Database Management Systems

Introduction

Suresh Jamadagni

Dept of Computer Science and Engineering

Text book and Slides Credit for all PPTs of this course

Text Book

- **Fundamentals of Database Systems**, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

Slides Credit for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the text book

Database Management Systems

Course Logistics: Evaluation policy

Course Name:	Database Management Systems	**Code:**	UE20CS301
Credits:	5	**Contact Hours:**	105
Session Break-up		**Assessment Scheme**	
Lectures	56	ESA	50
Lab hours	22	ISA	30
Lab quiz	2	Lab work	4
Unit Review/Assignment	5	Lab Quiz	4
CBT	5	Mini Project	7
Mini Project	15	Unit Review	5
Total	**105**	**Total**	**100**
		Hands-on sessions	

Database Management Systems

Definitions

- **Data**
  - Known facts that can be recorded and that have implicit meaning
- When data is processed, organized, structured or presented in a given context, so as to make it useful, it is called **Information**.
- Data is a collection of facts, while **information** puts those facts into context.
- While data is raw and unorganized, information is organized.
- **Information** is utilized by humans in some significant way (such as to make decisions, forecasts etc).
- Data doesn’t depend on information. Information depends on data

Database Management Systems

Definitions

- **Database**:

  - A collection of related data

    - represents some aspect of the real world, sometimes called the miniworld or the universe of discourse (UoD)
    - logically coherent collection (not a random collection) with some inherent meaning.
    - designed, built and populated for a specific purpose
    - can be of any size and complexity

Database Management Systems

Definitions

- **Database Management System**:
  - General-purpose **software system** that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications
    - **Defining** a database involves specifying the data types, structures, and constraints of the data to be stored in the database.
    - **Constructing** the database is the process of storing the data on some storage medium that is controlled by the DBMS.
    - **Manipulating** a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
    - **Sharing** a database allows multiple users and programs to access the database simultaneously

Database Management Systems

Why study databases?

- Databases help us
  - Store it (file structures, disk management)
  - Understand it (data models)
  - Keep it secure (security, recovery)
  - Find it and use it (query languages, concurrency control and data analysis tools)
- DBMSs are a key part of most applications
  - banking, making reservations, purchasing, borrowing, schedules, browsing online catalogues, video on demand, interactive maps and images, employee information, web servers

Applications

Traditional Applications:

- Numeric and Textual Databases

More Recent Applications:

- Multimedia Databases
- Geographic Information Systems (GIS)
- Data Warehouses
- Mobile databases
- Real-time and Active databases
- In-memory databases

Database Management Systems

Recent Developments (1)

- Social Media started capturing a lot of information about people and about communications among people - posts, tweets, photos, videos in systems such as:

  - Facebook

  - Twitter

  - Linked-In

- All of the above constitutes data

- Search Engines- Google, Bing, Yahoo : collect their own repository of web pages for searching

Database Management Systems

Recent Developments (2)

- New Technologies are emerging from the so-called non-database software vendors to manage vast amounts of data generated on the web.
- Big Data storage systems involving large clusters of distributed computers
- NoSQL (Not Only SQL) systems
- A large amount of data now resides on the “cloud” which means it is in huge data centers using thousands of machines.

Database Management Systems

Impact of Databases and Database Technology

- Businesses:

  Banking, Insurance, Retail, Transportation, Manufacturing

- Service Industries:

  Financial, Real-estate, Legal, Electronic Commerce, Healthcare,

- Social Networks, Environmental and Scientific Applications, Medicine and Genetics

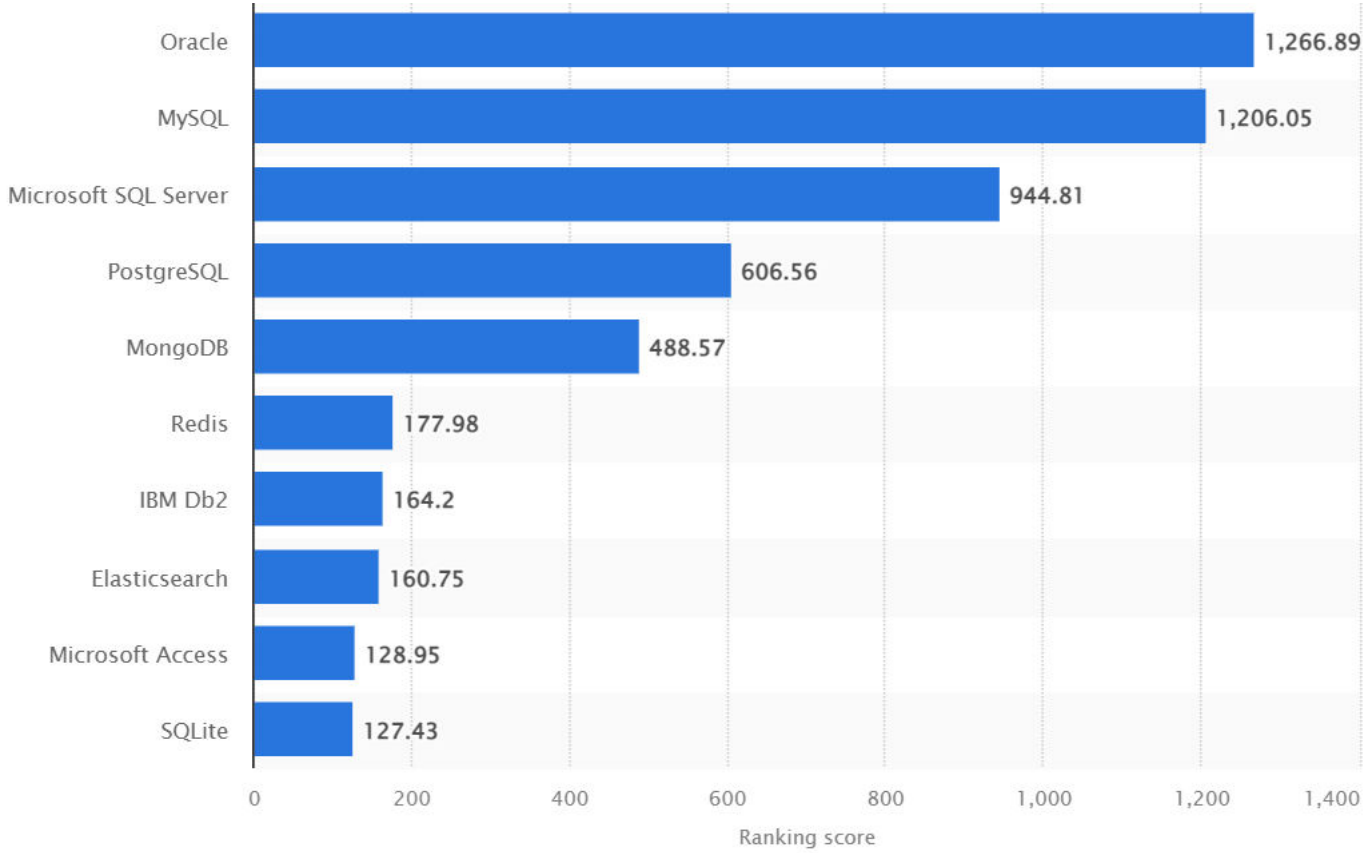
- More recently:
  - Education : Resources for content and Delivery

- Personalized Applications:

  Based on smart mobile devices

Database Management Systems

Popular DBMSs



Database Management Systems

Database system environment

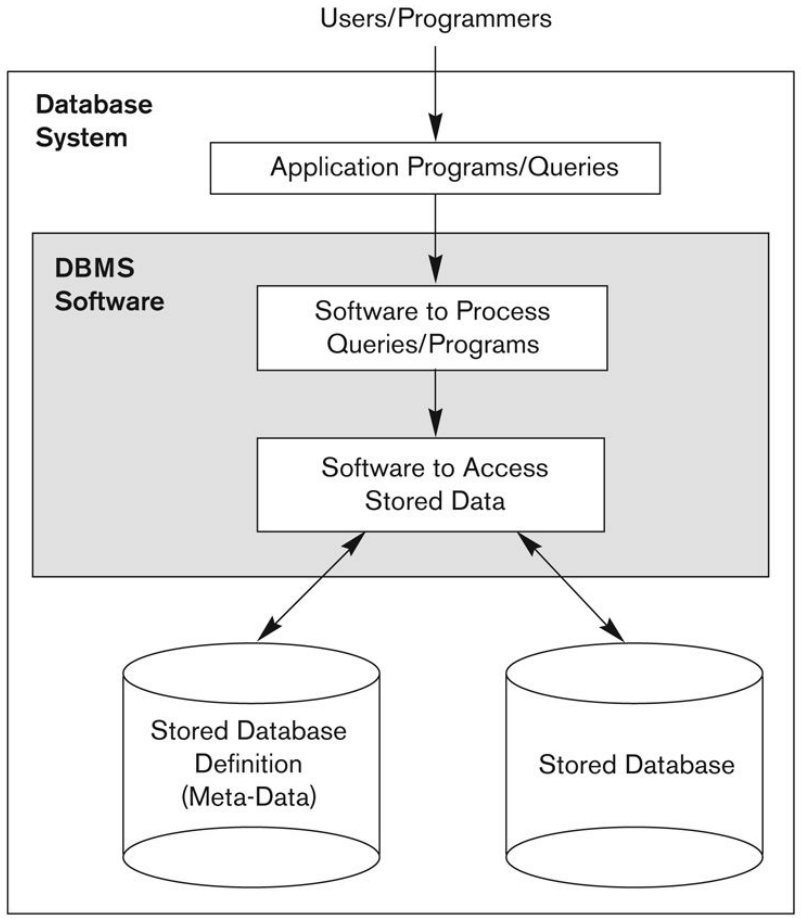


Figure 1.1
A simplified database system environment.

Database Management Systems

Example of a simple database

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Database Management Systems

Characteristics of the Database Approach

- Self-describing nature of a database system
  - The database system contains not only the database itself but also a complete definition or description of the database structure and constraints (Metadata).
- Insulation between programs and data, and data abstraction
  - The structure of data files is stored in the DBMS catalog separately from the access programs (program-data independence)
- Support of multiple views of the data
  - DBMSs support different types of users, each of whom may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.
- Sharing of data and multiuser transaction processing
  - A multiuser DBMS, allows multiple users to access the database at the same time. It ensures that data changes are effected in a controlled manner so that the result of the changes are correct

Database Management Systems

Advantages of using a Database system

- Controlling redundancy in data storage.
- Sharing of data among multiple users.
- Restricting unauthorized access to data
- Providing persistent storage for program Objects
  - E.g., Object-oriented DBMSs make program objects persistent
- Providing Storage Structures and search techniques for efficient Query Processing
- Providing backup and recovery
- Providing multiple interfaces to different classes of users.

Database Management Systems

Advantages of using a Database system

- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Drawing inferences and actions from the stored data using deductive and active rules and triggers
- Potential for enforcing standards
- Reduced application development time
- Flexibility to change data structures
- Availability of current information
- Economies of scale

Database Management Systems

Brief history of Data management

- Hierarchical and Network models
- Relational model
- Object-oriented model
- Data on the Web and E-commerce Applications
- Special functionality added to DBMSs in the following areas:
  - Scientific Applications – Physics, Chemistry, Biology - Genetics
  - Earth and Atmospheric Sciences and Astronomy
  - XML (eXtensible Markup Language)
  - Image Storage and Management
  - Audio and Video Data Management
  - Data Warehousing and Data Mining
  - Spatial Data Management and Location Based Services
  - Time Series and Historical Data Management

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Data Models

Suresh Jamadagni

Dept of Computer Science and Engineering

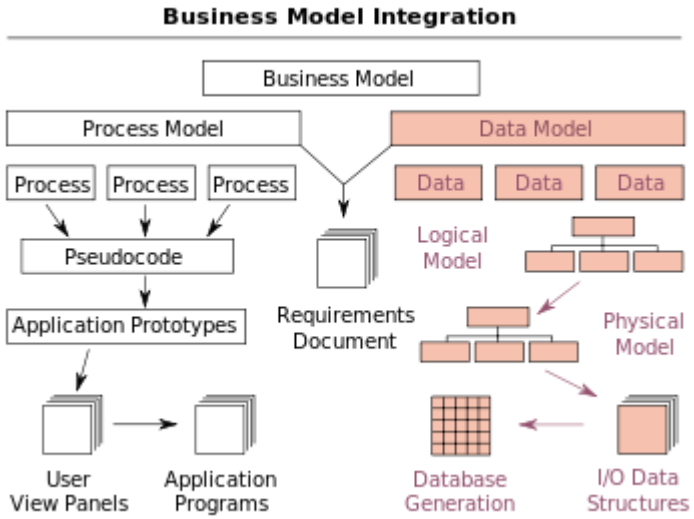
Database Management Systems

Data Model

- A collection of concepts that can be used to describe the structure of a database
- Provides the necessary means to achieve data abstraction
- The term **data model** can refer to two distinct but closely related concepts.
- It refers to an abstract formalization of the objects and relationships found in a particular application domain

Example:

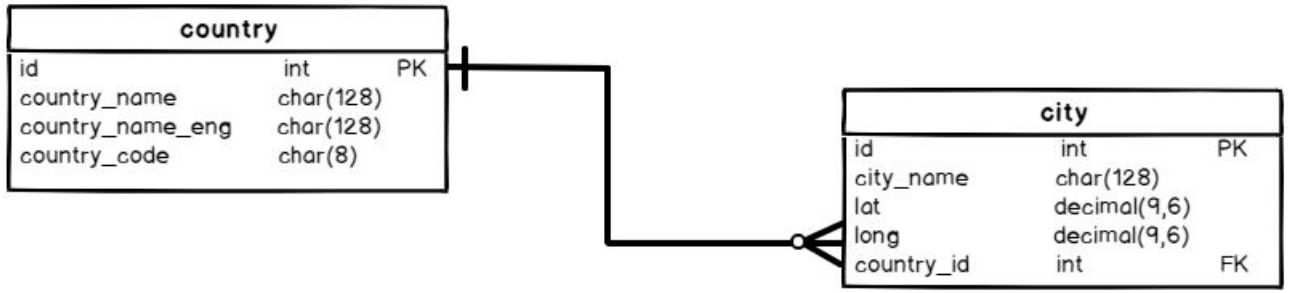
1. Customers, products and orders in a manufacturing organization
2. Students, courses and instructors in a University



