MCSE506L– Database Systems

Module 1: **Design and Implementation of Relational Model**

Dr. A.Balasundaram

Associate Professor (SCOPE) & Deputy-Director (CCPS)

VIT Chennai

Module 1

Module:1	Design and Implementation of Relational Model	6 hours
Database System Concepts and Architecture, Entity-Relationship (ER) Modelling, Relational Model-Keys, and Integrity Constraints, Mapping ER model to Relational Schema, Normalization, Boyce Codd Normal Form, Multi-valued dependency and Fourth Normal form		

Book(s)

- Text Book:
  - Database System Concepts by Abraham Silberschatz, Henry F.Korth and S.Sudarshan, Tata Mc Graw Hill, 2011
  - Fundamentals of Database Systems by Ramez Elmasri and Shamkant B.Navathe Pearson Education,2013.

Acknowledgement

- Profound thanks to the authors of the book: Fundamentals of Database Systems by Ramez Elmasri and Shamkant B.Navathe Pearson Education,2013. as the content of the book were helpful in preparing this presentation.

Basic Definitions

- Database
- Data
- Database management System (DBMS)
- Database System

Image Source: https://www.theteflcentre.com/news/skills-reading-6-tasks-for-reading-activities-matching-words-to-definitions

Basic Definitions

- **Database**

A collection of **related data**.

Image Source: https://tdwi.org/articles/2019/03/11/dtw-all-five-database-requirements-for-digital-transformation.aspx

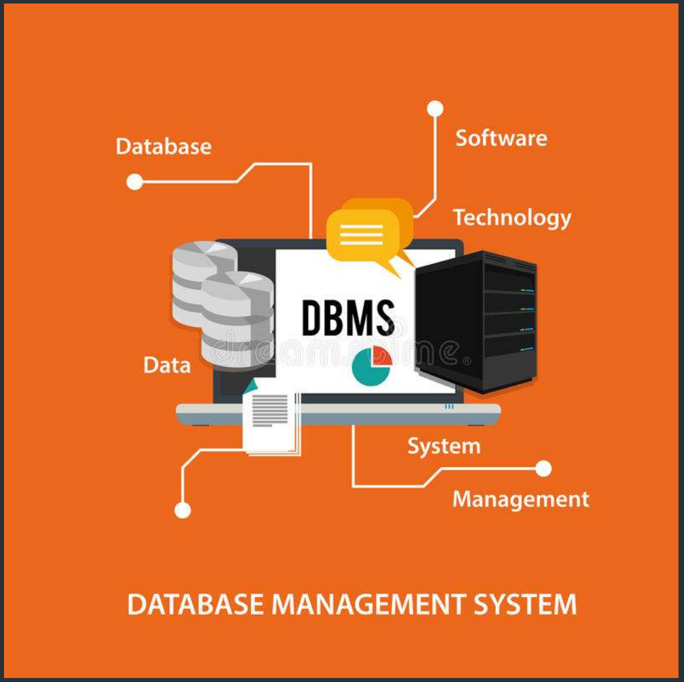
Basic Definitions

- **Data**

Known facts that can be recorded and have an **implicit meaning**.

Image Source: https://analyticsindiamag.com/10-best-data-cleaning-tools-get-data/

Basic Definitions

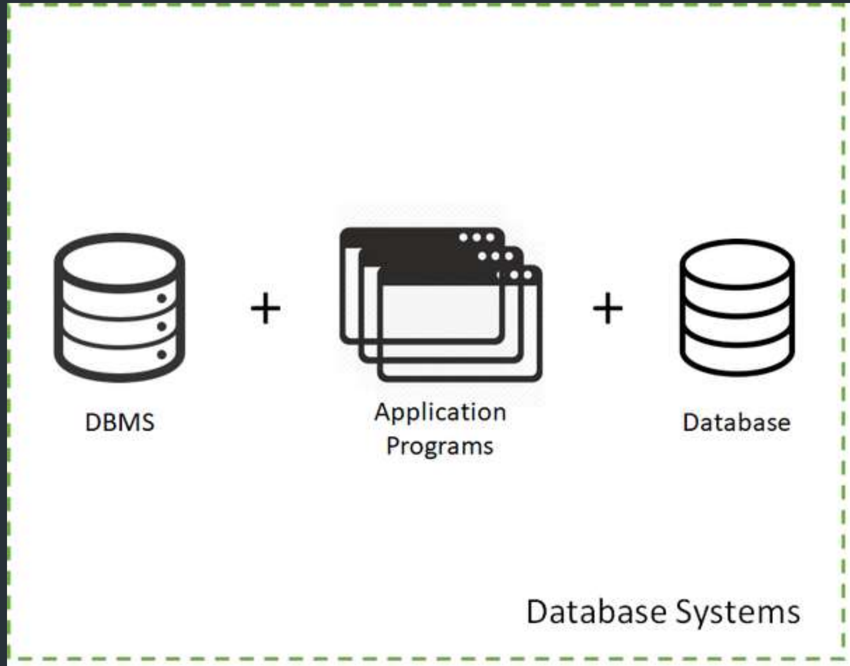


- **Database Management System**

A software package/ system to facilitate the creation and maintenance of a computerized database.

Image Source: https://www.webinhindi.com/2019/04/what-is-dbms-in-hindi.html

Basic Definitions



- **Database System**

The DBMS software together with the data itself. Sometimes, the applications are also included.

Image Source: https://www.quora.com/What-is-database-system

Types of Databases and Database Applications

Image Source: https://www.slideshare.net/PAQUIAAIZEL/types-of-databases

Types of Databases and Database Applications

- **Traditional Applications:**
  - Numeric and Textual Databases
- **More Recent Applications:**
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications

Typical Database Functionalities

- Data definitions
- Data retrieval
- Data manipulation
- Access control
- Data sharing
- Data integrity

Example of a University 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

Image Source: Ramez Elmasri and Shamkant B. Navathe

Example of a University Database

GRADE_REPORT

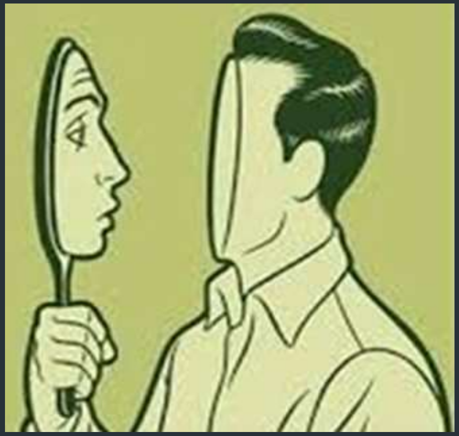
Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Image Source: Ramez Elmasri and Shamkant B. Navathe

Characteristics of Database Approach



- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data**.
  - This allows the DBMS software to work with different database applications.

Image Source: https://www.quora.com/How-should-I-write-my-self-description-in-SSB-interviews

Characteristics of Database Approach

- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

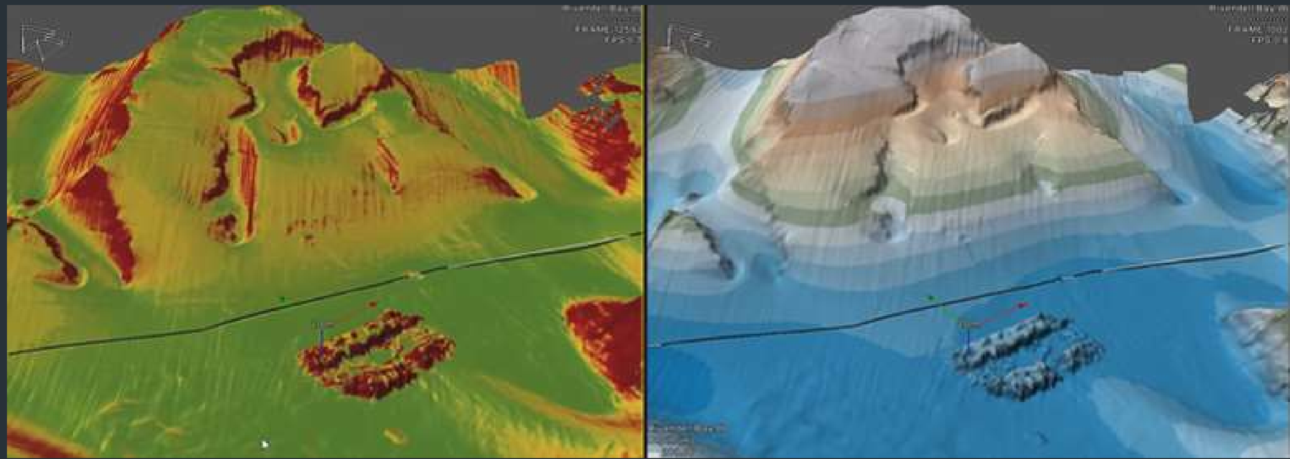
Image Source: https://www.indiamart.com/proddetail/insulation-tape-19584466491.html

Characteristics of Database Approach

- **Data Abstraction:**
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details.

Image Source: https://www.hitechnectar.com/blogs/data-abstraction-level/

Characteristics of Database Approach



- **Support of multiple views of the data:**
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Characteristics of Database Approach

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

Advantages of Database Approach

- Controlling redundancy in data storage and in development and maintenance efforts.
  - Sharing of data among multiple users.
- Restricting unauthorized access to data.
- Providing persistent storage for program Objects

Image Source: https://www.ringlead.com/blog/the-benefits-of-using-database-management-systems

Advantages of Database Approach

- Providing Storage Structures (e.g. indexes) for efficient Query Processing.
- Providing backup and recovery services.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.

Image Source: https://www.ringlead.com/blog/the-benefits-of-using-database-management-systems

Advantages of Database Approach

- Enforcing integrity constraints on the database.
- Drawing inferences and actions from the stored data using deductive and active rules.
- Potential for enforcing standards.
- Reduced application development time.

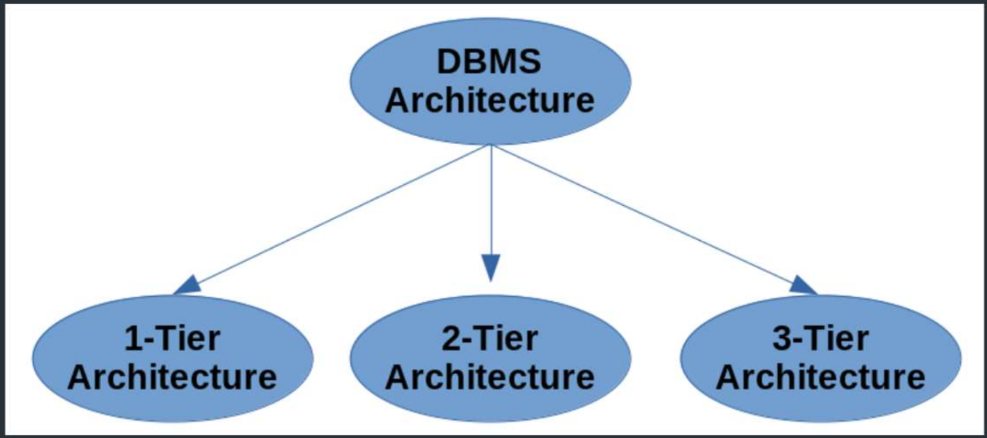
Image Source: https://www.ringlead.com/blog/the-benefits-of-using-database-management-systems

Advantages of Database Approach

- Flexibility to change data structures.
- Availability of current information.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments

Image Source: https://www.ringlead.com/blog/the-benefits-of-using-database-management-systems

DBMS Architecture



- DBMS design depends upon its architecture.
- Client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- Client/server architecture consists of many PCs and a workstation which are connected via the network.

1- Tier Architecture

- In this architecture, the database is directly accessed by the user.
- Any changes will be performed over the database itself.
- Used for development of the local application, where programmers can directly communicate with the database for the quick response.

