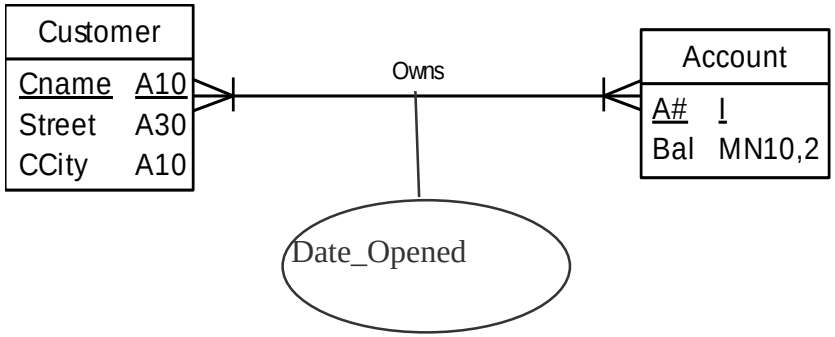
- Attribute Types
  - Null Attributes
    - Attributes that can have no value.
      - Null is different from zero or blank. It is Null!
      - To work with Null attributes, we need special type of functions such as “Is_Null” or “Is_not_Null”, which are Boolean functions
    - Example
      - Height attribute of the Employee entity type will be defined as NULL if we do not care about (or do not have) the height of employees

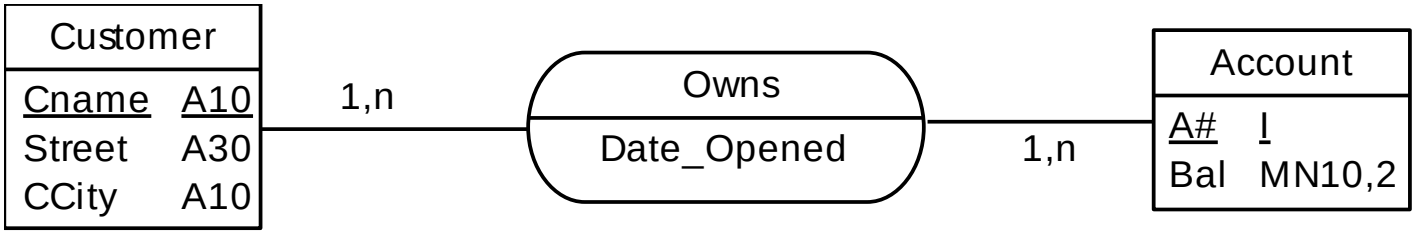
Entity Relationship Model

- Relationships with Attributes
  - In ER, we can also attach an attribute to a relationship
    - Example:
      - An employee works on one or more project
      - A project is worked on by one or more employees
      - Question:
        - » Where do we put the attribute that indicates the number of hours per week that an employee works on each project?
      - ***In general, semantics of the attribute and cardinality of the relationship determine where the attribute belongs***

Entity Relationship Model

- Relationships with Attributes
  - Data modeling tools may not have the capability of drawing relationships with attributes
    - PowerDesigner uses a special type of entity called an associative entity to represent the relationship





Lab

- In your company:
  - An employee must work for one and only one dept
  - A dept must have one or more employees
  - A dept can run one or more projects, but it does not have to.
  - A project must have at least one department that sponsors it but it can belong to more. Each department that participates in a project has a budget for the project.
  - Each dept must have a mgr.
  - Each employee must report to a supervisor
  - An employee may be a supervisor for one or more employees
  - Employees can work on one or more projects for a given number of hours per week
  - A project must have at least one or more employees working on it
- Draw an ER diagram to represent this information