Database Management Systems

Data Model – Structure and Constraints

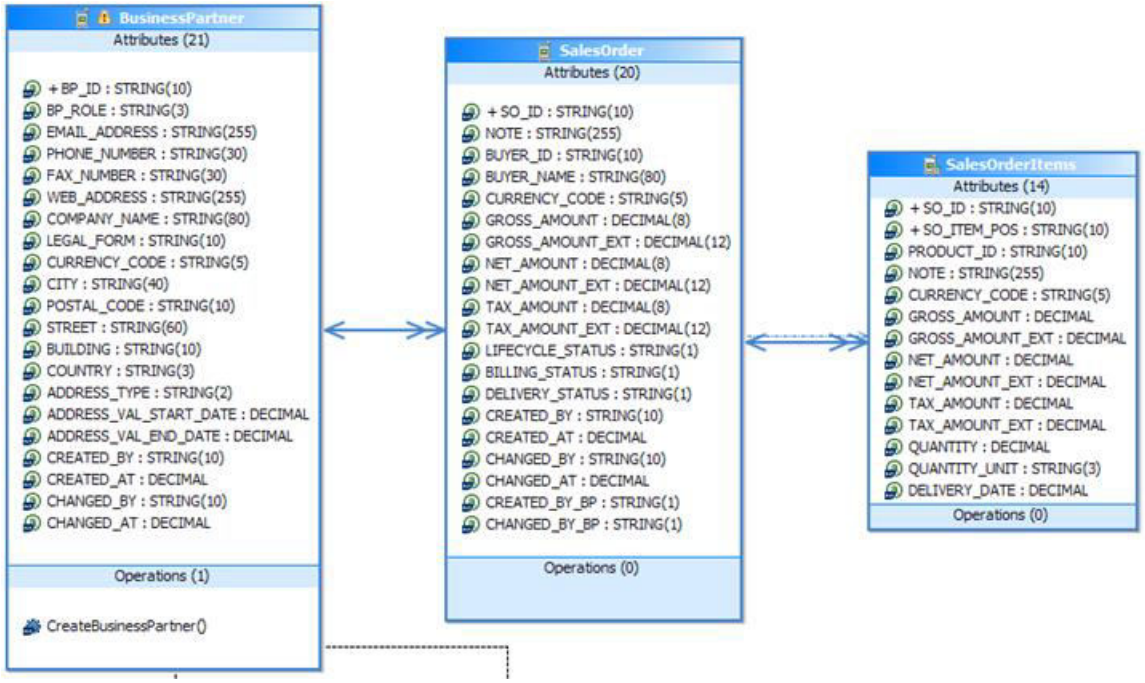
- Constructs are used to define the database structure
- Constructs typically include elements and their data types as well as groups of elements (e.g. entity), and relationships among such groups
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times



Database Management Systems

Data Model – Operations

- Most data models also include a set of basic operations for specifying retrievals and updates on the database (select, insert, update, delete) .
- It is becoming more common to include concepts in the data model to specify the dynamic aspect or behavior of a database application.
- Example: A set of valid user-defined operations that are allowed on the database objects



Database Management Systems

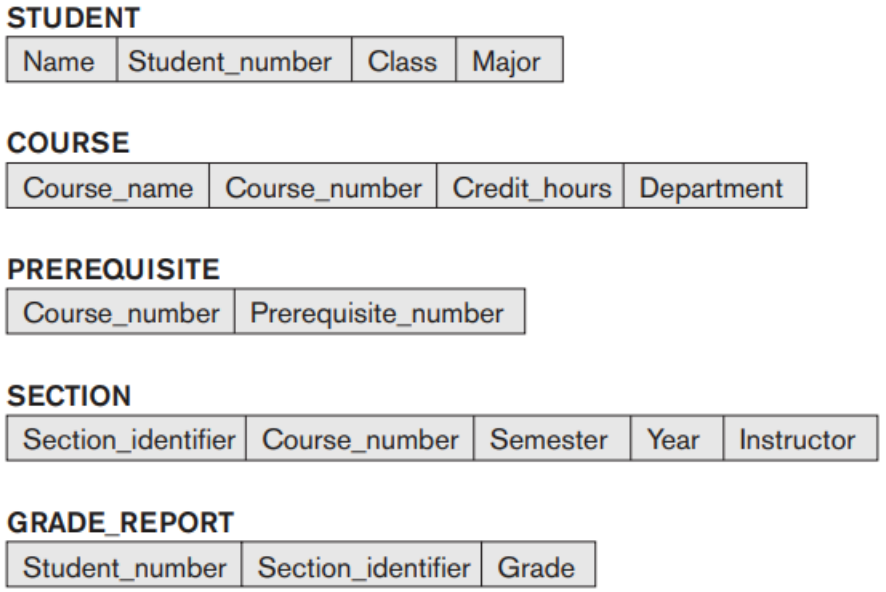
Data Model – Categories

- **High-level or conceptual data models** provide concepts that are close to the way many users perceive data
- **Low-level or physical data models** provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant for computer specialists, not for end users.
- **Representational (or implementation) data models** provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.
- **Self-describing data models** combine the description of the data with the data values themselves. Examples – XML, key-value data stores and NoSQL systems

Database Management Systems

Database schema

- The description of a database is called the **database schema**
- Database schema is specified during database design and is not expected to change frequently
- A displayed schema is called a schema diagram.
- The diagram displays the structure of each record type



Database Management Systems

Database state

- The actual data in a database may change quite frequently
- The data in the database at a particular moment in time is called a **database state** or **snapshot**
- In a given database state, each schema construct has its own **current set of instances**

  Example - COURSE construct will contain the set of individual course records as its instance.
- Every time we insert or delete a record or change the value of a data item in a record, we change one state of the database into another state
- Initial Database State - Refers to the database state when it is initially loaded into the system.
- Valid State - A state that satisfies the structure and constraints of the database.

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

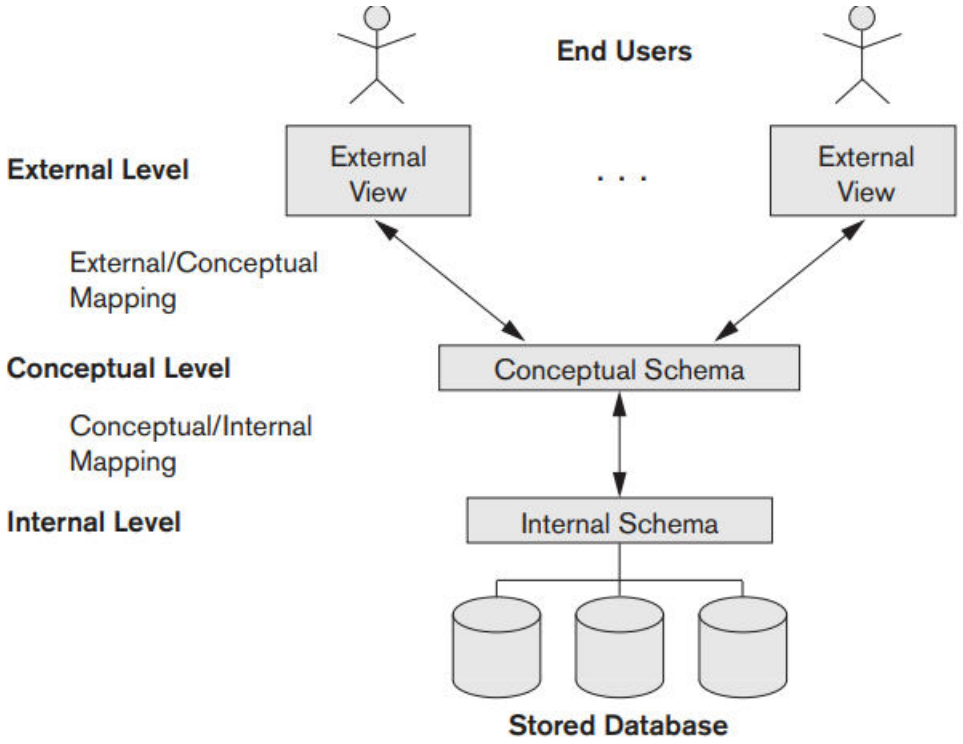
Figure 1.2
A database that stores student and course information.

Database Management Systems

Three schema architecture

- The goal of the three-schema architecture is to separate the user applications from the physical database.
- In this architecture, schemas can be defined at the following three levels:

1. **The internal level** has an internal schema, which describes the physical storage structure of the database.
2. **The conceptual level** has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures
3. **The external or view level** includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group



THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Data Indepedence

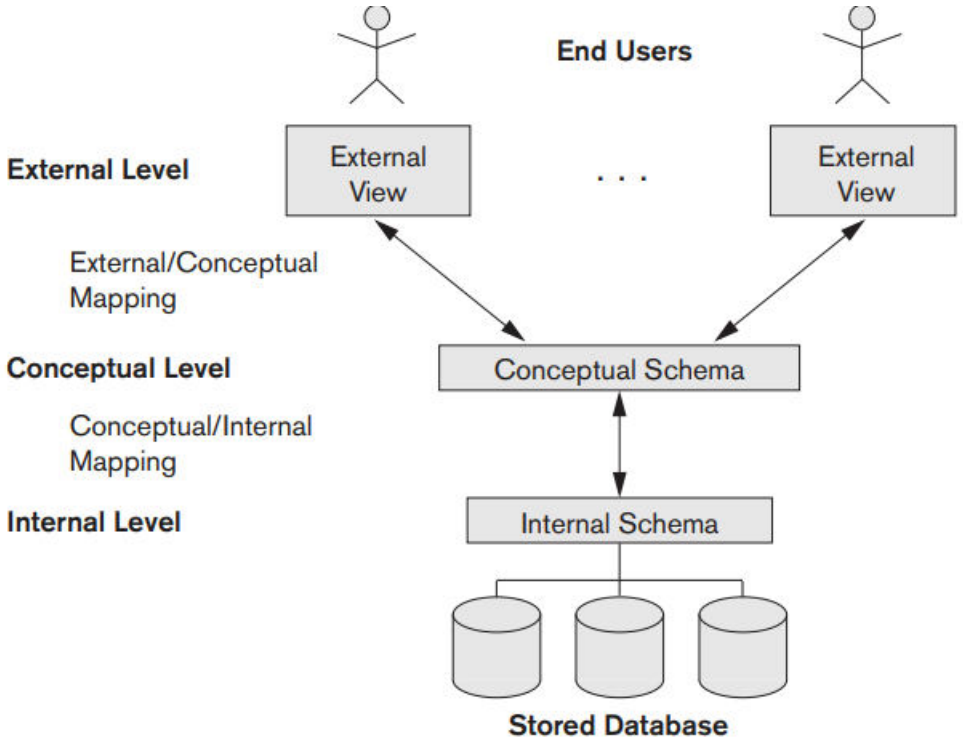
Suresh Jamadagni

Dept of Computer Science and Engineering

Database Management Systems

Data Independence

- **Data independence** is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.
- **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs.
- **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema



Database Languages

- **Data definition language (DDL),** is used by the DBA and by database designers to define both conceptual and internal schemas for the database and any mappings between the two
- The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only. **Storage definition language (SDL)**, is used to specify the internal schema.
- In most DBMSs, the DDL is used to define both conceptual and external schemas.

Database Management Systems

Data Definition Language - Example

```
/* SDL */
CREATE PARTITION FUNCTION CustRangePF1 (int)
   AS RANGE LEFT FOR VALUES (1000,2000) ;
GO

/* SDL */
CREATE PARTITION SCHEME CustRangePS1
   AS PARTITION CustRangePF1
   TO (FG1, FG2, FG3) ;
GO
```

```
/* DDL */
CREATE TABLE Customer (CustId int not null primary key,
                       CustName varchar(50) not null,
                       Street varchar(50) not null,
                       City varchar(20) not null,
                       State varchar(20) not null,
                       PostalCode char(6) not null,
                       Phone char(10) not null)
   ON CustRangePS1 (CustId)
GO
```

Database Management Systems

Database Languages - Data manipulation language (DML)

- **Data manipulation language (DML) -** set of operations or a language provided by DBMS to insert, retrieve, delete and modify data.
- There are two main types of DMLs. A **high-level or nonprocedural DML** which can be used on its own to specify complex database operations concisely.
- High-level DML statements can be entered interactively from a display monitor or terminal or embedded in a general-purpose programming language.
- High-level DMLs are called **set-at-a-time** or set-oriented DMLs as they can specify and retrieve many records in a single DML statement
- **A low-level or procedural DML** must be embedded in a general-purpose programming language. This type of DML typically retrieves individual records or objects from the database and processes each separately.
- Low-level DMLs are also called **record-at-a-time** DMLs because typically they retrieve individual records or objects from the database and processes each separately.

Database Management Systems

Data Manipulation Language - Example

```
/* High level DML */
select * from Customer where CustName = '829869ABC'
go
insert into Customer (CustId, CustName, Address, City,
State, PostalCode)
Values (1000000, ‘ABC’, ‘Ring Road', 'Bangalore',
'Karnataka', '560085‘)
go
Update Customer set PostalCode = ‘560001’  where
CustId = 1000000
go
delete from Customer where CustId = 1000000
go
```

```
/* Low Level DML */
declare @i int
set @i = 1
while @i <= 1000000
begin
  insert into Customer (CustId, CustName, Address, City, State,
PostalCode)
  select @i, convert(varchar, rand()*1000000) + 'ABC', ‘Ring Road',
'Bangalore', 'Karnataka', '560085'
  set @i = @i + 1
end
go
```

Database Management Systems

Database interfaces

Menu-based Interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request

Forms-based Interfaces display a form to each user to fill out the required data to perform an operation.

Graphical User Interfaces display a schema to the user in a diagrammatic form. The user then can specify an operation by manipulating the diagram.

Natural Language Interfaces accept requests written in English or some other language and interprets the request. The interface generates a high-level query corresponding to the natural language request and submits it to the DBMS for processing

Keyword-based Database Search similar to web search engines

Speech Input and Output - Limited use of speech as an input query and speech as an answer to a question or result of a request

Database Management Systems

Database interfaces

Interfaces for Parametric Users – Limited set of operations performed repeatedly.

Example: Bank tellers

Interfaces for the DBA – privileged commands executed by database administrators

Database Management Systems

Database programming language interfaces

- **Embedded Approach**: embedded SQL in C, C++, etc., SQLJ (for Java)

  Example:

  ```
  EXEC SQL SELECT name emp_number INTO host_name host_emp_number FROM
  employees WHERE emp_number = 10001
  ```

- **Procedure Call Approach**: e.g. JDBC for Java, ODBC (Open Database Connectivity) for other programming languages as API's (application programming interfaces)

  Example:

  ```
  while (SQL_SUCCEEDED(ret = SQLFetch(stmt)))
      ret = SQLGetData(stmt, i, SQL_C_CHAR, buf, sizeof(buf), &indicator);
  ```

Database Management Systems

Database programming language interfaces

- **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components

  Example:

```
DECLARE
   x NUMBER := 100;
BEGIN
   FOR i IN 1..10 LOOP
      IF MOD(i,2) = 0 THEN      -- i is even
         INSERT INTO temp VALUES (i, x, 'i is even');
      ELSE
         INSERT INTO temp VALUES (i, x, 'i is odd');
      END IF;
      x := x + 100;
   END LOOP;
   COMMIT;
END;
```

Database Management Systems

Database programming language interfaces

- **Scripting Languages**: PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

```php
<?php
$link = mysqli_connect("localhost", "root", "", "Mydb");

if ($link === false) {
    die("ERROR: Could not connect. "
                .mysqli_connect_error());
}

$sql = "SELECT * FROM Data";
if ($res = mysqli_query($link, $sql)) {
    if (mysqli_num_rows($res) > 0) {
        echo "<table>";
        echo "<tr>";
        echo "<th>Firstname</th>";
        echo "<th>Lastname</th>";
        echo "<th>age</th>";
        echo "</tr>";
        while ($row = mysqli_fetch_array($res)) {
            echo "<tr>";
            echo "<td>".$row['Firstname']."</td>";
            echo "<td>".$row['Lastname']."</td>";
            echo "<td>".$row['Age']."</td>";
            echo "</tr>";
        }
        echo "</table>";
        mysqli_free_result($res);
    }
    else {
        echo "No matching records are found.";
    }
}
else {
    echo "ERROR: Could not able to execute $sql. "
                                    .mysqli_error($link);
}
mysqli_close($link);
?>
```

Database Management Systems

Database system utilities/tools

- Loading data stored in files into a database. Includes data conversion tools.
- Extracting data from a database and storing it in files
- Job scheduler - Backing up the database periodically onto tapes.
- Reorganizing database file structures.
- Performance monitoring – Database usage and provide statistics to DBA.
- Data replication
- Data compression, encryption etc.