Image Source: guru99.com

2 - Tier Architecture

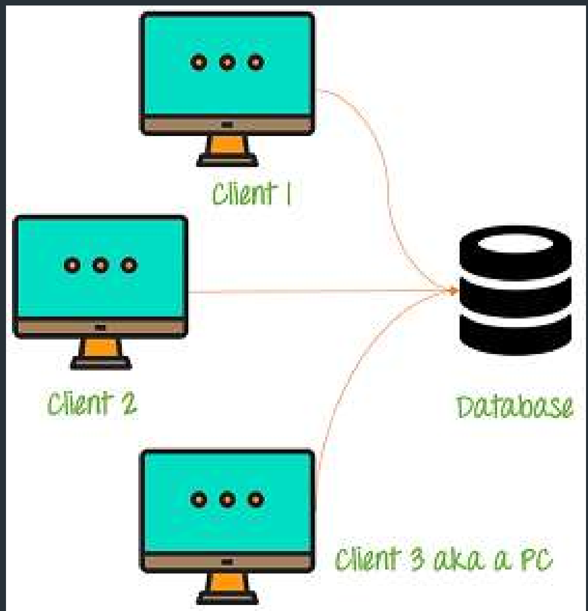
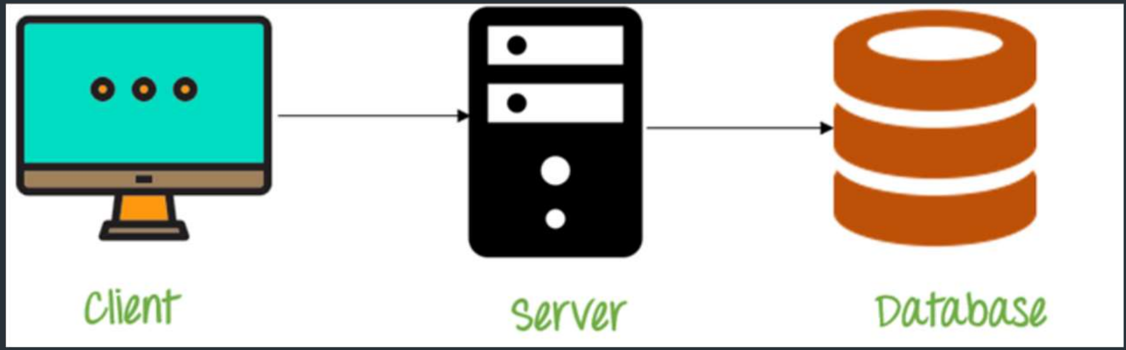


Image Source: guru99.com

- It is basic client-server. But Applications on the client end can directly communicate with the database at the server side.
- For this interaction, API's like: ODBC, JDBC are used. The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

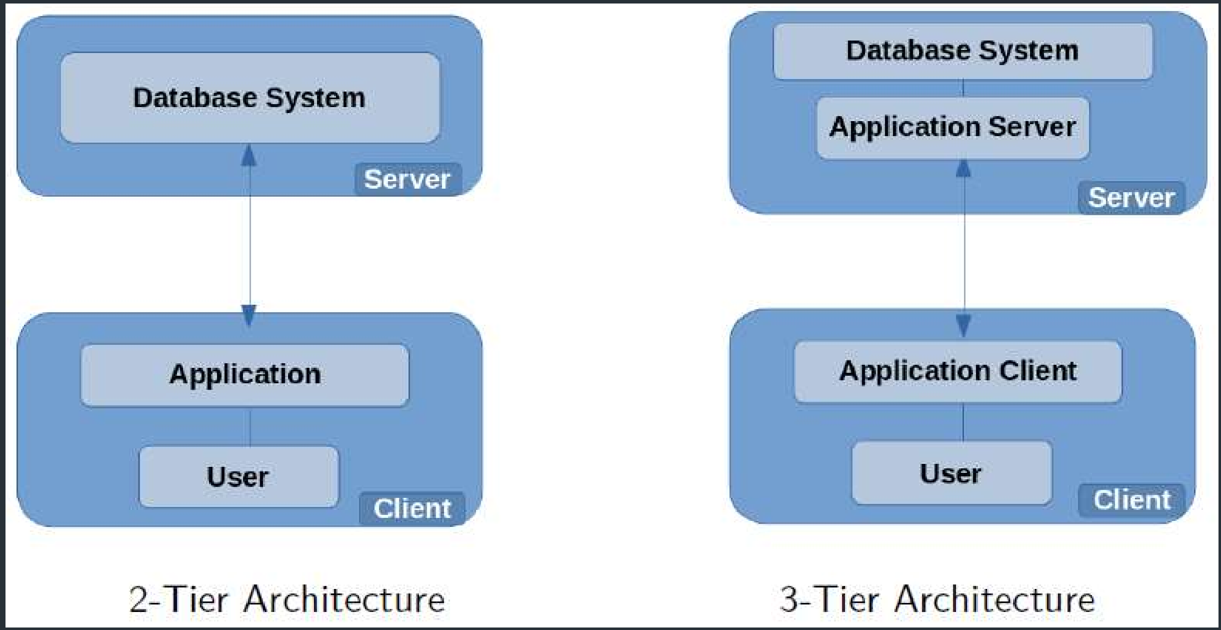
3 - Tier Architecture



- The 3-Tier architecture contains another layer between the client and server.
- In this architecture, client can’t directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server and the database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

Image Source: guru99.com

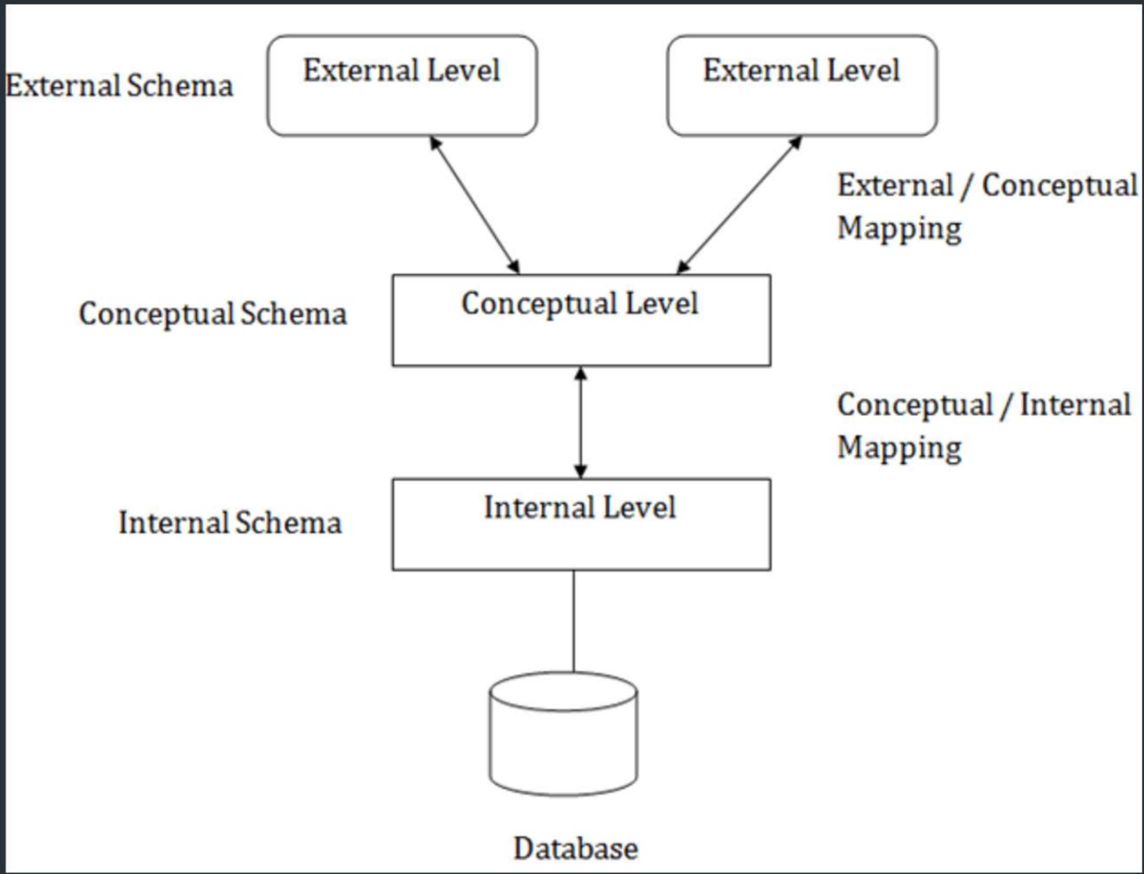
Comparing 2 - Tier & 3 - Tier Architecture



Three Schema Architecture

- It is also called ANSI/SPARC architecture or 3-level architecture.
- It is used to describe the structure of a specific database system.
- It is also used to separate the user applications and physical database.
- It contains 3-levels. It breaks the database down into three different
- categories.
  - **Internal Level**: Actual PHYSICAL storage structure and access paths.
  - **Conceptual or Logical Level**: Structure and constraints for the entire database
  - **External or View level**: Describes various user views

Three Schema Architecture



Three Schema Architecture – Internal Schema

- **Internal level/Schema** : It defines the physical storage structure of the database. It is a very low-level representation of the entire database.
- It contains multiple occurrences of multiple types of internal records. In the ANSI term, it is also called "stored record".
- **Facts about Internal schema**:
- The internal schema is the lowest level of data abstraction.
- It helps you to keep information about the actual representation of the entire database. It is similar to the actual storage of the data on the disk in theform of records
- The internal view tells us what data is stored in the database and how its stored.
- It never deals with the physical devices. Instead, internal schema views a physical device as a collection of physical pages.

Three Schema Architecture – Conceptual Schema

- **Conceptual Schema/Level** :
- It describes the Database structure of the whole database for the community of users.
- It hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc.
- This logical level comes between the user level and physical storage view.
- However, there is only single conceptual view of a single database.
- **Facts about Conceptual schema:**
- Defines all database entities, their attributes, and their relationships.
- Security and integrity information.
- In the conceptual level, the data available to a user must be contained in or derivable from the physical level.

Three Schema Architecture – External Schema

- **External Schema/Level** :
- It describes the part of the database which specific user is interested in & hides the unrelated details of the database from the user.
- There may be "n" number of external views for each database.
- Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.
- An external view is just the content of the database as it is seen by some specific particular user.
- **For example,** a user from the sales department will see only sales related data.