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Database Architecture

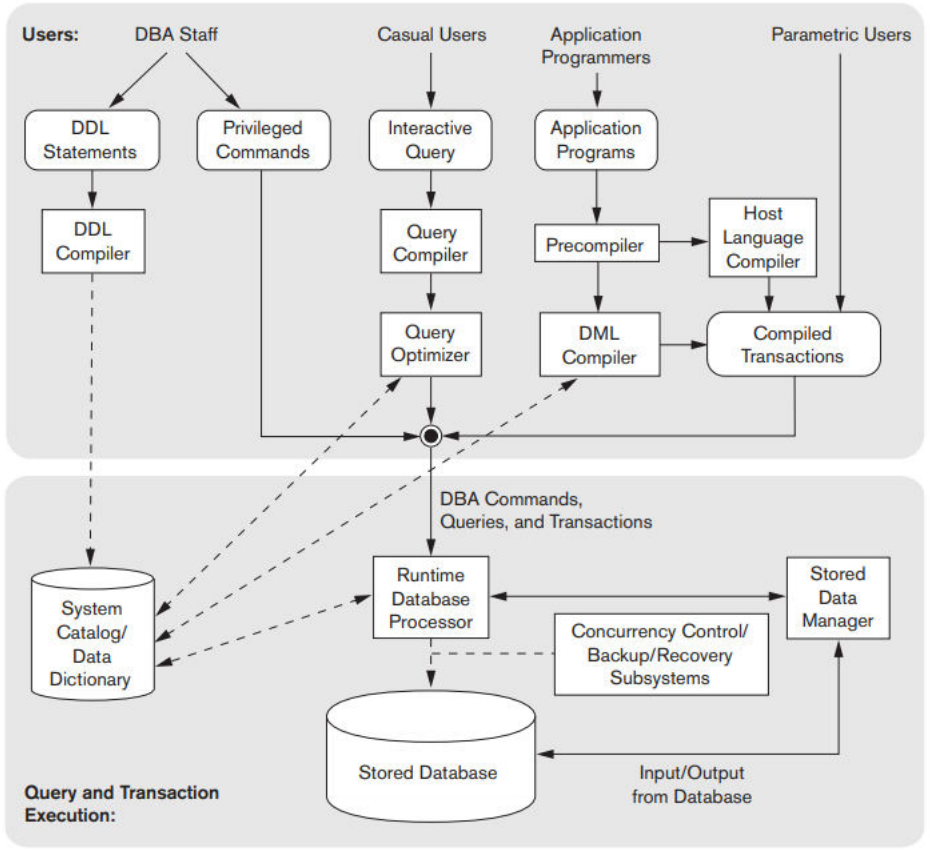
Suresh Jamadagni

Dept of Computer Science and Engineering

Database Management Systems

DBMS Component Modules

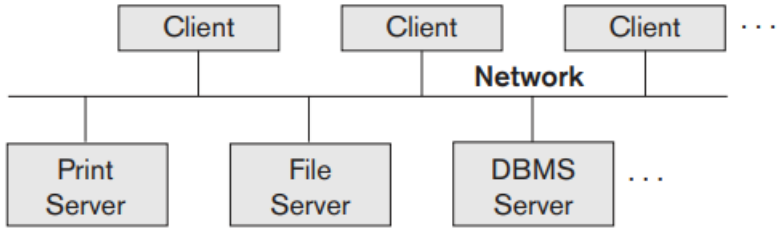
- The **DDL compiler** processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog
- The **catalog** includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints.
- The **query compiler** parses and validated for correctness of the query syntax, the names of files and data elements
- The **query optimizer** rearranges and reorders operations and uses efficient search algorithms during execution
- The **precompiler** extracts DML commands from an application program written in a host programming language (c/Java etc.) and sends them to the DML compiler for compilation into object code for database access
- The **runtime database processor** executes the privileged commands, the executable query plans and the canned transactions with runtime parameters



Database Management Systems

Basic Client/Server Architecture

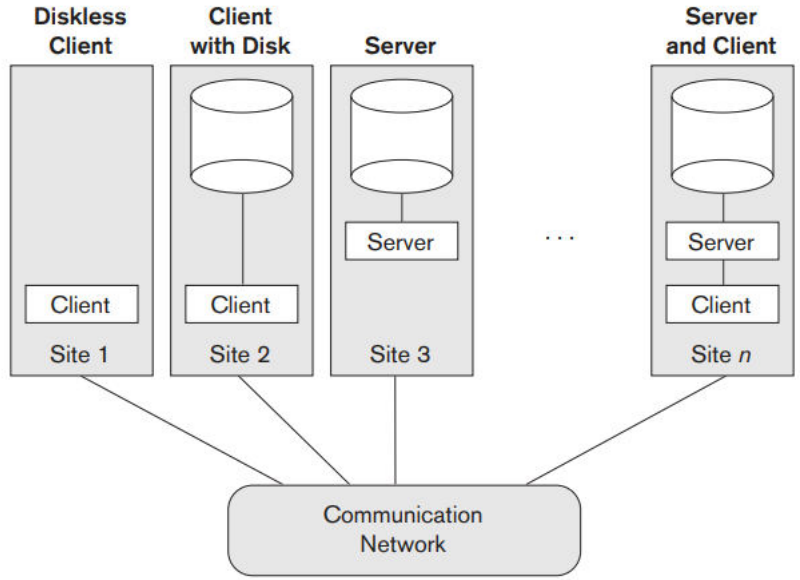
- The concept of **client/server** architecture assumes an underlying framework that consists of many PCs/workstations and a smaller number of server machines, connected via wireless networks or LANs and other types of computer networks
- A **client** in this framework is typically a user machine that provides user interface capabilities and local processing
- A **server** is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access



Database Management Systems

Two-Tier Client/Server Architectures for DBMSs

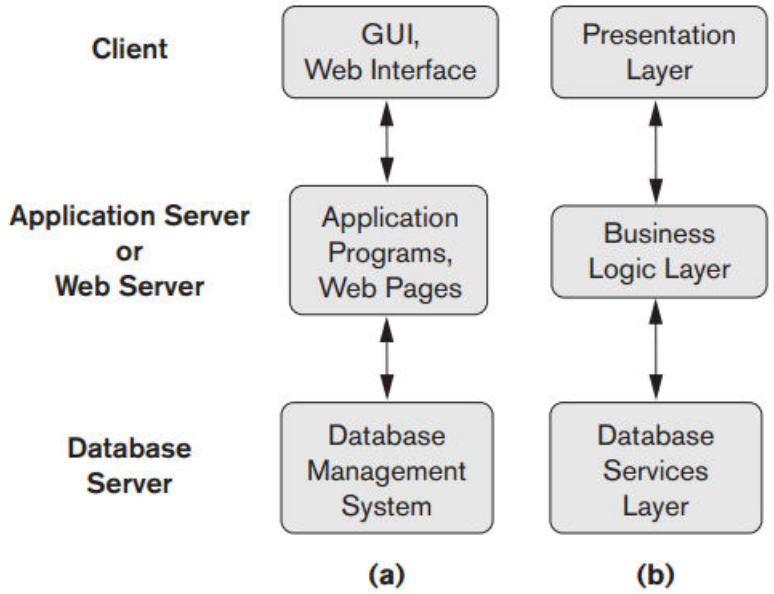
- In a **two-tier architecture**, the software components are distributed over two systems: **client and server**.
- The user interface programs and application programs can run on the **client**.
- The DBMS runs on the **server**
- The application running on the client establishes a connection to the DBMS. Once the connection is created, the client program can communicate with the DBMS
- The advantages of this architecture are its simplicity and seamless compatibility with existing systems



Database Management Systems

Three-Tier and n-Tier Architectures for DBMSs

- In a three-tier architecture, there exists an intermediate layer between the client and the database server
- This intermediate layer or middle tier is called the **application server** or the **Web server**, depending on the application
- This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server.
- Clients contain user interfaces and Web browsers.
- The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing processed data from the database server to the clients
- The user interface, application rules, and data access act as the three tiers



Classification of Database Management Systems

Based on the supported data model

- Hierarchical
- Networked
- Relational
- Object
- Object relational
- NoSQL
  - Document
  - Columnar
  - Key-value
  - Graph

Based on the number of users

- Single user
- Multi-user

Based on the number of sites at which the data is stored

- **Centralized** - Data is stored at a single computer site
- **Distributed** – Data and DBMS software distributed over many sites connected by a computer network.

  Homogeneous DDBMSs use the same DBMS software at all the sites, whereas **heterogeneous** DDBMSs can use different DBMS software at each site

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Database Design – Conceptual Model

Suresh Jamadagni

Dept of Computer Science and Engineering

Overview of Database Design Process

Two main activities:

1. Database design
2. Application design

- Focus is on conceptual database design
  - To design the conceptual schema for a database application
- Application design focuses on the programs and interfaces that access the database
  - Generally considered part of software engineering

Database Management Systems

Overview of Database Design Process

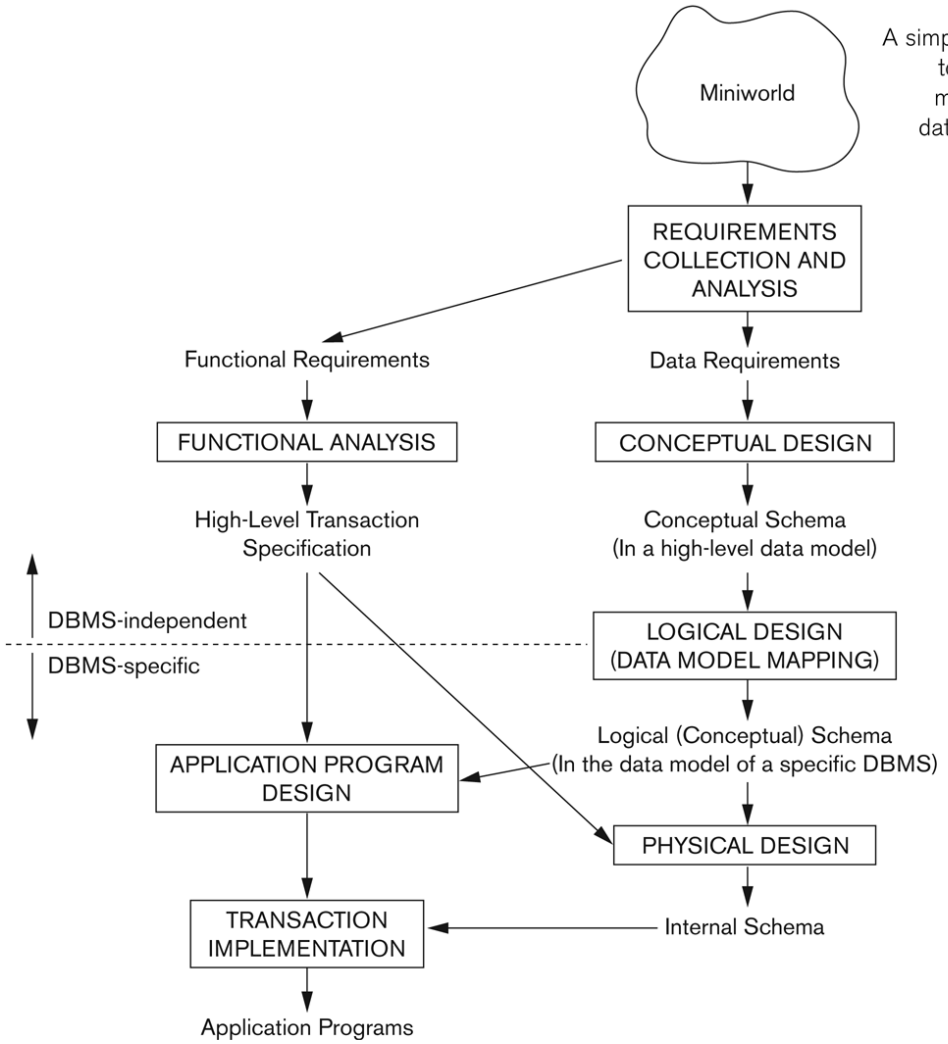


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Database Management Systems

ER Model Concepts

- **ER diagram or ER Model** – Graphical representation of the schema of a database application
- **Entities**
  - Entity is the basic concept represented in an ER model.
  - Entities are specific **things or objects in the mini-world** that are represented in the database.
    - Example: **EMPLOYEE** John Smith, the Research **DEPARTMENT**, the ProductX **PROJECT**
- **Attributes**
  - Attributes are properties used to describe an entity.
    - Example: EMPLOYEE entity may have the attributes **Name, SSN, Address, Gender, BirthDate**

Database Management Systems

ER Model Concepts (Continued)

- **Attributes**
  - A specific entity will have a value for each of its attributes.
    - Example: A specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Gender ='M', BirthDate ='09-JAN-55‘
  - Each attribute has a data type associated with it – e.g. integer, string, date, enumerated type, …
  - The attribute values that describe each entity become a major part of the data stored in the database

Database Management Systems

Types of Attributes

- **Simple**
  - Each entity has a **single atomic value** for the attribute.
  - Example: SSN
- **Composite**
  - The attribute may be composed of several components.

    For example:
    - Address (Apt#, House#, Street, City, State, ZipCode, Country)
    - Name (FirstName, MiddleName, LastName)
    - Composition may form a hierarchy where some components are themselves composite.

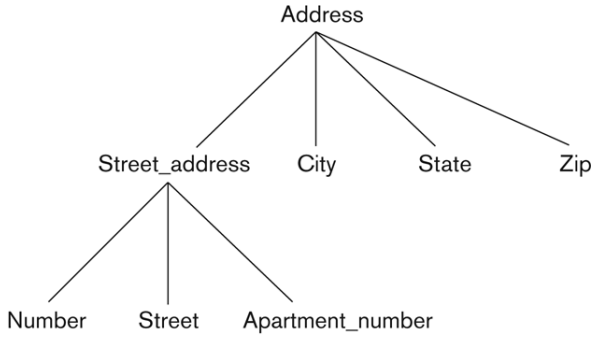


Figure 3.4
A hierarchy of composite attributes.

Types of Attributes

- **Multi-valued**
  - An entity may have multiple values for that attribute.
  - Example: PreviousDegrees of a STUDENT denoted as {PreviousDegrees}.
- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.

  Example:
  - PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
  - Multiple PreviousDegrees values can exist. Each has four subcomponent attributes: College, Year, Degree, Field

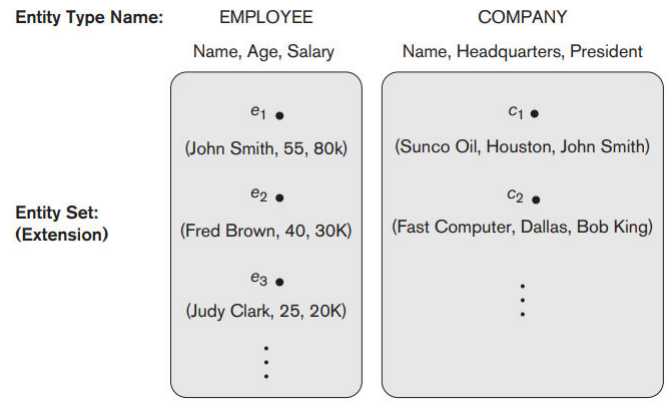
Types of Attributes

- **Stored versus Derived Attributes**
  - Two (or more) attribute values could be related. Example, the Age and Birth_date attributes of a person
  - The **Age** attribute is called a **derived attribute** since the value of Age can be determined from the current (today's) date and the value of that person's Birth_date.
  - The **Birth_date** attribute is called a **stored attribute**

Database Management Systems

Entity Types

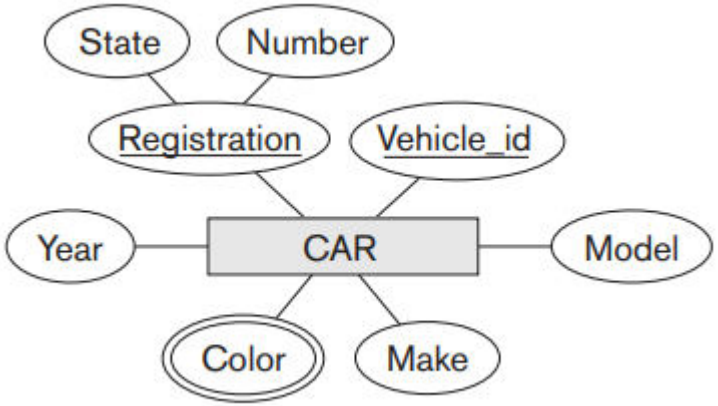
- A database usually contains groups of entities that are similar.
- Example, a company employing hundreds of employees may want to store similar information concerning each of the employees. These employee entities share the same attributes, but each entity has its own value(s) for each attribute.
- An **entity type** defines a collection (or set) of entities that have the same attributes
- Example: EMPLOYEE and COMPANY and some member entities of each



Database Management Systems

Key Attribute

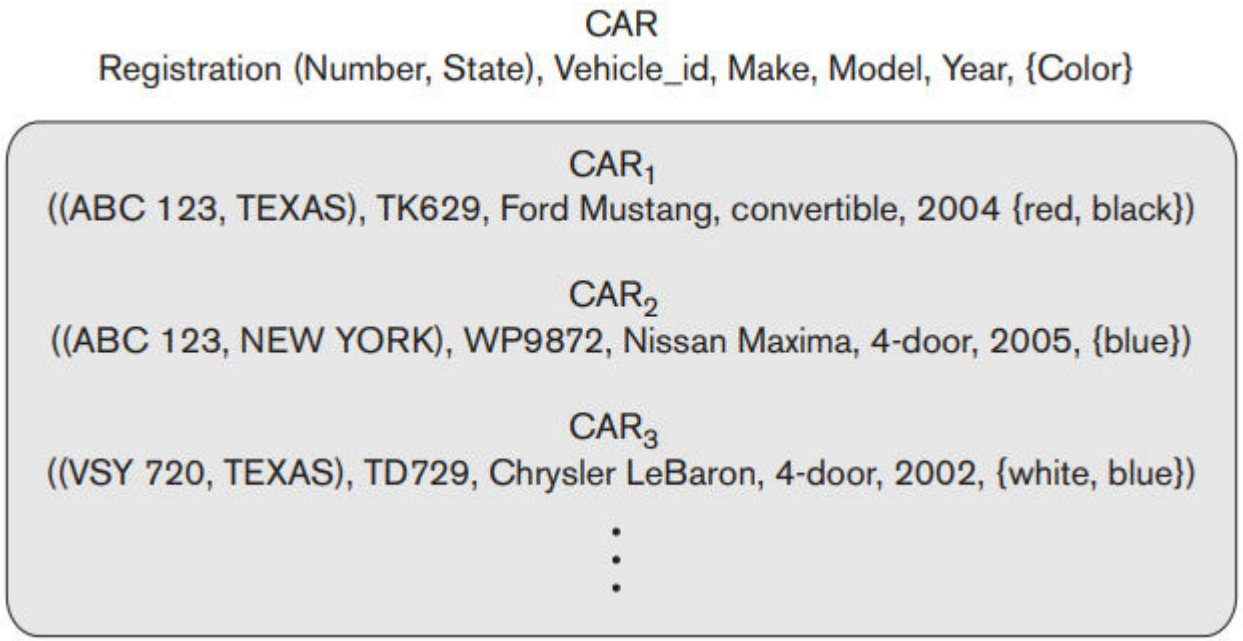
- An **attribute** of an entity type for which each entity must have a **unique value** is called a **key attribute** of the entity type.
  - Example: SSN of EMPLOYEE.
- A **key attribute** may be **composite**.
  - VehicleChassisNumber, VehicleEngineNumber is a key of the CAR entity type
- An **entity** type may have **more than one key**.
  - The CAR entity type may have three keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleRegistrationNumber aka license plate number.
    - VehicleChassisNumber, VehicleEngineNumber



Database Management Systems

Entity Set

- Each entity type will have a collection of entities stored in the database called the **entity set** or **entity collection**
- Entity set is the ***current state*** of the entities of that type that are stored in the database
- Entity set for CAR



Database Management Systems

Value Sets (domains) of Attributes

- Each simple attribute is associated with a value set
  - Example: Lastname has a value which is a character string of up to 15 characters
  - Date has a value consisting of DD-MM-YYYY where each letter is an integer
- A **value set** specifies the set of values associated with an attribute
- Value sets are similar to data types in most programming languages – e.g., integer, character (n), real, bit
- Mathematically, an attribute A for an entity type E whose value set is V is defined as a function

  A : E -> P(V) where P(V) indicates a power set (which means all possible subsets) of V.
- The above definition covers simple and multivalued attributes.
- We refer to the value of attribute A for entity e as A(e).

Database Management Systems

ER Diagram Notation

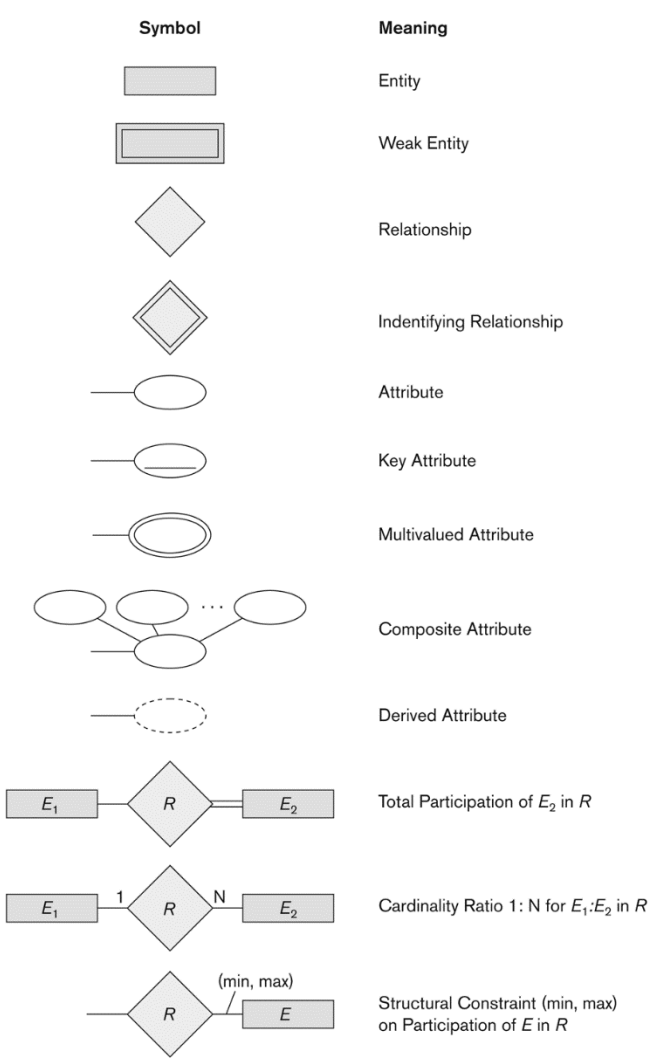


Figure 3.14
Summary of the notation for ER diagrams.

Database Management Systems

ER Diagram example - COMPANY

We need to create a database schema design based on the following **requirements** of the COMPANY Database:

- The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We need to keep track of the start date of the department manager. A department may have several locations.
- Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.
- The database will store each EMPLOYEE's social security number, address, salary, gender and birthdate.
- Each employee works for one department but may work on several projects.

Database Management Systems

ER Diagram example - COMPANY

- The DB will keep track of the number of hours per week that an employee currently works on each project.
- It is required to keep track of the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs.
- For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee.

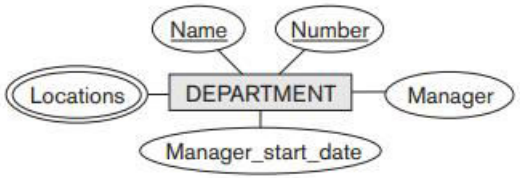
Database Management Systems

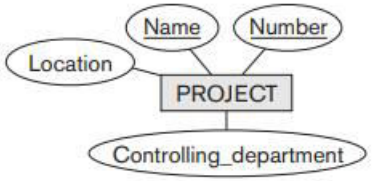
ER Diagram example - COMPANY

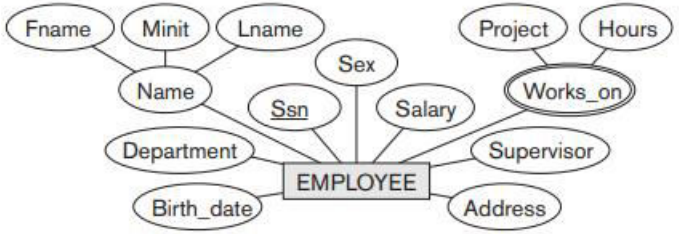
Based on the requirements, we can identify four initial entity types in the COMPANY database:

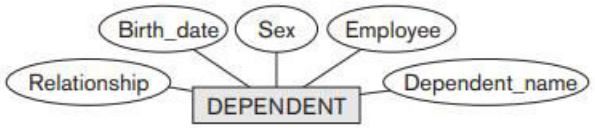
1. DEPARTMENT
2. PROJECT
3. EMPLOYEE
4. DEPENDENT

The attributes shown are derived from the requirements description









THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Database Design – Relationships

Suresh Jamadagni

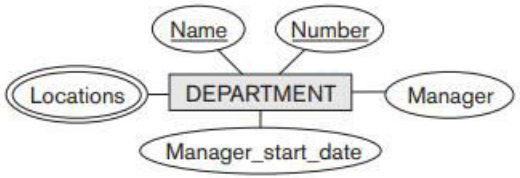
Dept of Computer Science and Engineering

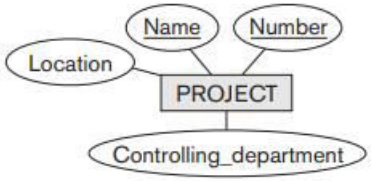
Database Management Systems

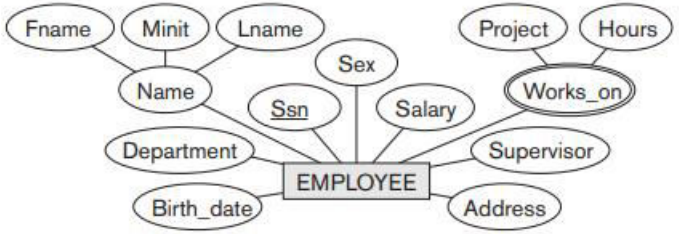
ER Diagram example - COMPANY

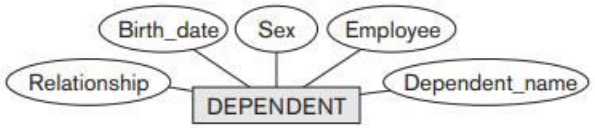
Based on the requirements, we can identify four initial entity types in the COMPANY database:

1. DEPARTMENT
2. PROJECT
3. EMPLOYEE
4. DEPENDENT









Database Design - Relationships

- Whenever an attribute of one entity type refers to another entity type, some relationship exists between the entities.
- Examples:
  1. The attribute Manager of DEPARTMENT refers to an employee who manages the department
  2. The attribute Supervisor of EMPLOYEE refers to another employee
  3. the attribute Department of EMPLOYEE refers to the department for which the employee works
- A **relationship** relates two or more distinct entities with a specific meaning
- Example: EMPLOYEE Franklin Wong manages the Research DEPARTMENT

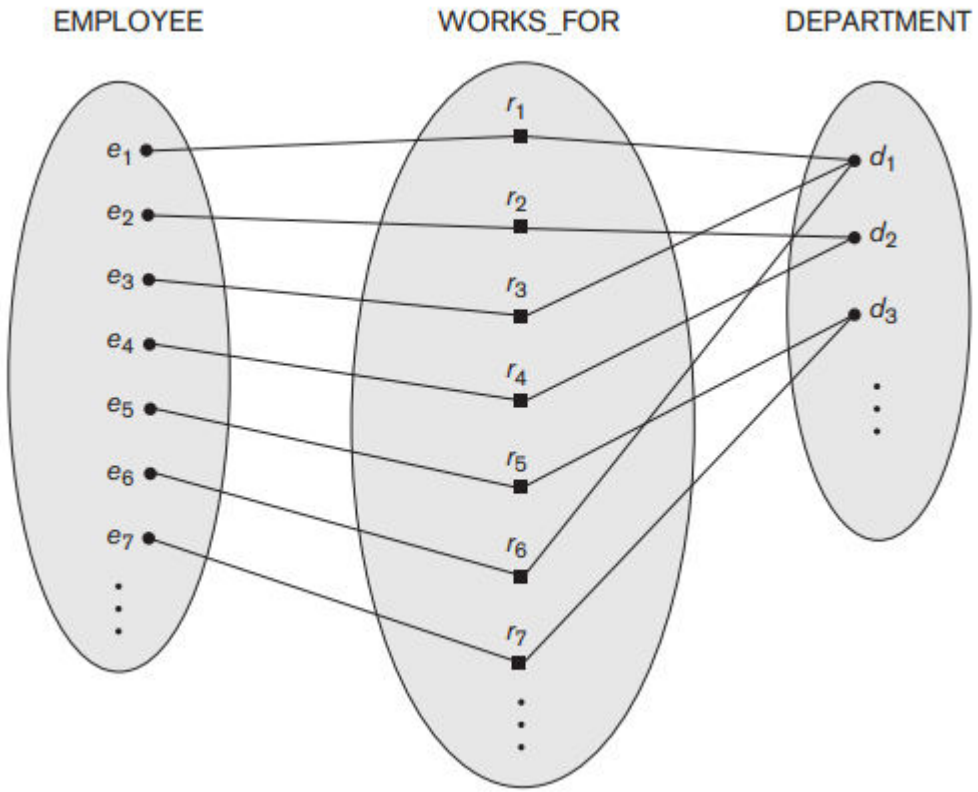
Database Design – Relationship Types

- Relationships of the same type are grouped into a **relationship type**
- Example**:**
  - The WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate
  - The MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- A relationship type R among n entity types E_1, E_2, . . . , E_n defines a set of associations or a **relationship set** among entities from these entity type
- Mathematically, the relationship set R is a set of relationship instances r_i, where each r_i associates n individual entities (e_1, e_2, . . . , e_n), and each entity e_j in r_i is a member of entity set E_j , $1 \leq j \leq n$