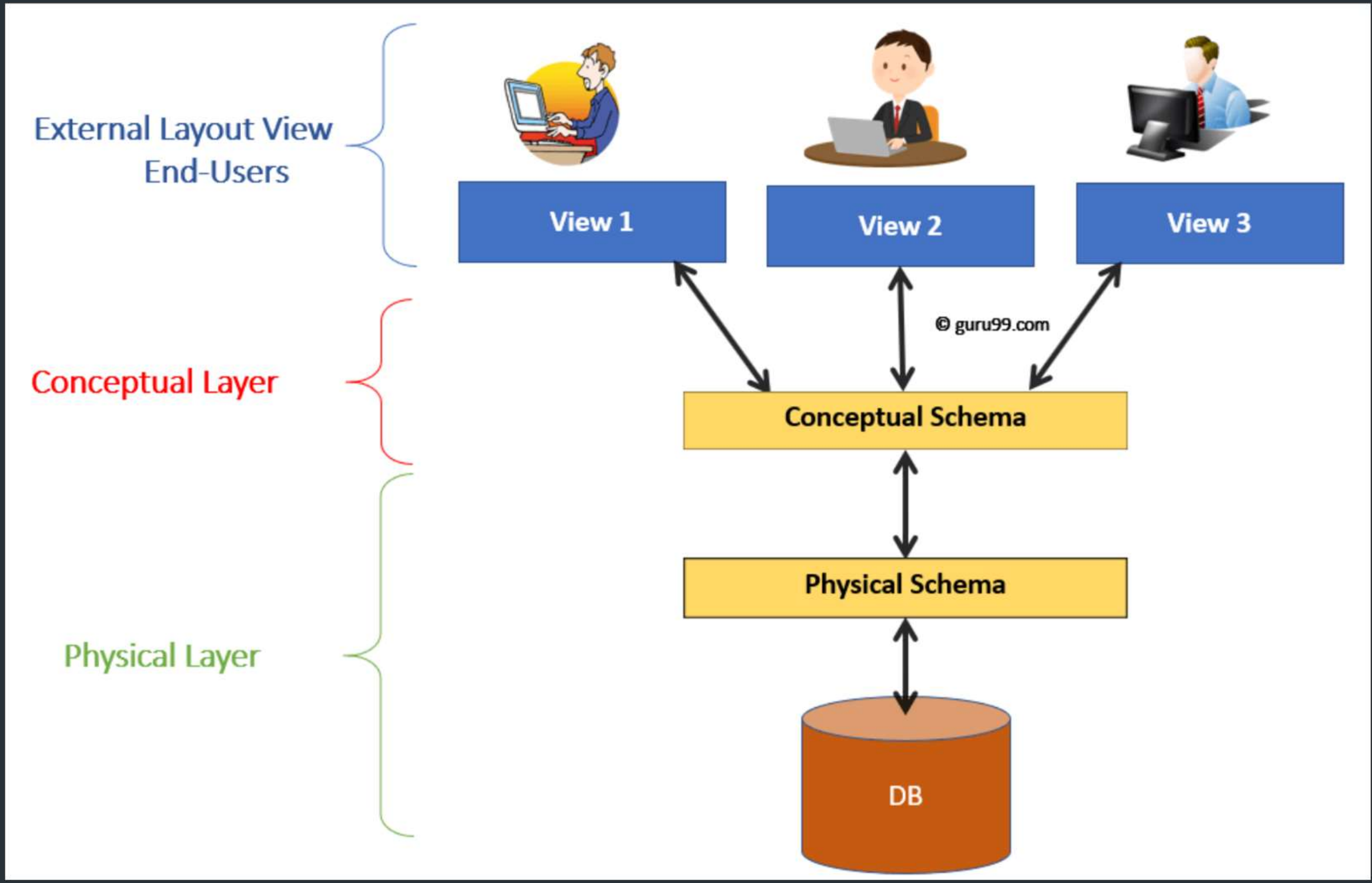
Objectives of Three Schema Architecture

- An external level is only related to the data which is viewed by specific end users.
- This level includes some external schemas.
- External schema level is nearest to the user.
- The external schema describes the segment of the database which is needed for a certain user group and hides the remaining details from the database from the specific user group.

Data Independence

- Data Independence is defined as a property of DBMS that helps the user to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Data independence helps the user to keep data separated from all programs that make use of it.
- **Types of Data Independence**
  - Physical Data Independence
  - Logical Data Independence

Levels of Database



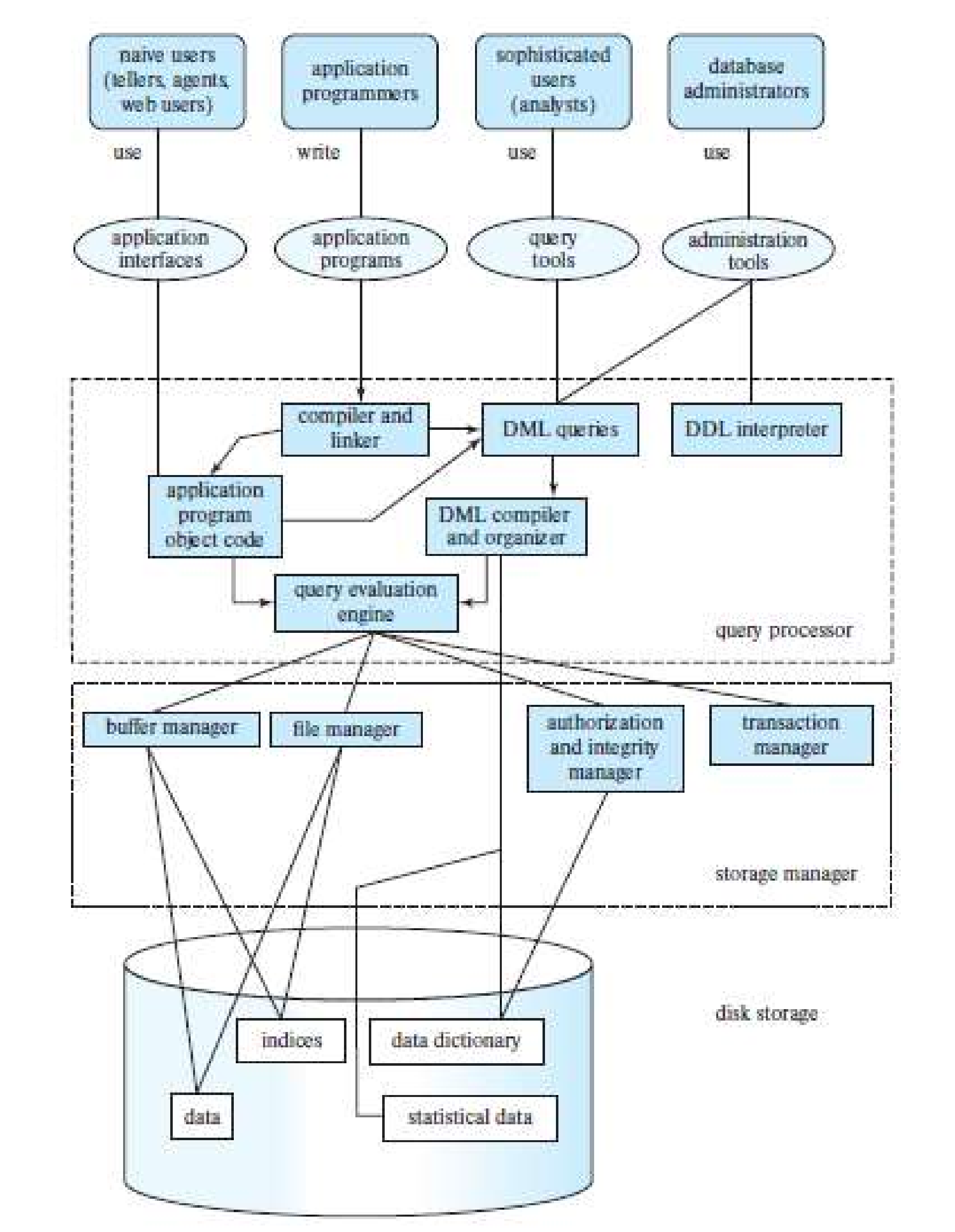
Levels of Database

Type of Schema	Implementation
External Schema	**View 1**: Course info(cid:int,cname:string) **View 2**: studeninfo(id:int. name:string)

Conceptual Shema	`Students(id: int, name: string, login: string, age: integer)`
	`Courses(id: int, cname.string, credits:integer)`
	`Enrolled(id: int, grade:string)`

Physical Schema	• Relations stored as unordered files. • Index on the first column of Students.

DBMS Architecture



DBMS Components

- DML : DML processor must interact with the query processor to generate the appropriate code
- DDL interacts with Data Dictionary/ System Catalog
- System Catalog : It is a collection of tables and views that contain important information about a database. It is available for each database. It defines the structure of the database.
- For example, the DDL (data dictionary language) for all tables in the database is stored in the system catalog.
- Query processor : It transforms user queries into a series of low level instructions. It is used to interpret the online user's query and convert it into an efficient series of operations in a form capable of being sent to the run time data manager for execution.

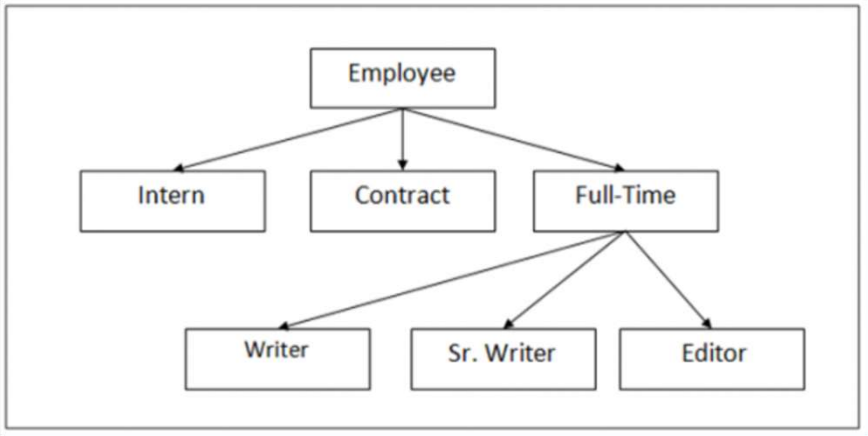
DBMS Components

- It uses the data dictionary to find the structure of the relevant portion of the database and uses this information in modifying the query and preparing and optimal plan to access the database.
- Data Dictionary : It contains all the information about the database. As the name suggests, it is the dictionary of all the data items. It contains description of all the tables, view, materialized views, constraints, indexes, triggers etc.

DBMS Languages

- **Data Definition Language (DDL)** allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints.
- **Data Manipulation Language (DML)** provides basic data manipulation operations on data held in the database.
- **Data Control Language (DCL)** defines activities that are not in the categories of those for the DDL and DML, such as granting privileges to users, and defining when proposed changes to a databases should be irrevocably made.
- **Low Level or Procedural DML**: allow user to tell system exactly how to manipulate data (e.g., Network and hierarchical DMLs).
- **High Level or Non-procedural DML**(declarative language): allow user to state what data is needed rather than how it is to be retrieved (e.g., SQL).

Hierarchical Data Model

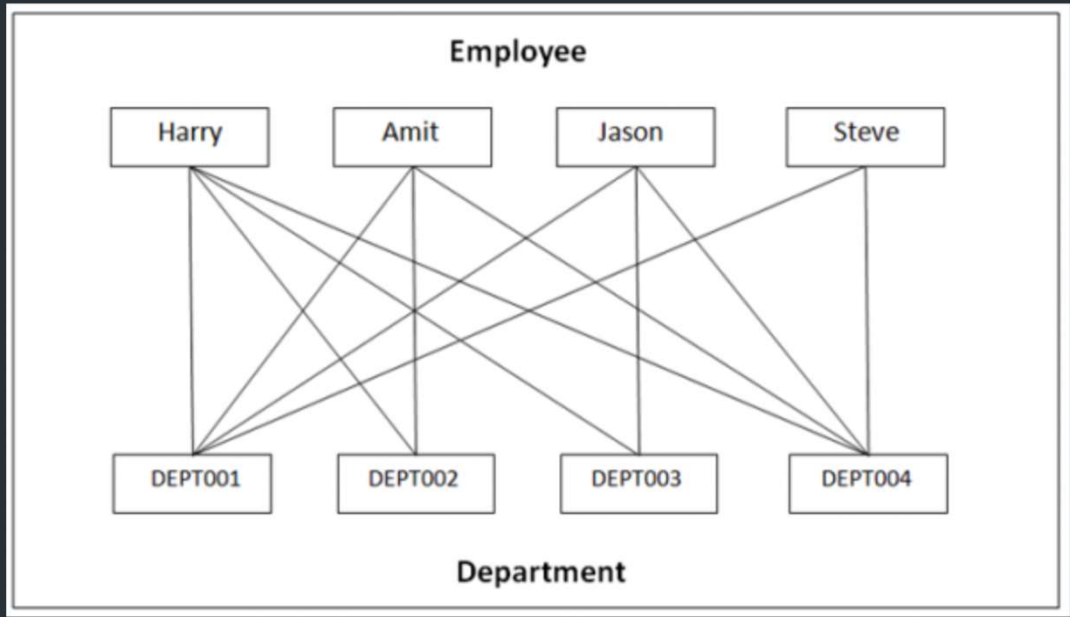


- In Hierarchical Model, a hierarchical relation is formed by collection of relations and forms a tree-like structure.
- The relationship can be defined in the form of parent child type.
- One of the first and most popular Hierarchical Model is Information Management System (IMS), developed by IBM.