Database Management Systems

Database Design – Relationship Types

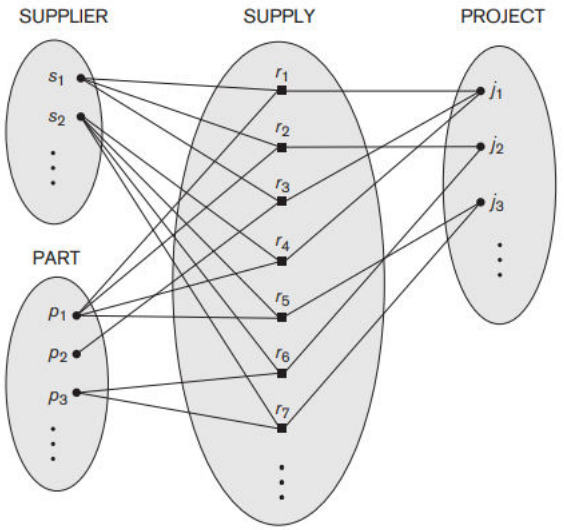
- WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT



Database Management Systems

Database Design – Relationship Degree

- The **degree** of a relationship type is the number of participating entity types
- Example: The WORKS_FOR relationship is of degree two.
- A relationship type of degree two is called **binary**
- A relationship type of degree three is called **ternary**
- Example of ternary relationship – SUPPLY which associates three entities supplier s, part p and project j



Database Management Systems

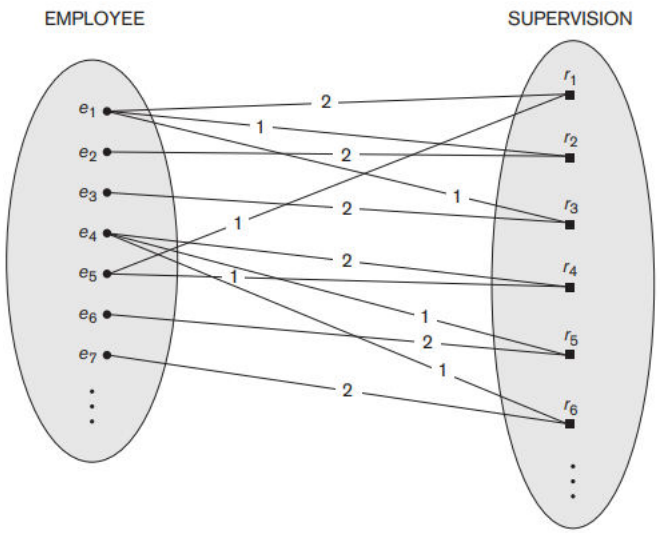
Database Design – Relationship Role name

- The **role name** signifies the role that a participating entity from the entity type plays in each relationship instance
- It helps to explain what the relationship means
- Example:
  - In the WORKS_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer

Database Management Systems

Database Design – Recursive Relationship

- In some cases the same entity type participates more than once in a relationship type in different roles.
- Such relationship types are called **recursive relationships** or **self-referencing relationships**
- It is a relationship type between the same participating entity type in **distinct roles**
- Example**:** A recursive relationship **SUPERVISION** between EMPLOYEE in the supervisor role (1) and EMPLOYEE in the subordinate role (2).



Database Design – Constraints on Relationship Types

- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set
- Example: Each employee must work for exactly one department

- Two main types of binary relationship constraints:
  1. **Cardinality ratio**
  2. **Participation Constraints and Existence Dependencies.**

- The cardinality ratio and participation constraints together, are known as the **structural constraints** of a relationship type

Database Design – Constraints on Relationship Types

Cardinality ratio

- The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.
- The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1 and M:N

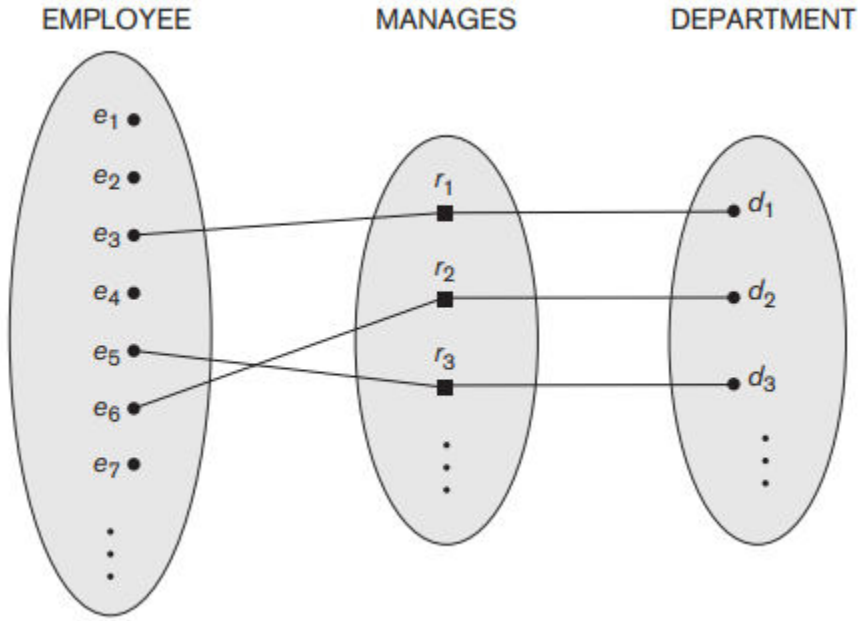
  N stands for any number of related entities (zero or more)

Database Management Systems

Database Design – Constraints on Relationship Types

Example of a relationship with 1:1 cardinality ratio

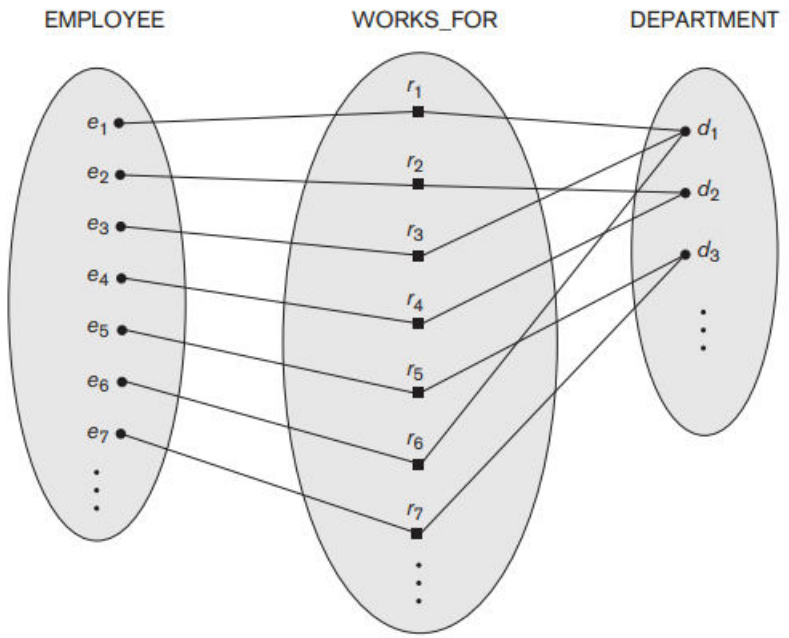
- Relationship MANAGES relates a department entity to the employee who manages that department.
- An employee can manage at most one department and a department can have at most one manager



Database Design – Constraints on Relationship Types

Example of a relationship with N:1 cardinality ratio

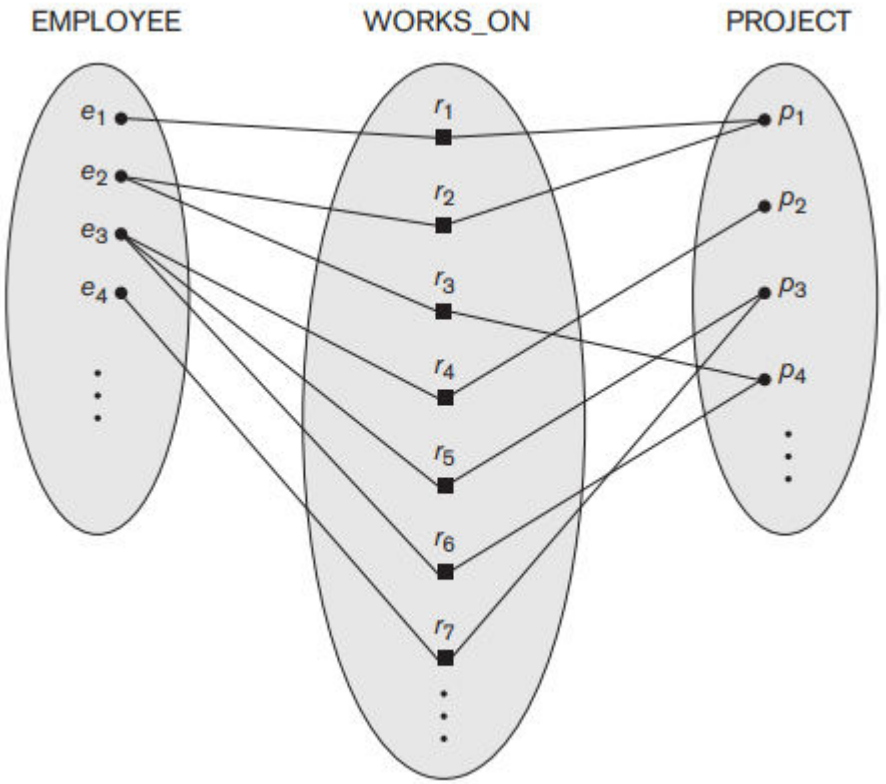
- Relationship WORKS_FOR relates employee entity to the department entity.
- Many employees work for at most one department and a department can have many employees



Database Design – Constraints on Relationship Types

Example of a relationship with M:N cardinality ratio

- The relationship type WORKS_ON is of cardinality ratio M:N
- An employee can work on several projects and a project can have several employees



Database Design – Constraints on Relationship Types

Participation Constraints and Existence Dependencies.

- This constraint specifies the minimum number of relationship instances that each entity can participate in and is sometimes called the minimum cardinality constraint.
- There are two types of participation constraints - **total and partial**
- Since every employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance.
- Thus, the participation of EMPLOYEE in WORKS_FOR is called **total** participation
- Total participation is also called existence dependency
- Since every employee doesn't manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is **partial**

Database Management Systems

Database Design – Attributes of Relationship Types

- Relationship types can also have attributes, similar to those of entity types
- We can include an attribute Hours for the WORKS_ON relationship type to record the number of hours per week that a particular employee works on a particular project
- We can also include an attribute Start_date for the MANAGES relationship type to capture the date on which a manager started managing a department
- Attributes of 1:1 or 1:N relationship types can be migrated to one of the participating entity types.
- For a 1:1 relationship type, a relationship attribute can be migrated to either of the entity types of the relationship
- For a 1:N relationship type, a relationship attribute can be migrated only to the entity type on the N-side of the relationship

Database Design – Representing Relationship Types in ER Diagram

In ER diagrams, the *relationship type* is represented as follows:

- Diamond-shaped box is used to display a relationship type
- Connected to the participating entity types via straight lines
- Note that the relationship type is not shown with an arrow. The name should be typically be readable from left to right and top to bottom.
- Total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line

Database Management Systems

Database Design – Relationship Types in COMPANY database

- WORKS_FOR (between EMPLOYEE, DEPARTMENT)
- MANAGES (also between EMPLOYEE, DEPARTMENT)
- CONTROLS (between DEPARTMENT, PROJECT)
- WORKS_ON (between EMPLOYEE, PROJECT)
- SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
- DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

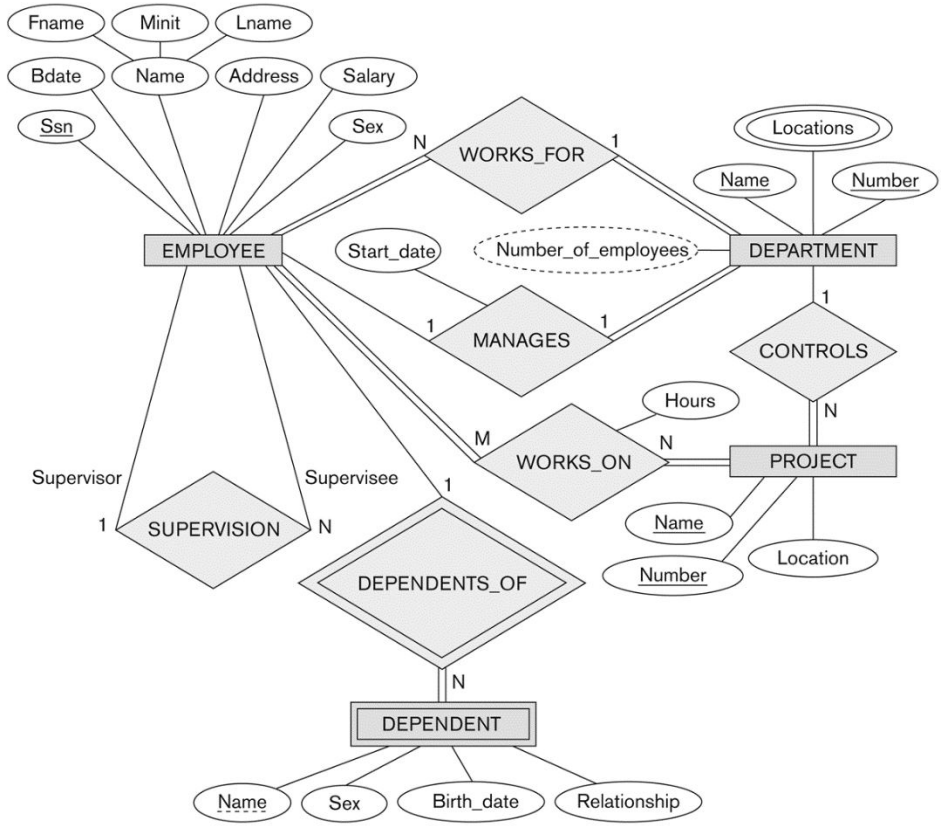


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Database Design – Weak Entity

Suresh Jamadagni

Dept of Computer Science and Engineering

Database Management Systems

Database Design – Weak Entity Types

- Entity types that do not have key attributes of their own are called **weak entity** types.
- Entity types that do have a key attribute are called **strong entity** types
- Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values
- We call this other entity type the **identifying** or **owner entity** type and we call the relationship type that relates a weak entity type to its owner the **identifying relationship** of the weak entity type
- A weak entity type always has a total participation constraint with respect to its identifying relationship because a weak entity cannot be identified without an owner entity
- **Example**: Entity type DEPENDENT, related to EMPLOYEE, which is used to keep track of the dependents of each employee via a 1:N relationship

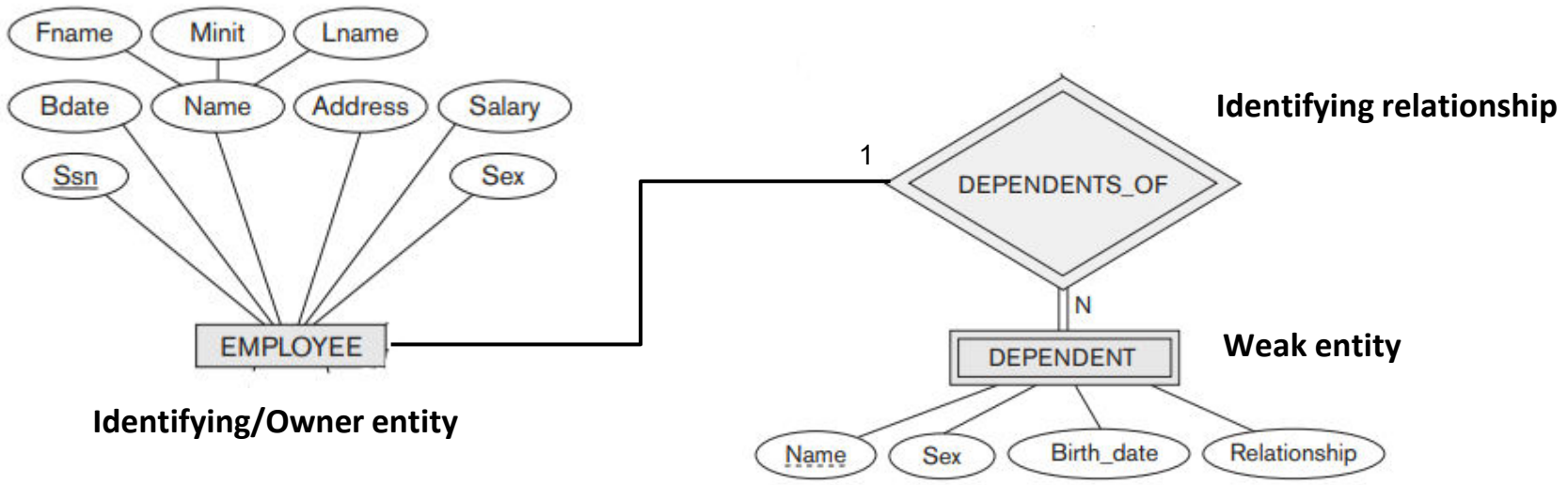
Database Design – Weak Entity Types

- A weak entity type normally has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity
- Weak entity types can sometimes be represented as complex (composite, multivalued) attributes
- **Example**: we could specify a multivalued attribute Dependents for EMPLOYEE, which is a multivalued composite attribute with the component attributes Name, Birth_date, Sex, and Relationship
- In general, any number of levels of weak entity types can be defined
- An owner entity type may itself be a weak entity type.
- A weak entity type may have more than one identifying entity type and an identifying relationship type of degree higher than two

Database Management Systems

Database Design – Weak Entity Types

- In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines
- In ER diagrams, the partial key attribute is underlined with a dashed or dotted line



THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Database Design – Refining ER Diagram

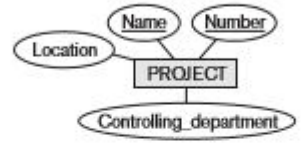
Suresh Jamadagni

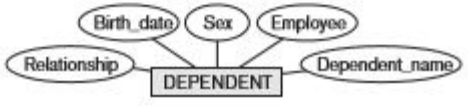
Dept of Computer Science and Engineering

Database Management Systems

Database Design – Refining ER Diagram

- Refine the database design by changing the attributes that represent relationships into relationship types
- The cardinality ratio and participation constraint of each relationship type are determined from the requirements
- If some cardinality ratio or dependency cannot be determined from the requirements, users need to determine these structural constraints

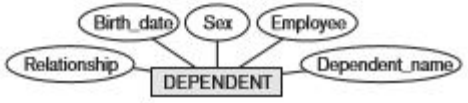




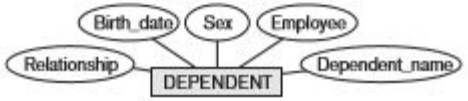
1. **MANAGES** is a 1:1 relationship type between EMPLOYEE and DEPARTMENT.
   - EMPLOYEE participation is partial.
   - DEPARTMENT participation is not clear from the requirements. Users say that a department must have a manager at all times, which implies total participation.
   - The attribute Start_date is assigned to this relationship type

Database Design – Refining ER Diagram

2. **WORKS_FOR** is a 1:N relationship type between DEPARTMENT and EMPLOYEE.
   - Both participations are total
3. **CONTROLS** is a 1:N relationship type between DEPARTMENT and PROJECT.
   - The participation of PROJECT is total
   - The participation of DEPARTMENT is partial since some departments may not control any project
4. **SUPERVISION** is a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role).
   - Both participations are determined to be partial, as users indicate that not every employee is a supervisor and not every employee has a supervisor.



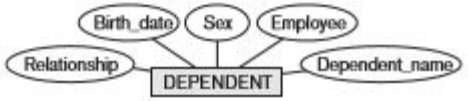
5. **WORKS_ON** is a M:N (many-to-many) relationship type between EMPLOYEE and PROJECT with attribute Hours
   - Users indicate that a project can have several employees working on it.
   - Both participations are determined to be total.
6. **DEPENDENTS_OF** is a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT.
   - The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total



Database Management Systems

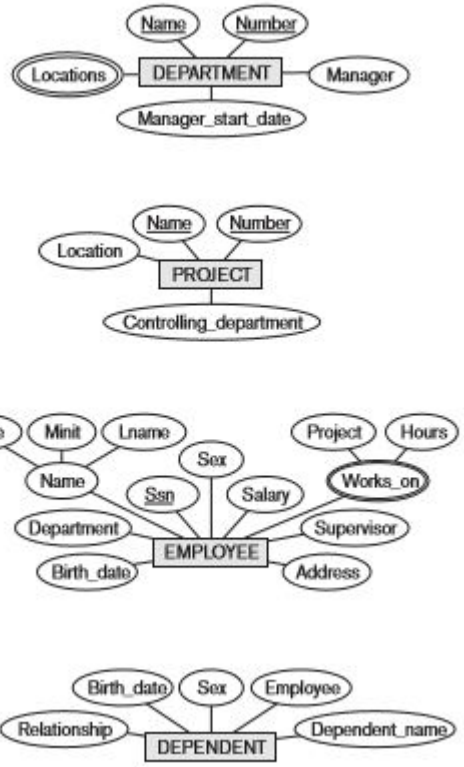
Database Design – Refining ER Diagram

- Remove all attributes from the entity types, that have been refined into relationships.
  - Manager and Manager_start_date from DEPARTMENT;
  - Controlling_department from PROJECT;
  - Department, Supervisor, and Works_on from EMPLOYEE;
  - Employee from DEPENDENT.
- It is important to have the least possible redundancy when we design the conceptual schema of a database



Database Management Systems

Database Design – Refining ER Diagram



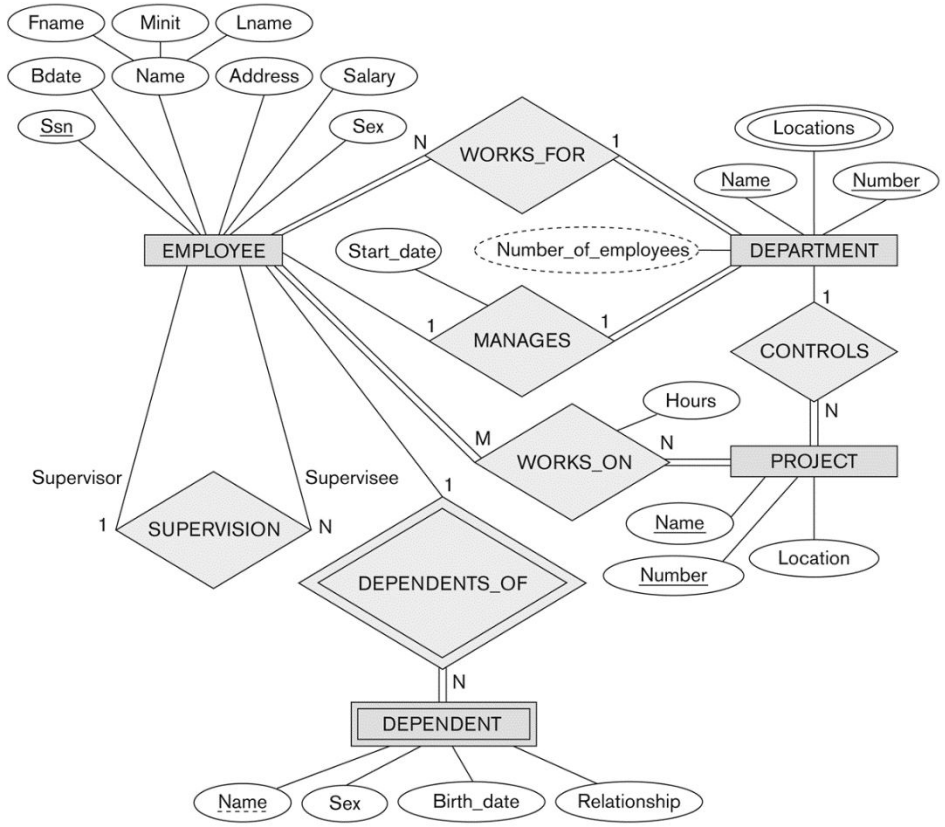


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Database Management Systems

Database Design – Best Practices for creating ER Diagrams

1. **Proper Naming of Schema Constructs**
   - Choose names that **convey**, as much as possible, the **meanings** attached to the different constructs in the schema
   - Choose to use **singular** names for **entity** types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type
   - Use the convention that **entity** type and **relationship** type names are in **uppercase** letters, **attribute** names have their **initial letter capitalized**, and **role names** are in **lowercase** letters.
   - As a general practice, given a narrative description of the database requirements, the **nouns** appearing in the narrative tend to give rise to **entity** type names, and the **verbs** tend to indicate names of **relationship** types.
   - **Attribute** names generally arise from **additional nouns** that describe the nouns corresponding to entity types
   - Choose binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom

Database Design – Best Practices for creating ER Diagrams

2. Design Choices for ER Conceptual Design

- The schema design process should be considered an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached
- Occasionally it may be difficult to decide whether a particular concept should be modeled as an entity type, an attribute, or a relationship type.
- A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type.
- Once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy

2. Design Choices for ER Conceptual Design

- An attribute that exists in several entity types may be elevated or promoted to an independent entity type.
- **Example**: Suppose that each of several entity types in a UNIVERSITY database, such as STUDENT, INSTRUCTOR, and COURSE, has an attribute Department in the initial design; the designer may then choose to create an entity type DEPARTMENT with a single attribute Dept_name and relate it to the three entity types (STUDENT, INSTRUCTOR, and COURSE) via appropriate relationships.
- An inverse refinement to the previous case may be applied in some cases
- **Example**: If an entity type DEPARTMENT exists in the initial design with a single attribute Dept_name and is related to only one other entity type, STUDENT, DEPARTMENT may be reduced or demoted to an attribute of STUDENT.

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Database Design – Relationship Types of Degree > 2

Suresh Jamadagni

Dept of Computer Science and Engineering

Database Management Systems

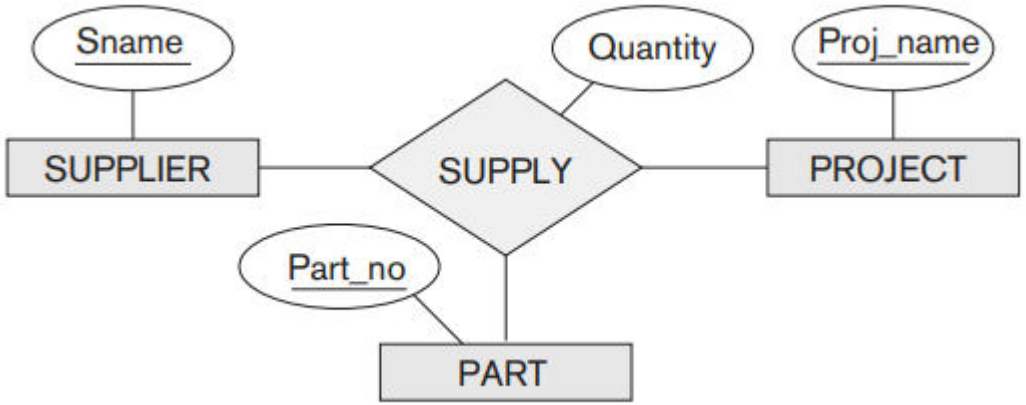
Database Design – Relationship Types of Degree Higher than Two

Recall from earlier classes:

- The degree of a relationship type as the number of participating entity types
- Relationship type of degree two - binary
- Relationship type of degree three - ternary
- Relationship type of degree n - n-ary

Database Design – Relationship Types of Degree Higher than Two

Ternary Relationship

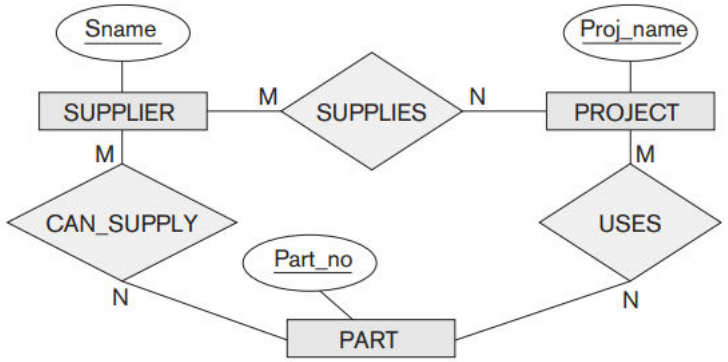


- The relationship set of SUPPLY is a set of relationship instances (s, j, p), where s is a SUPPLIER supplying a PART p to a PROJECT j.
- In general, a relationship type R of degree n will have n edges in an ER diagram, one connecting R to each participating entity type.