Hierarchical Data Model - Merits and Demerits

- **Merits**
  - The design of the hierarchical model is simple.
  - Provides Data Integrity since it is based on parent/ child relationship
  - Data sharing is feasible since the data is stored in a single database.
  - Even for large volumes of data, this model works perfectly.
- **De-Merits**
  - Implementation is complex.
  - This model has to deal with anomalies like Insert, Update and Delete.
  - Maintenance is difficult since changes done in the database may want you to do changes in the entire database structure.

Network Data Model



- The Hierarchical Model creates hierarchical tree with parent/ child relationship, whereas the Network Model has graph and links.
- The relationship can be defined in the form of links and it handles many-to-many relations. This itself states that a record can have more than one parent.

Network Data Model - Merits and Demerits

- **Merits**
  - Easy to design the Network Model
  - The model can handle one-one, one-to-many, many-to-many relationships.
  - It isolates the program from other details.
  - Based on standards and conventions.
- **De-Merits**
  - Pointers bring complexity since the records are based on pointers and graphs.
  - Changes in the database isn’t easy that makes it hard to achieve structural independence.

Relational Data Model

	Column1	Column2	Column3
Row1			
Row2			
Row3			
Row4			
Row5			

- A Relational model groups data into one or more tables. These tables are related to each other using common records.
- The data is represented in the form of rows and columns i.e. tables.

Relational Data Model

<Employee>

EmployeeID	**EmpName**	**DepartmentID**
E09	Emily	D001
E04	Tom	D002
E11	Emma	D003

<Department>

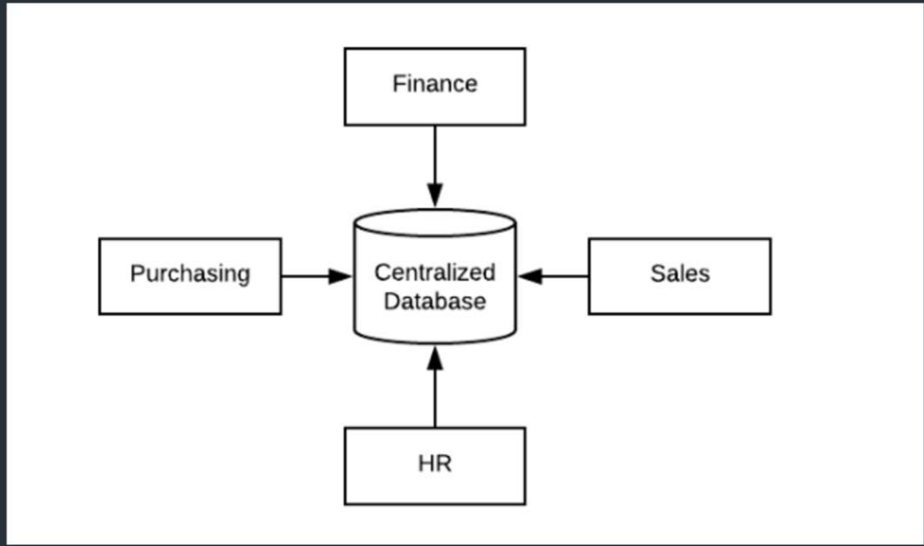
DepartmentID	**DeptName**	**DeptZone**
D001	Finance	North
D002	Operations	East
D003	Marketing	West

- **Example** : Let us see an example of two relations <**Employee**> and <**Department**> linked to each other, with DepartmentID, which is **Foreign Key** of <**Employee**> table and Primary key of <Department> table.

Relational Data Model - Merits and Demerits

- **Merits**
  - It does not have any issues that we saw in the previous two models i.e. update, insert and delete anomalies have nothing to do in this model.
  - Changes in the database do not require you to affect the complete database.
  - Implementation of a Relational Model is easy.
  - To maintain a Relational Model is not a tiresome task.
- **De-Merits**
  - Database inefficiencies hide and arise when the model has large volumes of data.
  - The overheads of using relational data model come with the cost of using powerful hardware and devices.

Centralized Database Management System



- A centralized database is stored at a single location such as a mainframe computer.
- It is maintained and modified from that location only and usually accessed using an internet connection such as a LAN or WAN.
- The centralized database is used by organizations such as colleges, companies, banks etc.

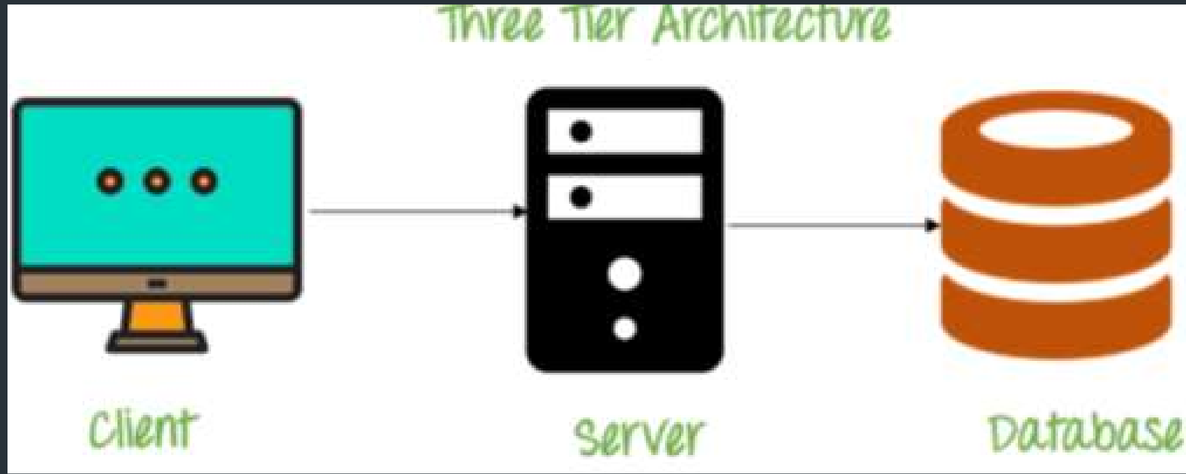
Merits and De-merits of Centralized DBMS

- **Advantages** :
- The **Data Integrity** is maximized as the whole database is stored at a single physical location. It is easier to coordinate the data and it is as accurate and consistent as possible.
- The **Data Redundancy** is minimal in the centralized database. All the data is stored together and not scattered across different locations. So, there is no redundant data available.
- Since all the data is in one place, there can be stronger security measures around it. So, It is much more secure.
- Data is easily portable because it is stored at the same place.
- It is cheaper than other types of databases as it requires less power and maintenance.
- All the information can be easily accessed from the same location and at the same time.

Merits and De-merits of Centralized DBMS

- **Disadvantages** :
- Since all the data is at one location, it takes more time to search and access it. If the network is slow, this process takes even more time.
- There is a lot of data access traffic for the centralized database. This may create a bottleneck situation.
- Since all the data is at the same location, if multiple users try to access it simultaneously it creates a problem. This may reduce the efficiency
- of the system.
- If there are no database recovery measures in place and a system failure occurs, then all the data in the database will be destroyed.

Client-Server Database Management System



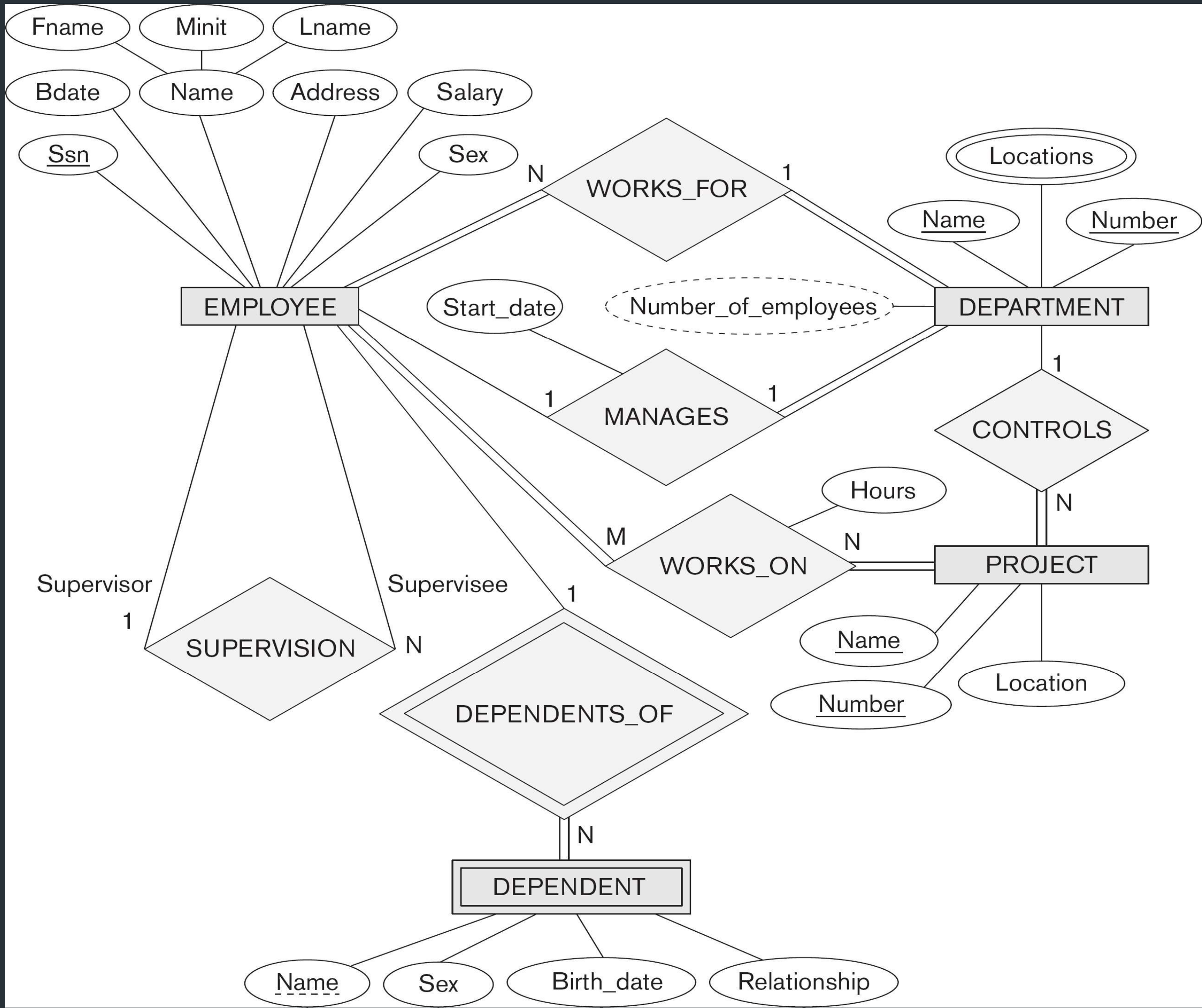
- A client does not share any of its resources, but requests a server's content or service function.
- Clients therefore initiate communication sessions with servers which await incoming requests.
- Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

Image Source: guru99.com

ER Diagram Example

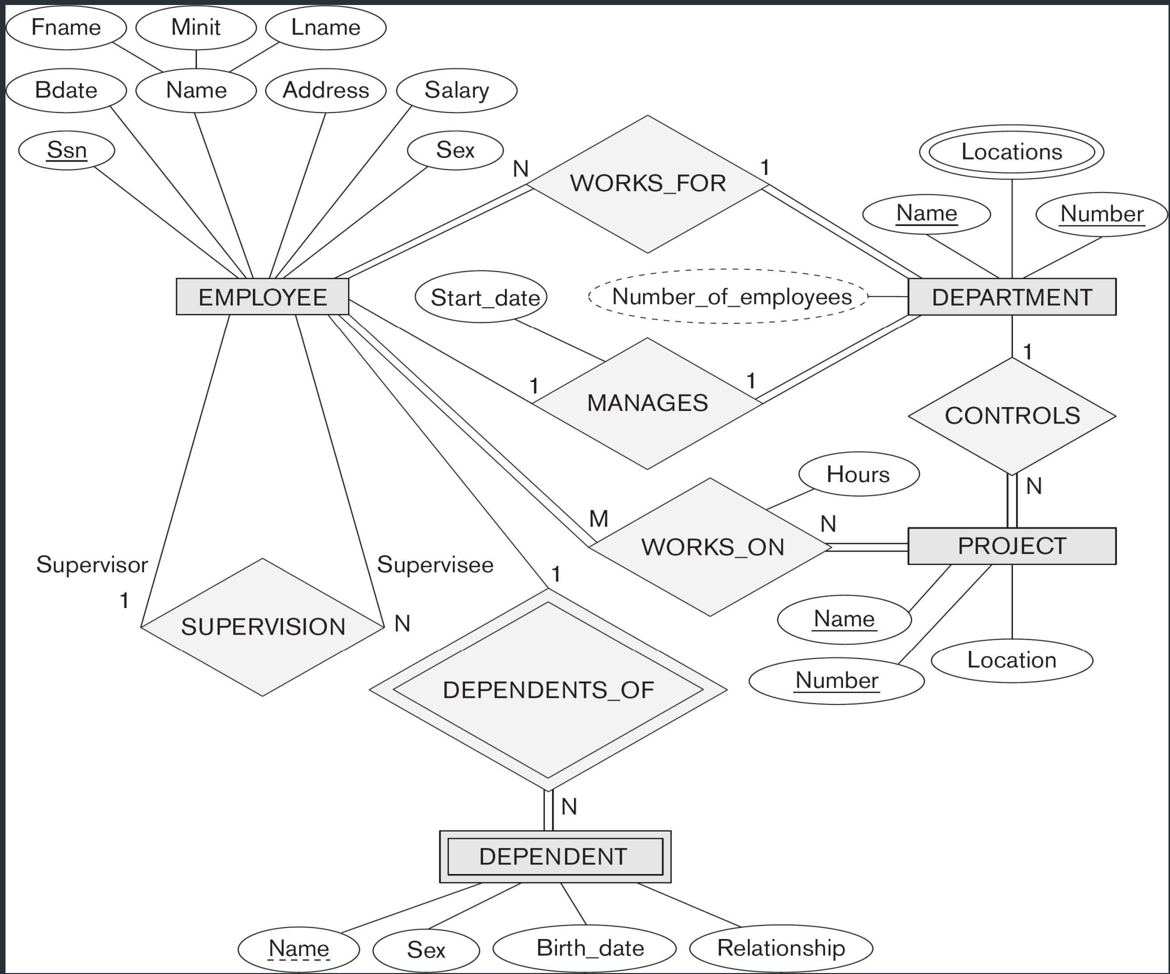
Draw an ER diagram for a Company based on the conditions specified as follows:

- A company has several departments. Each department may have several Locations.
- Departments are identified by a name, d_no, Location.
- A Manager control a particular department.
- Each department is associated with number of projects.
- Employees are identified by name, id, address, job, date_of_joining.
- An employee works in only one department but can work on several project.
- We also keep track of number of hours worked by an employee on a single project.
- Each employee has a dependent
- Dependent has D_name, Gender, and relationship.


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Number
Location
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

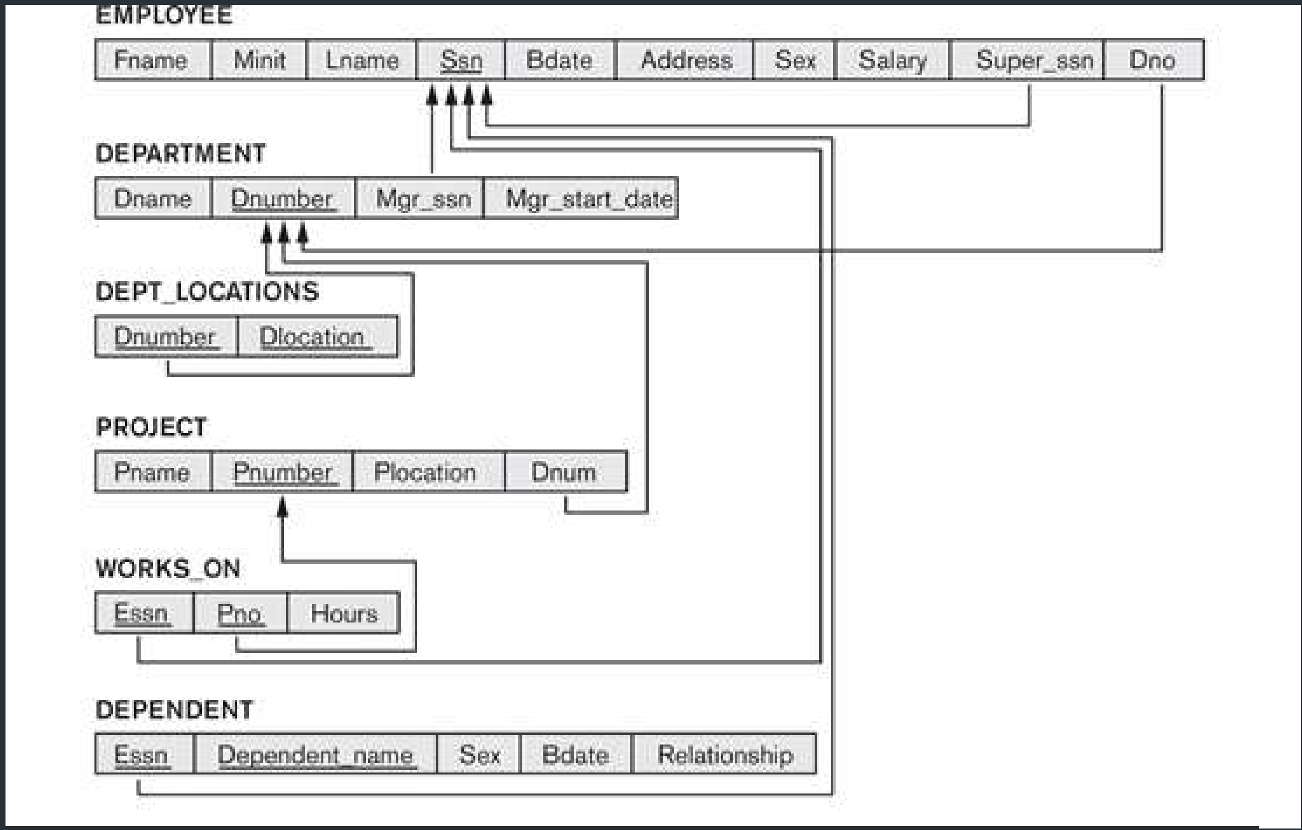
ER-to-Relational Mapping

Example ERD



Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Number
Location
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Resulting Relational Schema



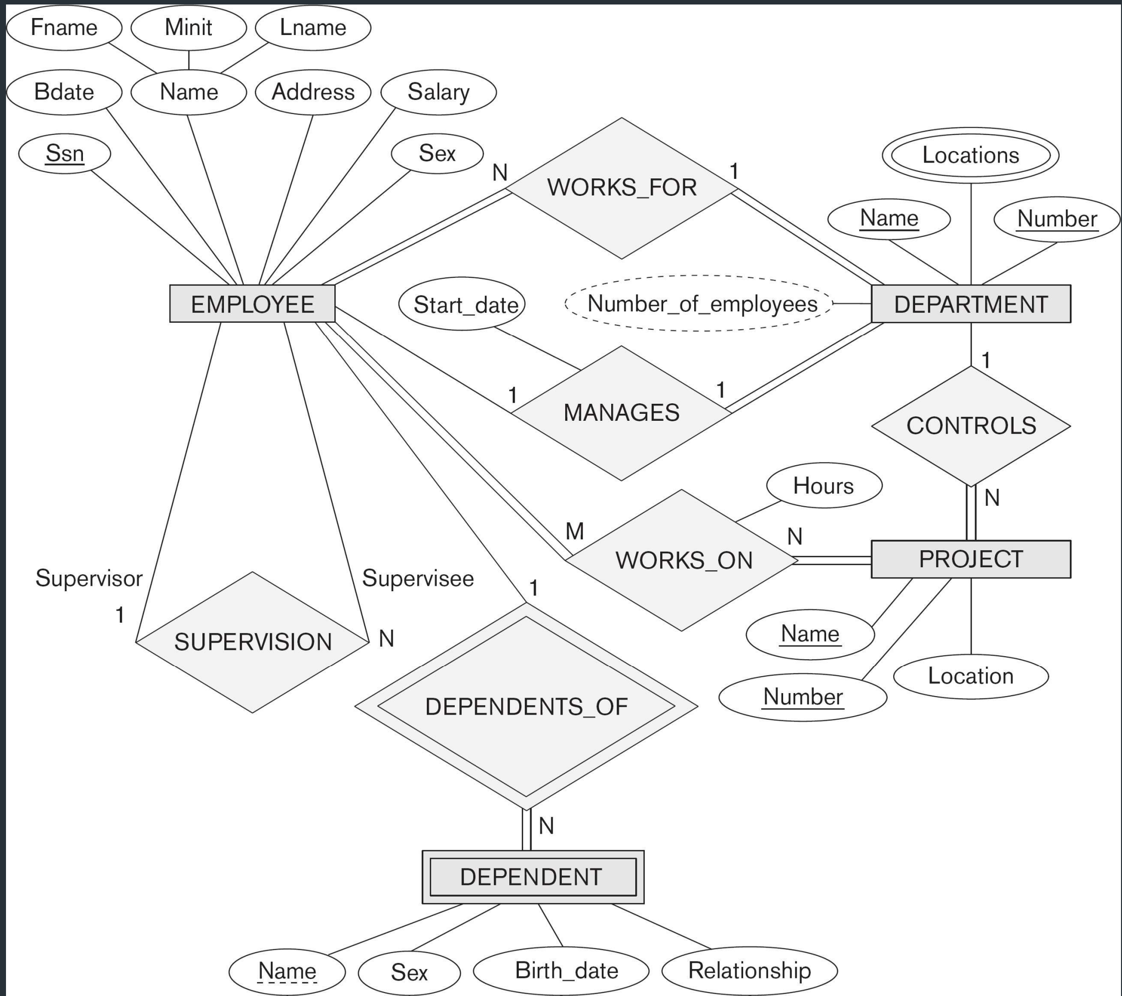
Step 1: Regular Entity Types

i. For each regular/strong entity type, create a corresponding relation that includes all the **simple** attributes (includes simple attributes of composite relations)

ii. Choose one of the key attributes as primary
   - If composite, the simple attributes together form the primary key

iii. Any remaining key attributes are kept as secondary unique keys (these will be useful for physical tuning w.r.t. indexing analysis)

Example ERD



Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Step 1 Result

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	<u>Dnumber</u>

PROJECT

Pname	<u>Pnumber</u>	Plocation

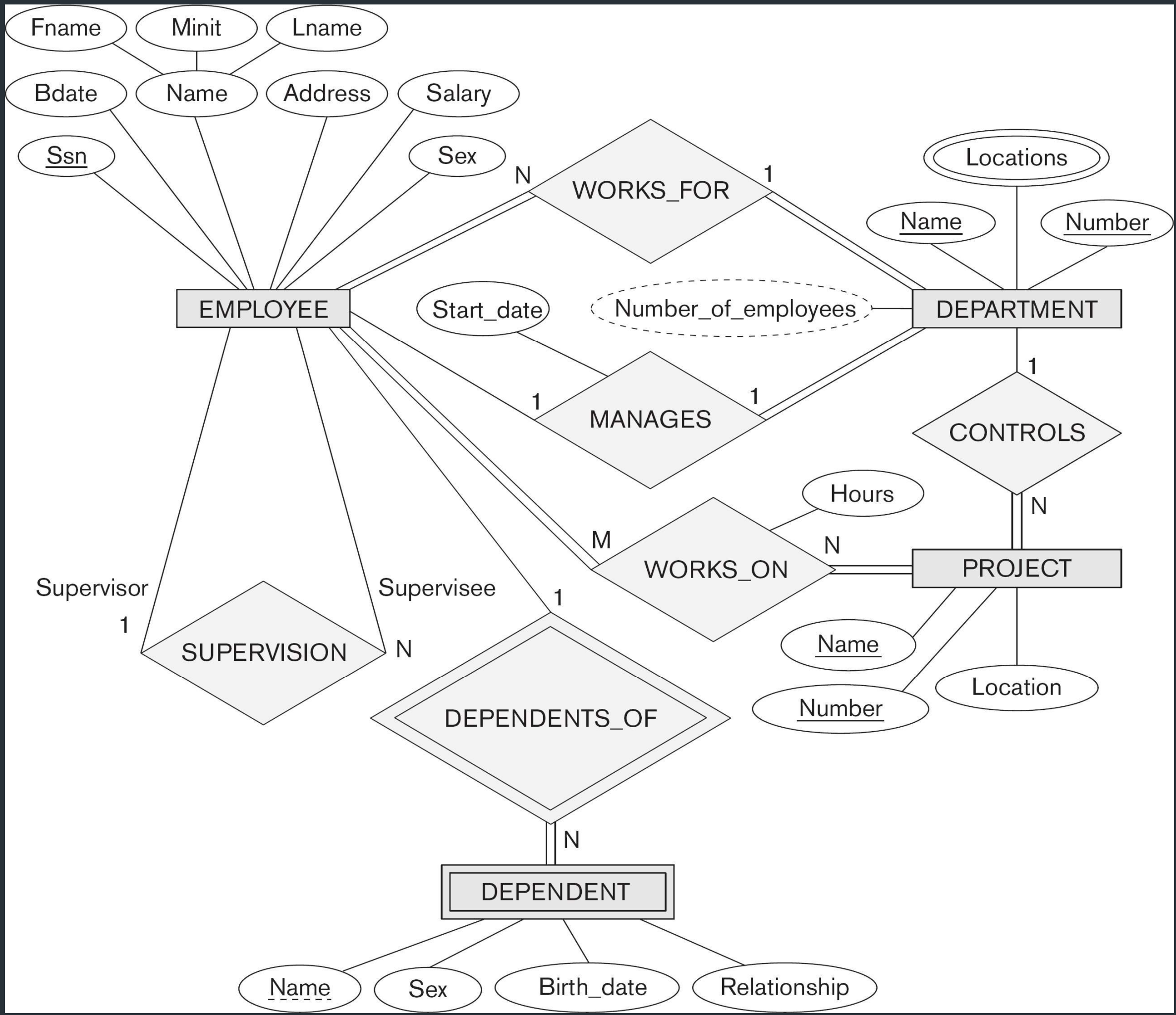
Step 2: Weak Entity Types

i. For each weak entity type, create a corresponding relation that includes all the simple attributes

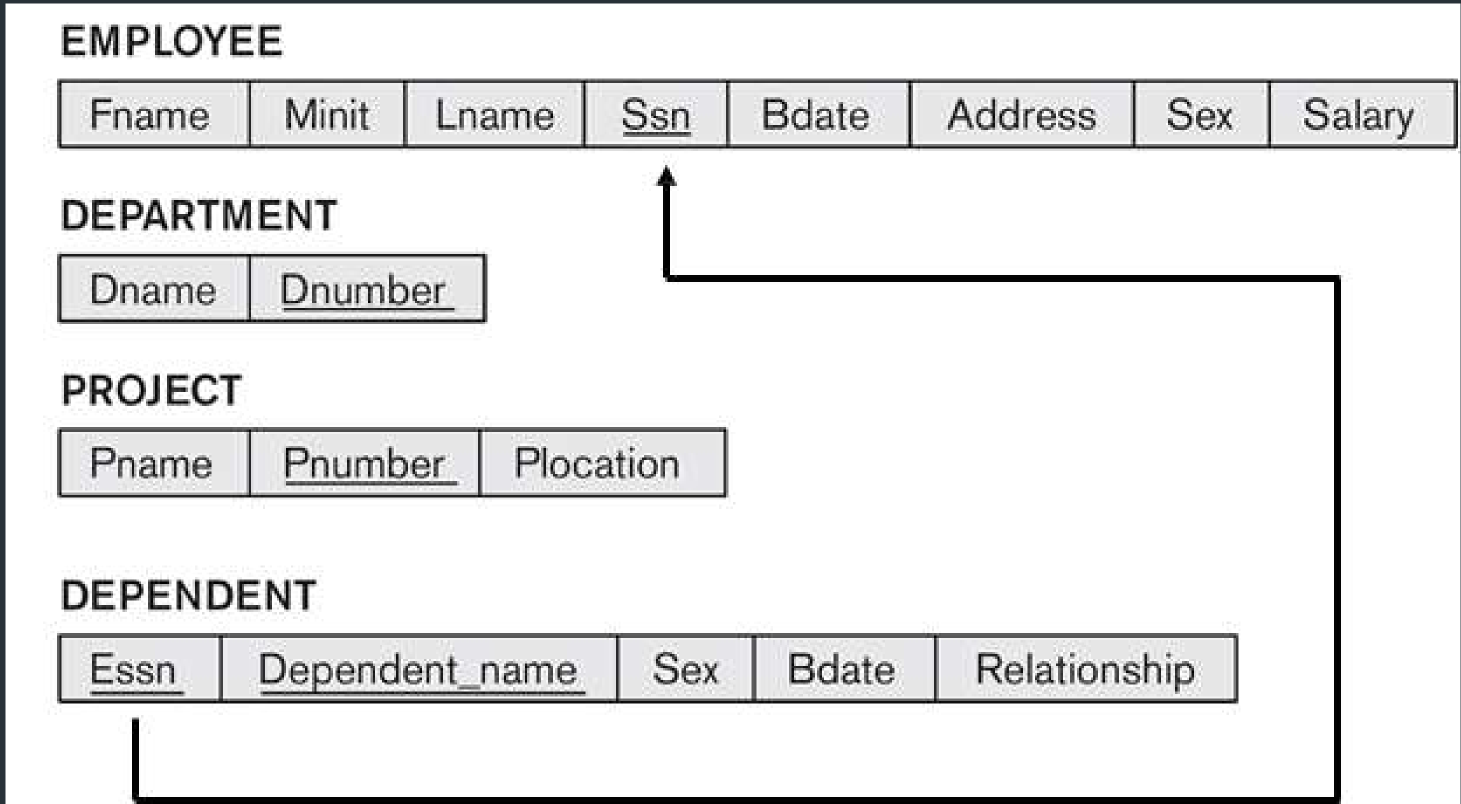
ii. Add as a **foreign key** all of the primary key attribute(s) in the entity corresponding to the owner entity type

iii. The primary key is the combination of all the primary key attributes from the owner and the partial key of the weak entity, if any

Example ERD


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
DEPENDENTS_OF
Number
Location
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Step 2 Result



Step 3: Mapping Binary 1-to-1

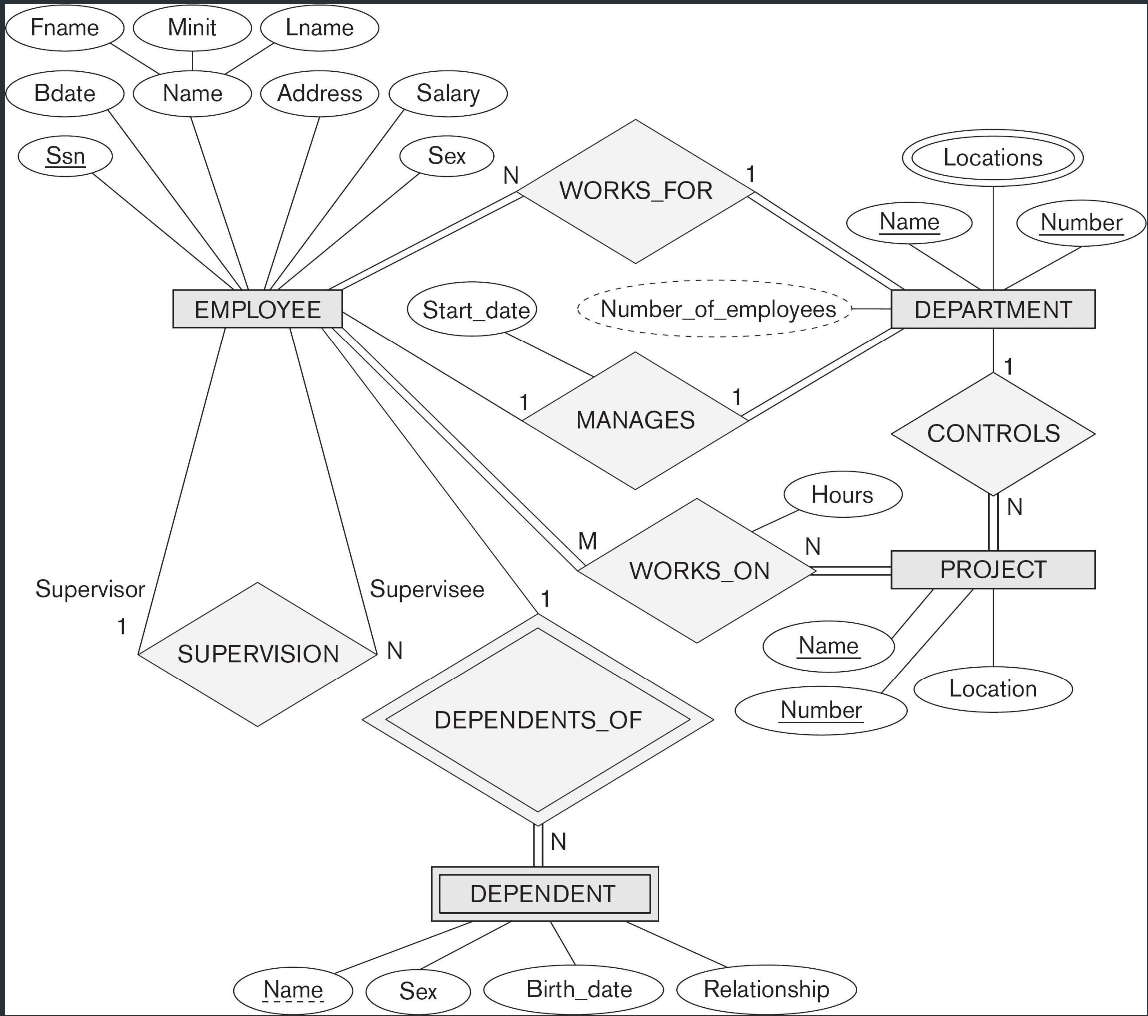
Foreign Key

i. Choose one relation as *S*, the other *T*
   - Better if S has total participation (reduces number of NULL values)

ii. Add to *S* all the simple attributes of the relationship

iii. Add as a foreign key in *S* the primary key attributes of *T*

Example ERD



Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Step 3 Result

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

Step 3 Result

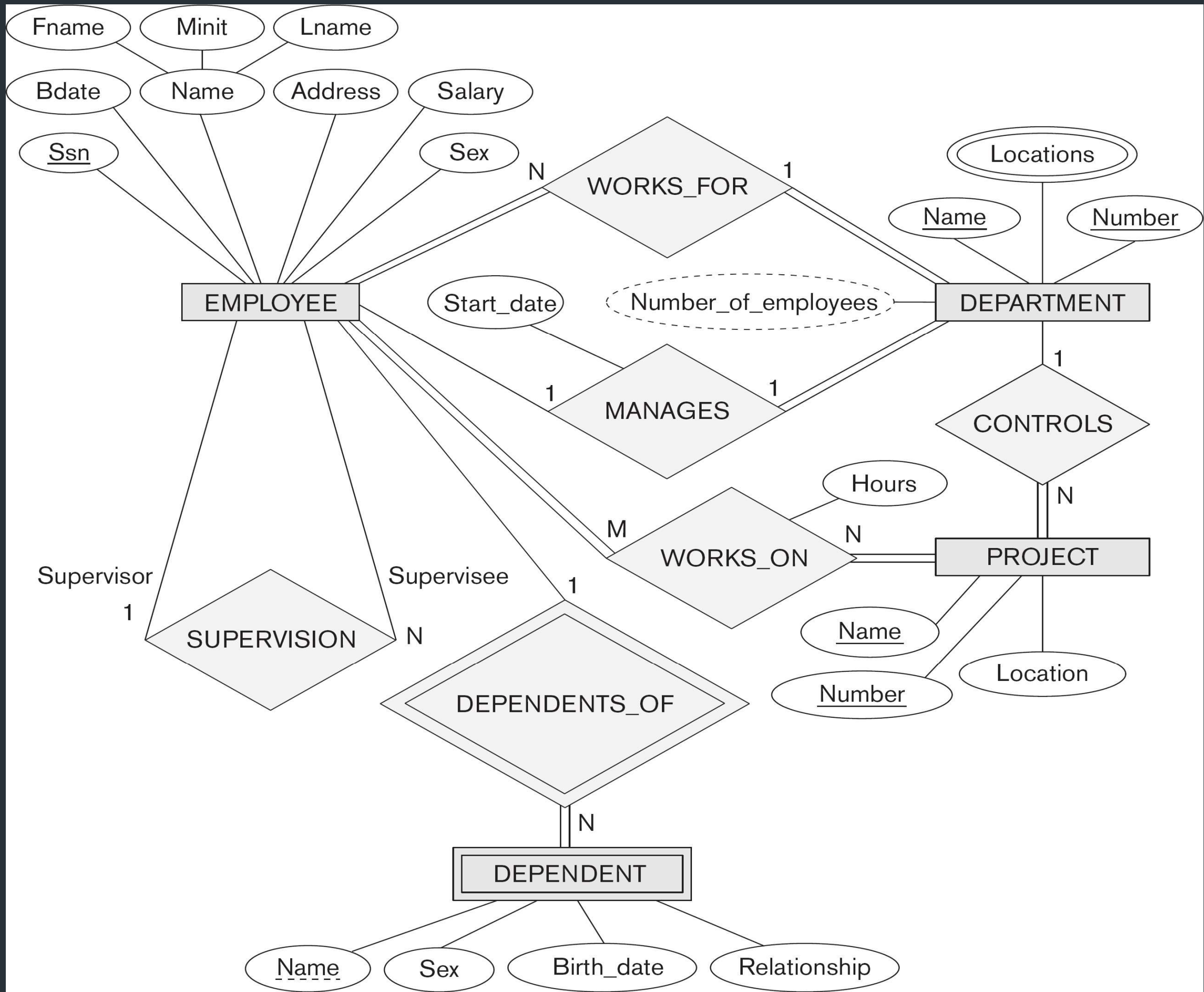
Step 4: Binary 1-to-N

i. Choose the *S* relation as the type at the N-side of the relationship, other is *T*

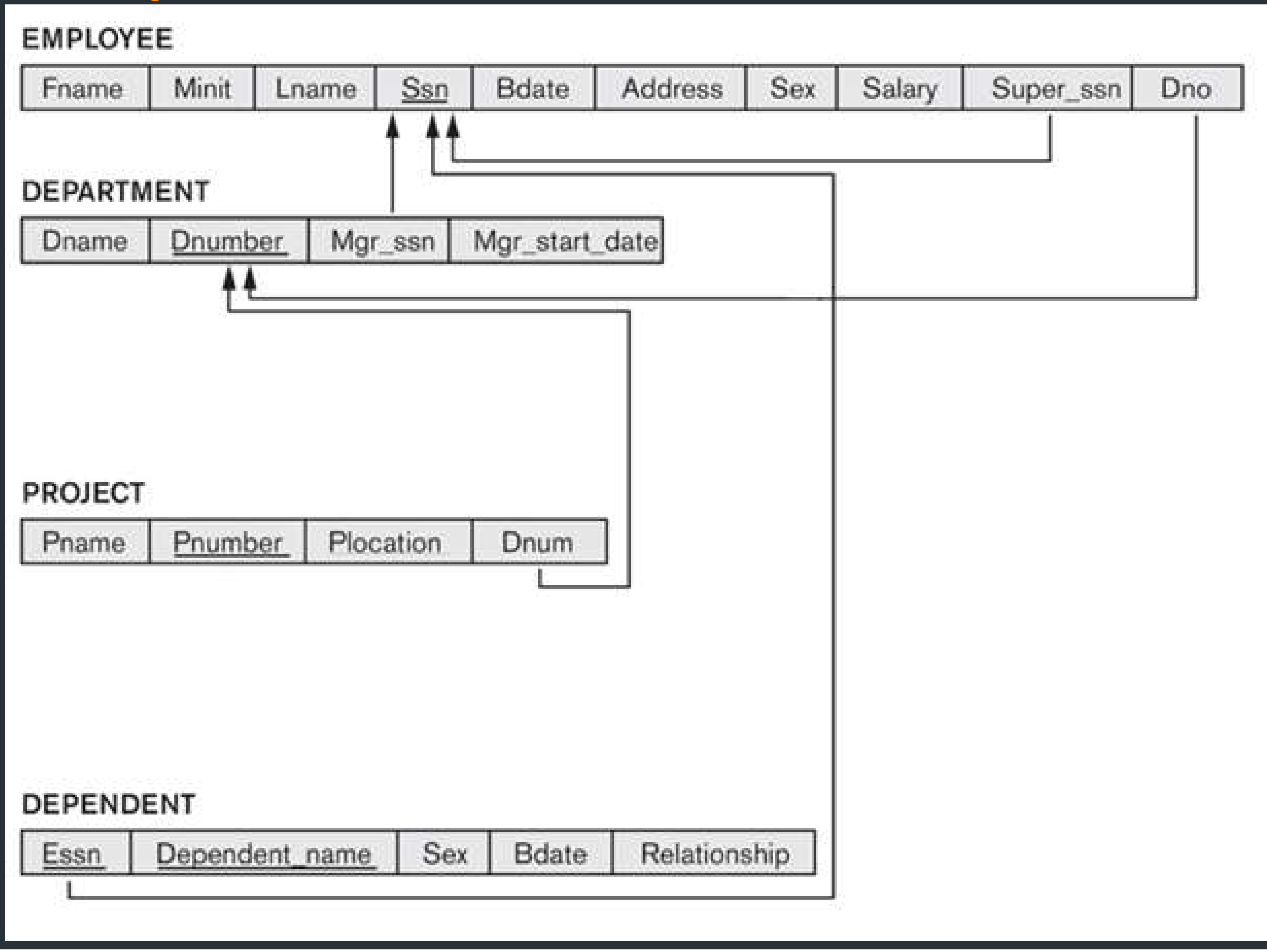
ii. Add as a **foreign key** to *S* all of the primary key attribute(s) of *T*

Another approach: create a relationship relation

Example ERD


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Step 4 Result



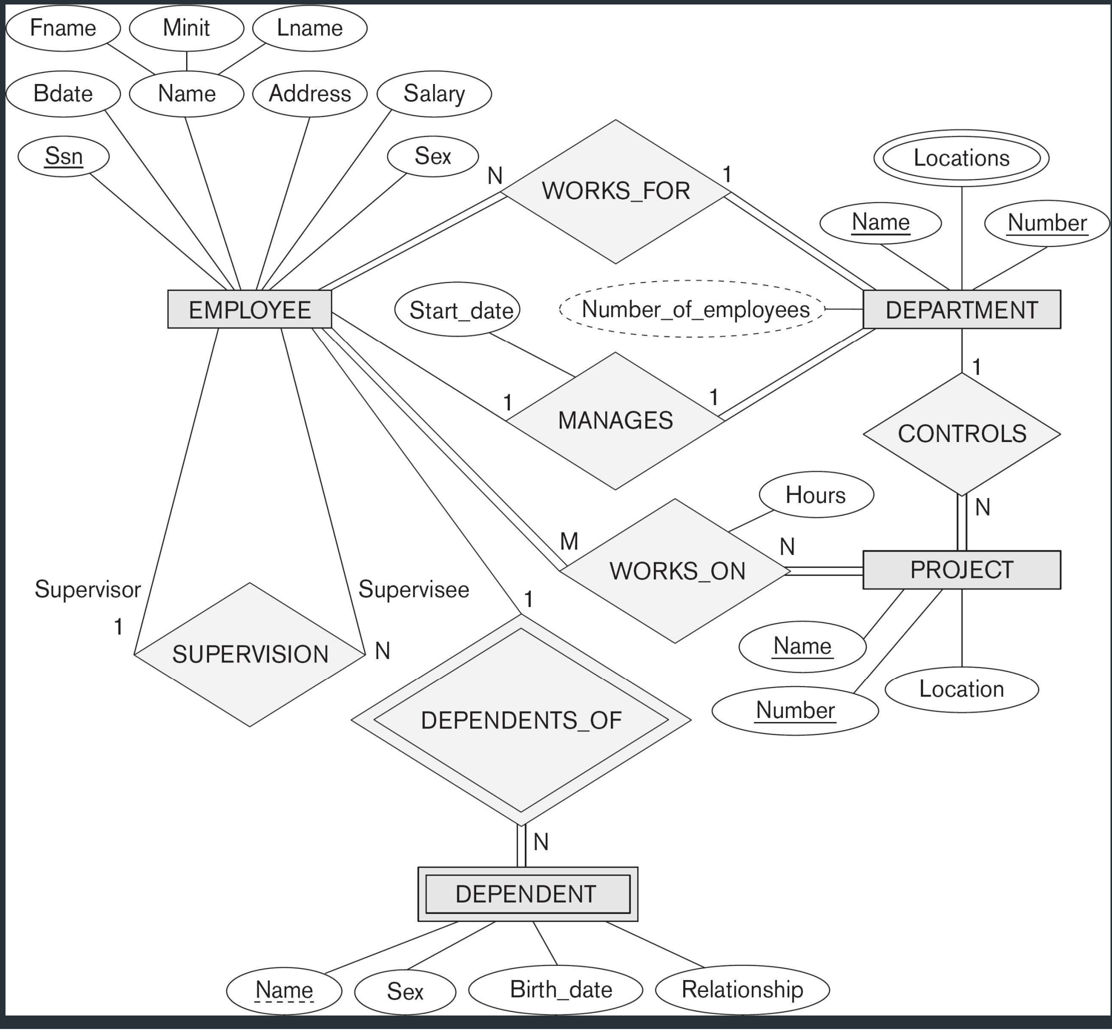
Step 5: Binary M-to-N

i. Create a **new** relation *S* (termed: *relationship relation*)

ii. Add as foreign keys the primary keys of both relations; their **combination** forms the primary key of *S*

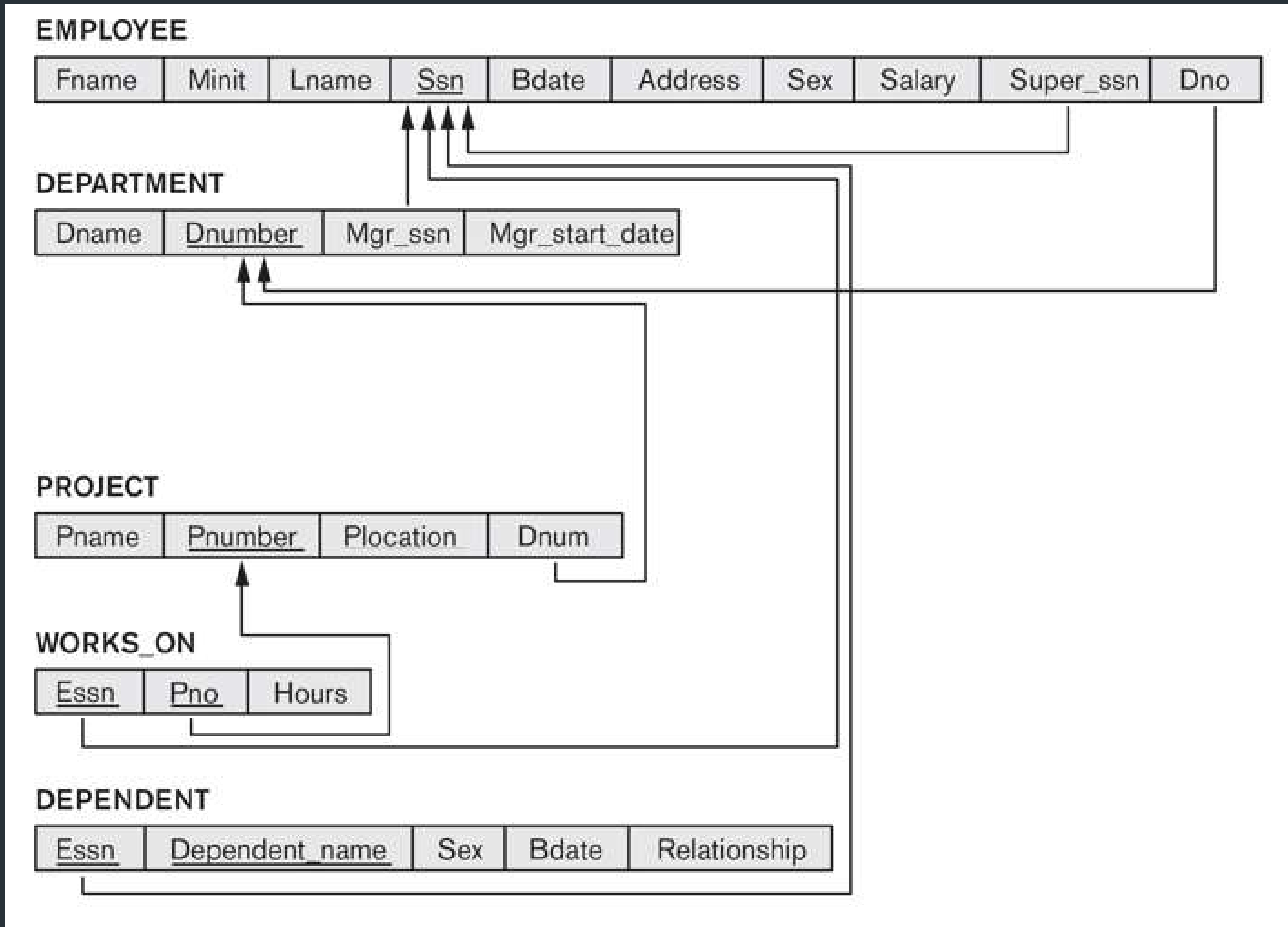
iii. Add any simple attributes of the M:N relationship to *S*

Example ERD



Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
DEPENDENTS_OF
Name
Location
Number
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Step 5 Result



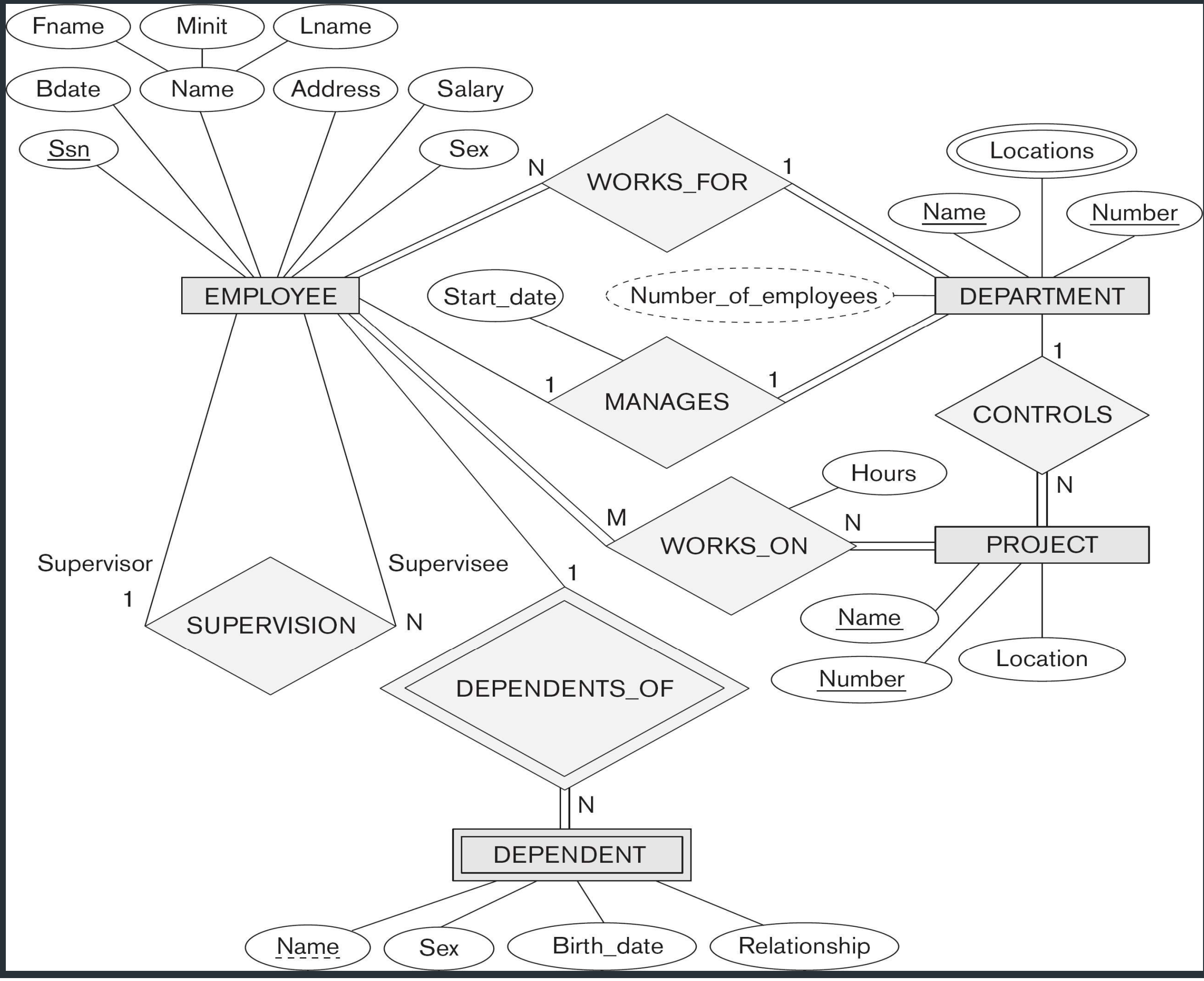
Step 6: Multivalued Attributes

i. Create a **new** relation *S*

ii. Add as foreign keys the primary keys of the corresponding relation

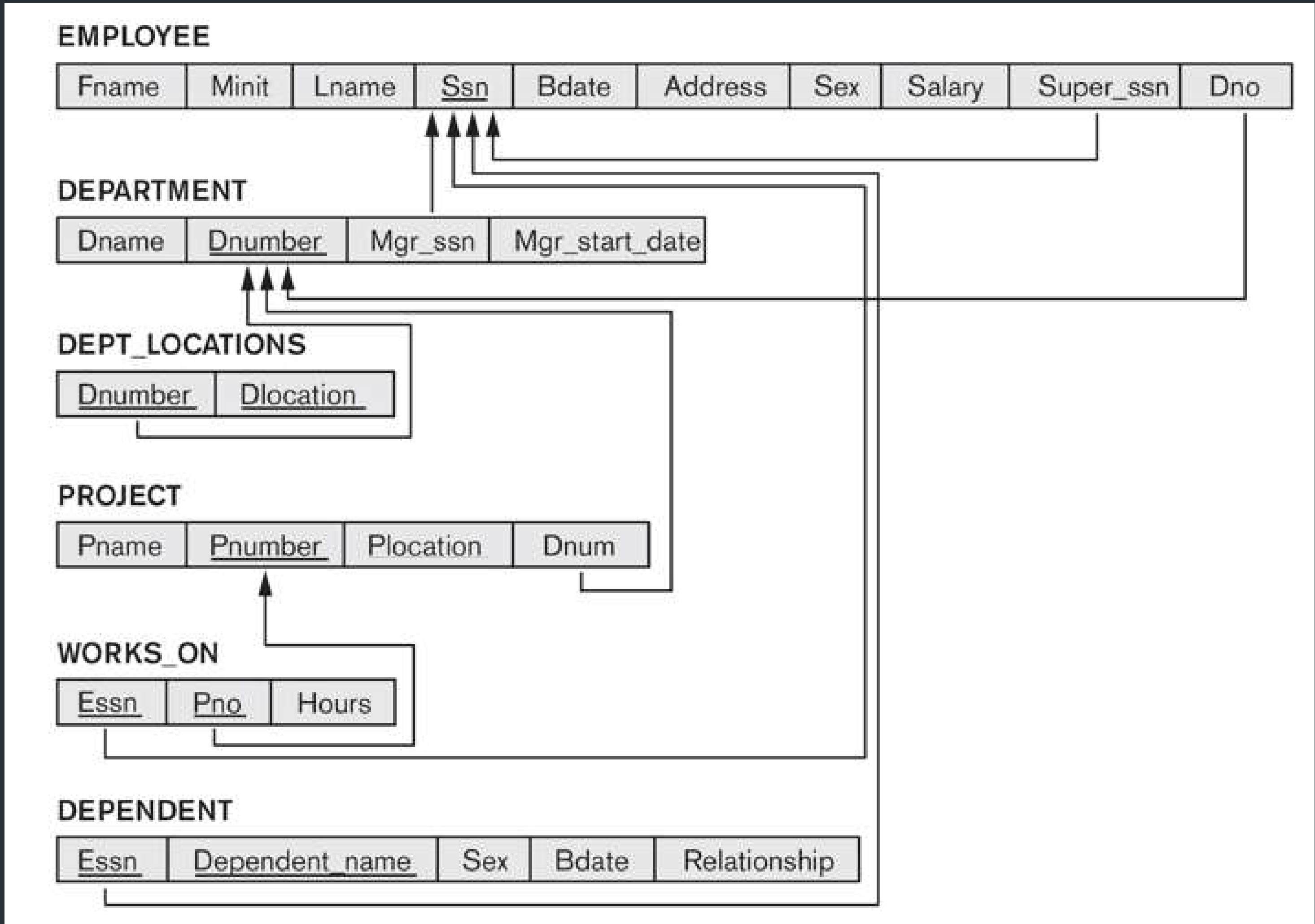
iii. Add the attribute to *S* (if composite, the simple attributes); the combination of all attributes in *S* forms the primary key

Example ERD



Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
DEPENDENTS_OF
Number
Location
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Step 6 Result




Thank you