Database Management Systems

Database Design – Relationship Types of Degree Higher than Two

Three Binary Relationships



- Consider the three binary relationship types CAN_SUPPLY, USES, and SUPPLIES.
- Relationship type CAN_SUPPLY, between SUPPLIER and PART, includes an instance (s, p) whenever supplier s can supply part p (to any project);
- Relationship type USES, between PROJECT and PART, includes an instance (j, p) whenever project j uses part p;
- Relationship type SUPPLIES, between SUPPLIER and PROJECT, includes an instance (s, j) whenever supplier s supplies some part to project j.
- The existence of three relationship instances (s, p), (j, p), and (s, j) in CAN_SUPPLY, USES, and SUPPLIES, respectively, does not necessarily imply that an instance (s, j, p) exists in the ternary relationship SUPPLY

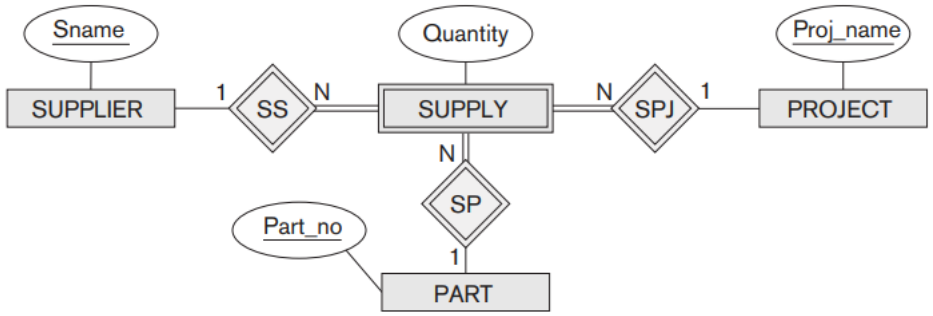
Database Design – Relationship Types of Degree Higher than Two

Choosing between Binary and Ternary (or Higher-Degree) Relationships

- It is often tricky to decide whether a particular relationship should be represented as a relationship type of degree n or should be broken down into several relationship types of smaller degrees
- The designer must base this decision on the semantics or meaning of the particular situation being represented. The typical solution is to include the ternary relationship plus one or more of the binary relationships, if they represent different meanings and if all are needed by the application

Database Design – Relationship Types of Degree Higher than Two

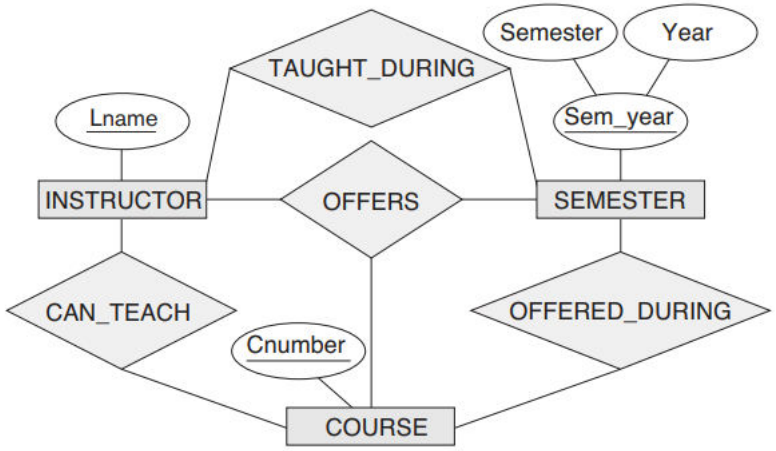
Choosing between Binary and Ternary (or Higher-Degree) Relationships



- In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types)
- If a particular binary relationship can be derived from a higher-degree relationship at all times, then it is redundant
- It is also possible to represent the ternary relationship as a regular entity type by introducing an artificial or surrogate key.
- In this example, a key attribute Supply_id could be used for the supply entity type, converting it into a regular entity type. Three binary N:1 relationships relate SUPPLY to each of the three participating entity types

Database Design – Relationship Types of Degree Higher than Two

Another Example of Ternary Relationship



- The ternary relationship type **OFFERS** represents information on instructors offering courses during particular semesters. It includes a relationship instance (i, s, c) whenever INSTRUCTOR i offers COURSE c during SEMESTER s
- The three binary relationship types shown in the above diagram have the following meanings:
  - **CAN_TEACH** relates a course to the instructors who can teach that course,
  - **TAUGHT_DURING** relates a semester to the instructors who taught some course during that semester
  - **OFFERED_DURING** relates a semester to the courses offered during that semester by any instructor

Database Design – Constraints on Higher Degree Relationship Types

- These ternary relationship type **OFFERS** and binary relationships **CAN_TEACH, TAUGHT_DURING** and **OFFERED_DURING** represent different information
- But certain constraints hold among the relationships.
- A relationship instance (i, s, c) should not exist in OFFERS unless an instance (i, s) exists in TAUGHT_DURING, an instance (s, c) exists in OFFERED_DURING, and an instance (i, c) exists in CAN_TEACH
- It is possible to have instances (i, s), (s, c), and (i, c) in the three binary relationship types with no corresponding instance (i, s, c) in OFFERS
- We can infer the instances of TAUGHT_DURING and OFFERED_DURING from the instances in OFFERS, but cannot infer the instances of CAN_TEACH
- Therefore, TAUGHT_DURING and OFFERED_DURING are redundant and can be left out

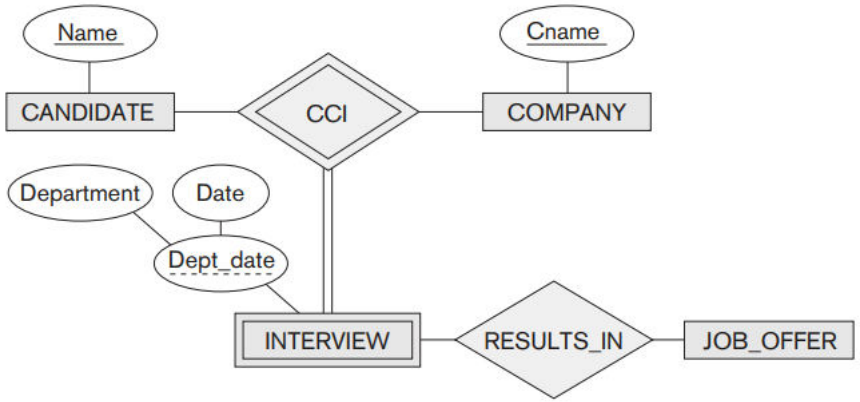
Database Management Systems

Database Design – Constraints on Higher Degree Relationship Types

- In general, an n-ary relationship is not equivalent to n binary relationships
- However, n-ary relationship can be replaced with n binary relationships under certain circumstances by introducing additional constraints
- In the example, if the CAN_TEACH relationship is 1:1 (an instructor can teach only one course, and a course can be taught by only one instructor), then the ternary relationship OFFERS can be left out because it can be inferred from the three binary relationships CAN_TEACH, TAUGHT_DURING, and OFFERED_DURING
- The schema designer must analyze the meaning of each specific situation to decide which of the binary and ternary relationship types are needed

Database Management Systems

Database Design – Constraints on Higher Degree Relationship Types

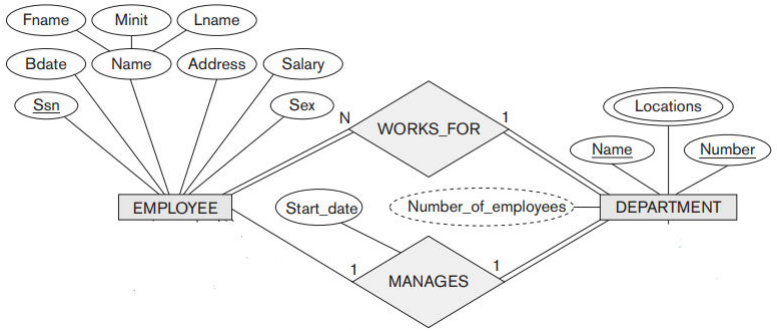


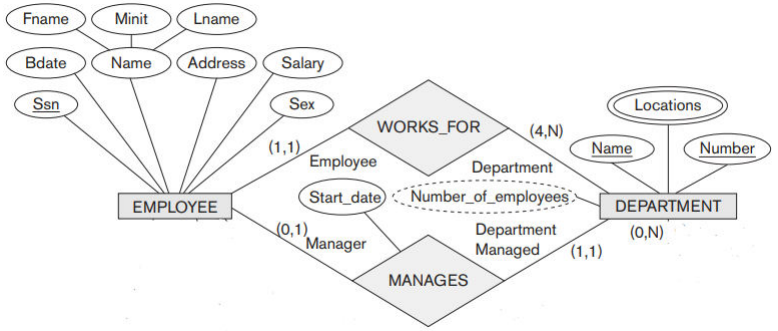
- The above example shows part of a database that keeps track of candidates interviewing for jobs at various companies
- A candidate can have multiple interviews with the same company (for example, with different departments or on separate dates), but a job offer is made based on one of the interviews.
- Relationship type INTERVIEW is represented as a weak entity with two owners CANDIDATE and COMPANY, and with the partial key Dept_date.
- An INTERVIEW entity is uniquely identified by a candidate, a company, and the combination of the date and department of the interview

Database Design – Higher Degree Relationship Types

Notations for specifying constraints on Higher Degree Relationship Types

- There are two notations for specifying structural constraints on n-ary relationships, and they specify different constraints. Thus, both should be used if it is important to fully specify the structural constraints on a ternary or higher-degree relationship.
- The first notation is based on the **cardinality ratio** notation of binary relationships. A 1, M, or N is specified on each participation arc.
- The second notation is based on the (min, max) notation for binary relationships. A **(min, max) on a participation** here specifies that each entity is related to at least min and at most max relationship instances in the relationship set





THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu

Database Management Systems

Database Design – Case Study

Suresh Jamadagni

Dept of Computer Science and Engineering

Database Management Systems

Database Design – University Database

Scope:

- Design a database to keep track of student enrollments in classes and students' final grades.

Requirements:

- The university is organized into colleges (COLLEGE), and each college has a unique name (CName), a main office (COffice) and phone (CPhone), and a particular faculty member who is dean of the college.
- Each college administers a number of academic departments (DEPT). Each department has a unique name (DName), a unique code number (DCode), a main office (DOffice) and phone (DPhone), and a particular faculty member who chairs the department. Need to keep track of the start date (CStartDate) when that faculty member began chairing the department

Database Management Systems

Database Design – University Database

Requirements:

- A department offers a number of courses (COURSE), each of which has a unique course name (CoName), a unique code number (CCode), a course level, a course credit hours (Credits), and a course description (CDesc).
- The database also needs to keep track of instructors (INSTRUCTOR). Each instructor has a unique identifier (Id), name (IName), office (IOffice), phone (IPhone), and rank (Rank). Each instructor works for one primary academic department.
- The database needs to keep track of student data (STUDENT) and store each student's name (SName, composed of first name (FName), middle name (MName), last name (LName), student id (Sid, unique for every student), address (Addr), phone (Phone), major code (Major), and date of birth (DoB).
- A student is assigned to one primary academic department. It is required to keep track of the student's grades in each section the student has completed

Database Management Systems

Database Design – University Database

Requirements:

- Courses are offered as sections (SECTION). Each section is related to a single course and a single instructor and has a unique section identifier (SecId)
- A section also has a section number (SecNo: this is coded as 1, 2, 3, . . . for multiple sections offered during the same semester/year), semester (Sem), year (Year), classroom (CRoom: this is coded as a combination of building code (Bldg) and room number (RoomNo) within the building), and days/times (DaysTime: for example, 'MWF 9am-9.50am' or 'TR 3.30pm-5.20pm'— restricted to only allowed days/time values).
- The database will keep track of all the sections offered for the past several years, in addition to the current offerings. The SecId is unique for all sections, not just the sections for a particular semester.
- The database should keep track of the students in each section, and the grade is recorded when available (this is a many-to-many relationship between students and sections). A section must have at least five students.

Database Management Systems

Database Design – University Database

Entities?

- COLLEGE
- DEPT
- COURSE
- INSTRUCTOR
- SECTION
- STUDENT

Database Design – University Database

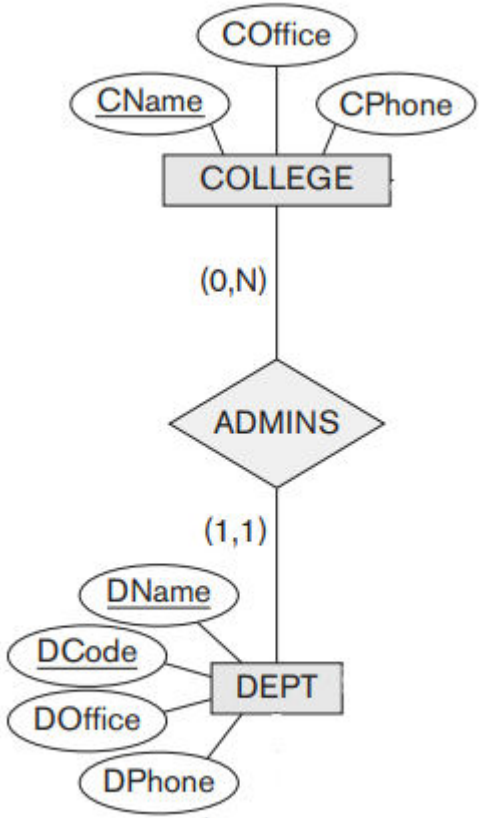
Relationships?

1. ADMINS between COLLEGE and DEPT
2. CHAIR (chairperson) between DEPT and INSTRUCTOR
3. DEAN between COLLEGE and INSTRUCTOR
4. EMPLOYS between DEPT and INSTRUCTOR
5. OFFERS between DEPT and COURSE
6. SECTIONS between COURSE and SECTION
7. HAS between DEPT and STUDENT
8. TAKES (ENROLS) between STUDENT and SECTION
9. TEACHES between INSTRUCTOR and SECTION

Database Design – University Database

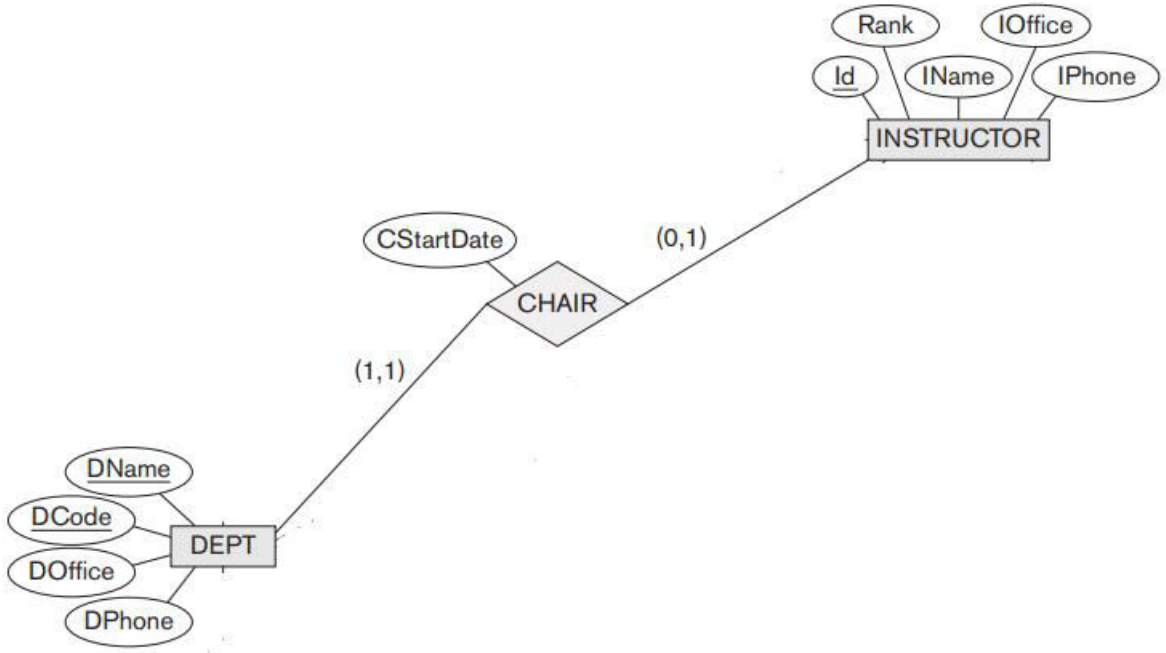
Constraint on Relationships?

1. ADMINS between COLLEGE and DEPT



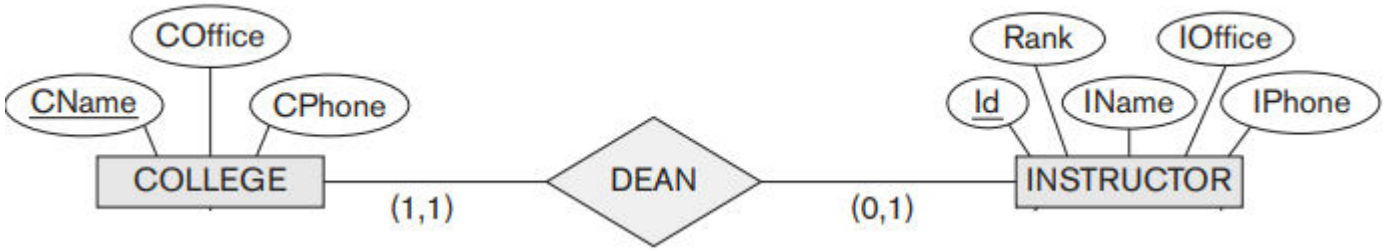
Constraint on Relationships?

2. CHAIR (chairperson) between DEPT and INSTRUCTOR



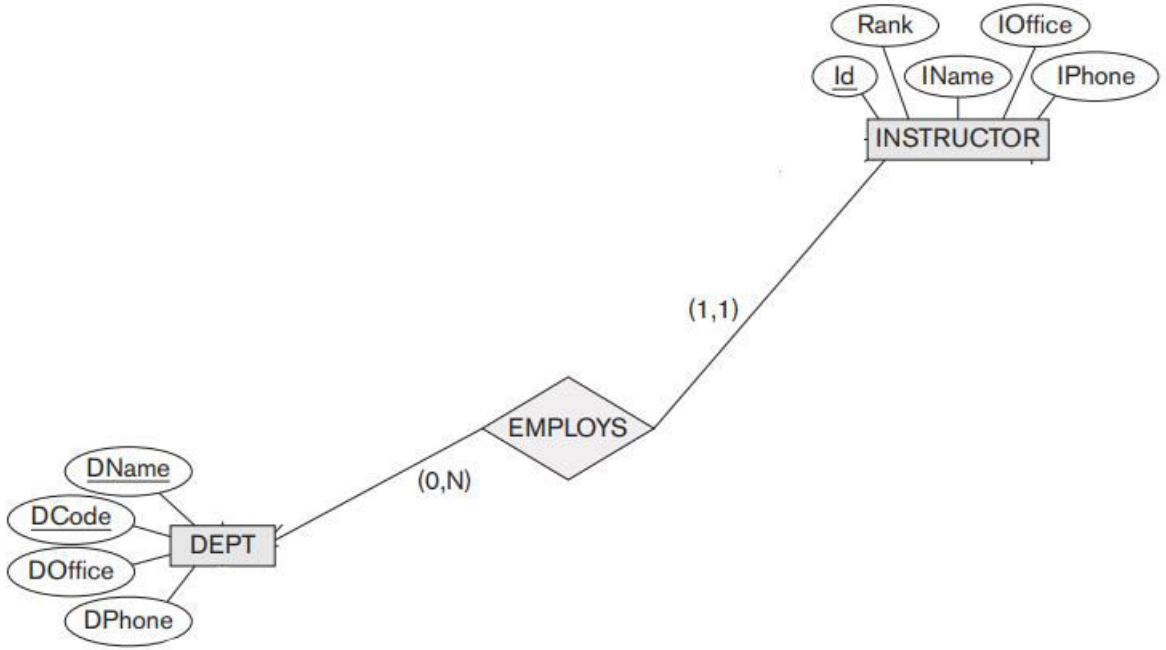
Constraints on Relationships?

3. DEAN between COLLEGE and INSTRUCTOR



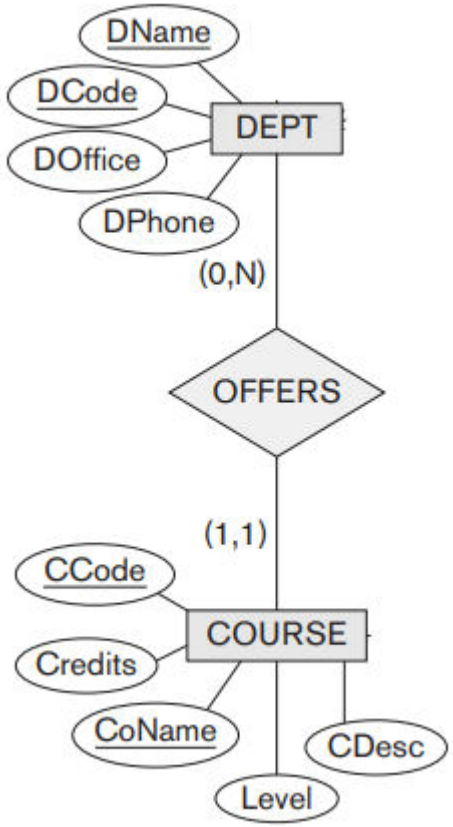
Constraints on Relationships?

4. EMPLOYS between DEPT and INSTRUCTOR



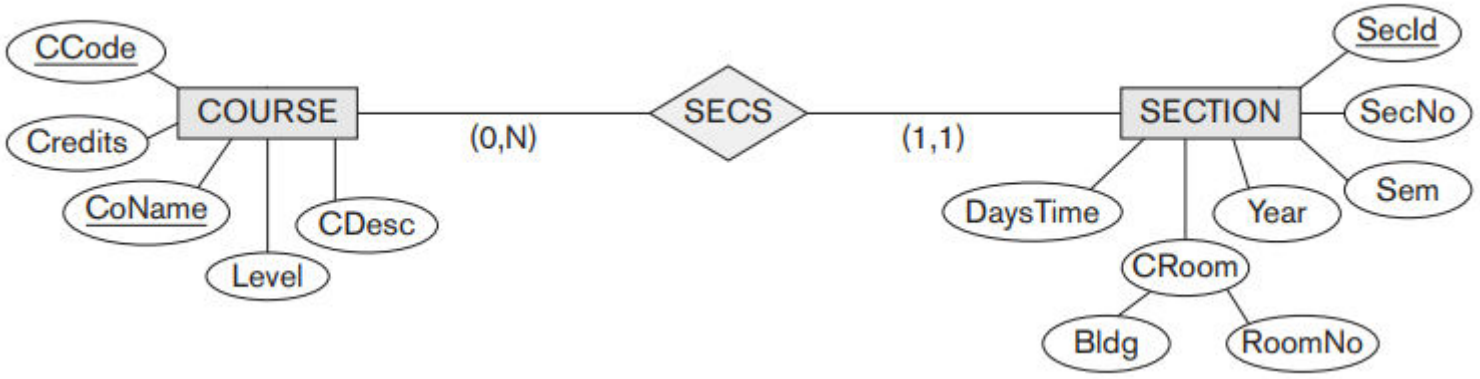
Constraints on Relationships?

5. OFFERS between DEPT and COURSE



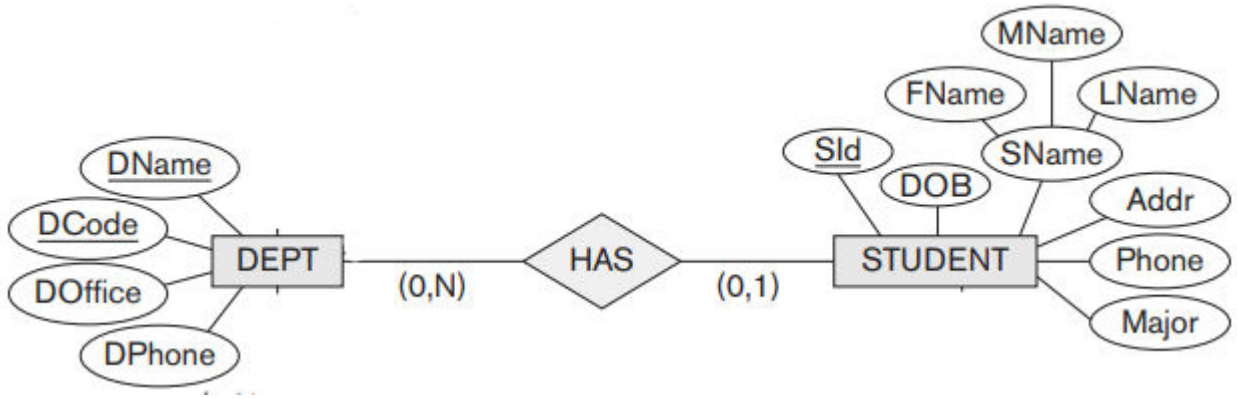
Constraints on Relationships?

6. SECTIONS between COURSE and SECTION



Constraints on Relationships?

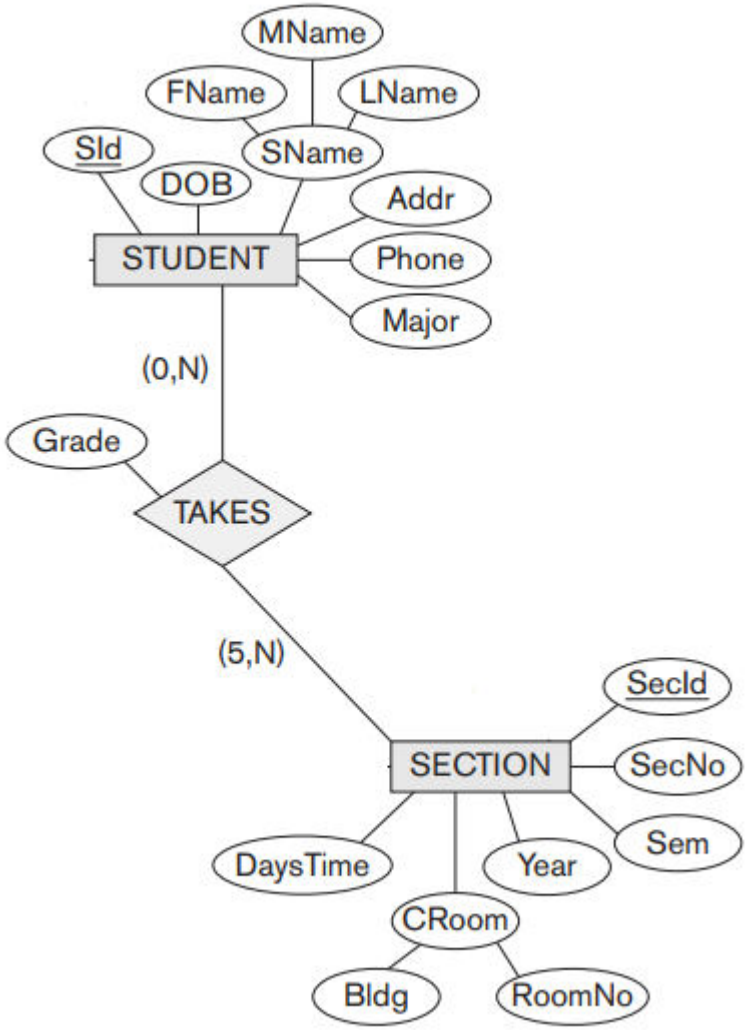
7. HAS between DEPT and STUDENT



Constraints on Relationships?

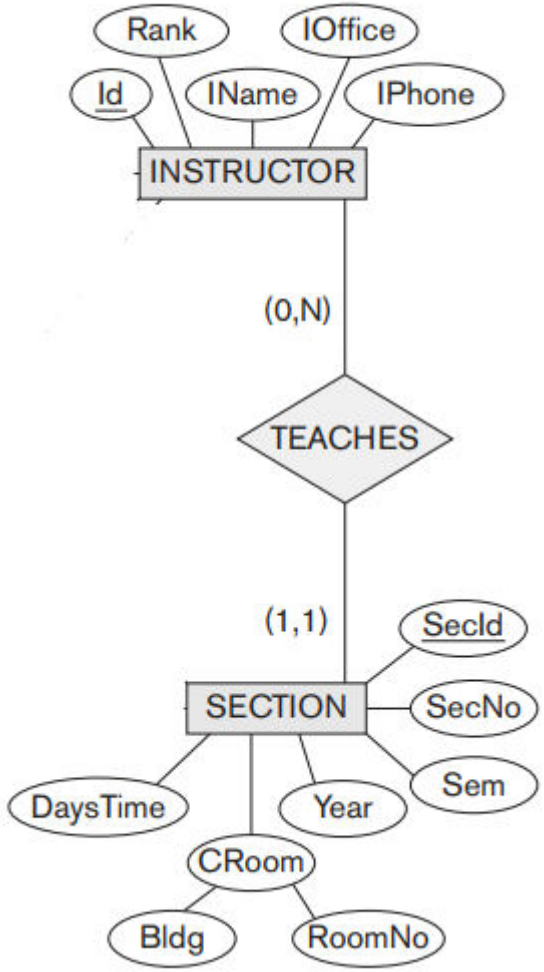
8. TAKES (ENROLS) between STUDENT and SECTION

Note: A section must have at least five students



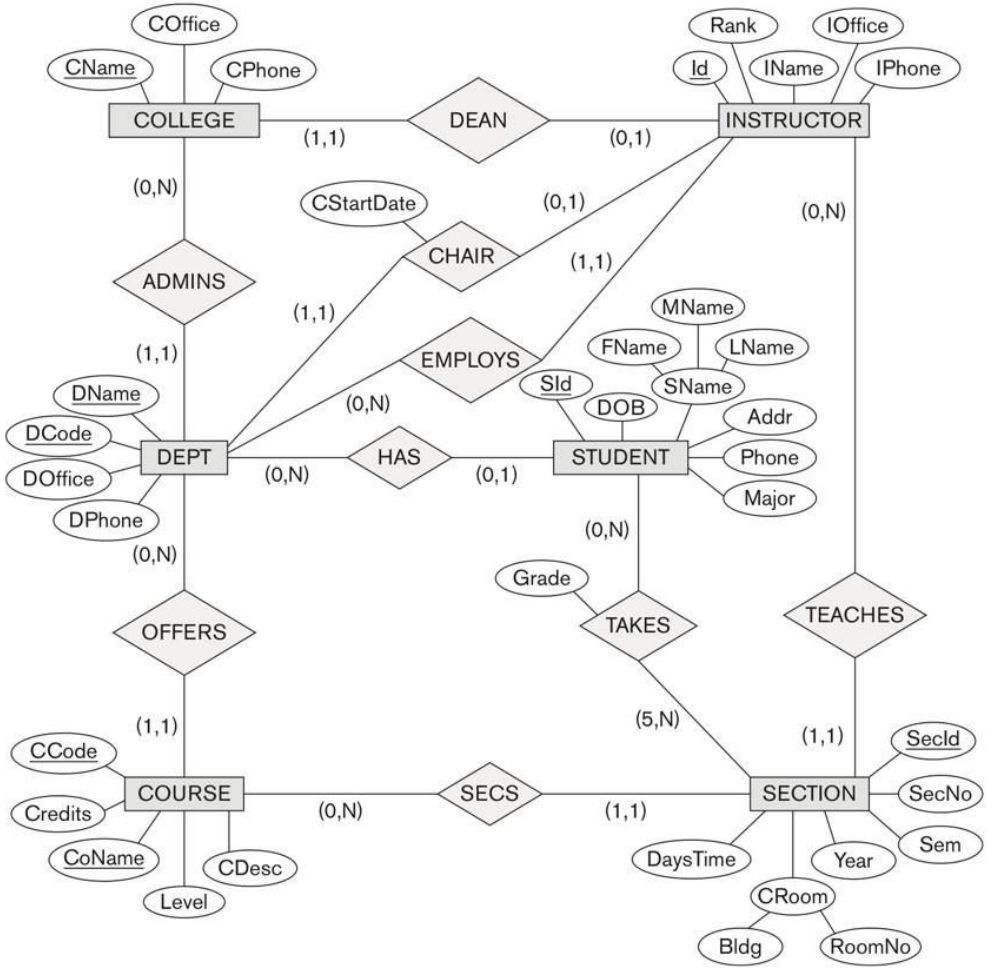
Constraints on Relationships?

9. TEACHES between INSTRUCTOR and SECTION



Database Management Systems

Database Design – University Database ER Diagram



THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu