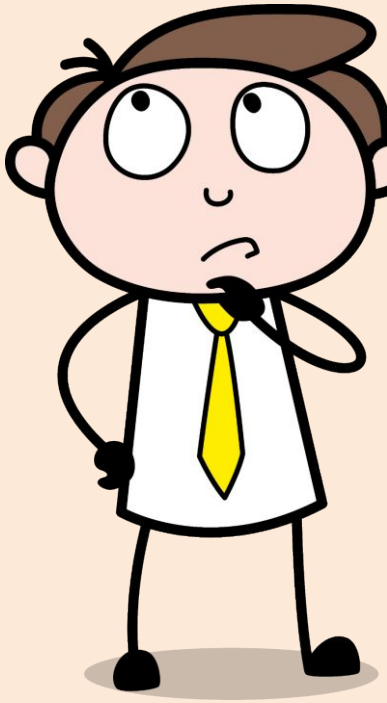
Module 1

- Concept & Overview of Database Management Systems (DBMS) - Characteristics of Database system, Database Users, structured, semi-structured and unstructured data. Data Models and Schema - Three Schema architecture. Database Languages, Database architectures and classification.

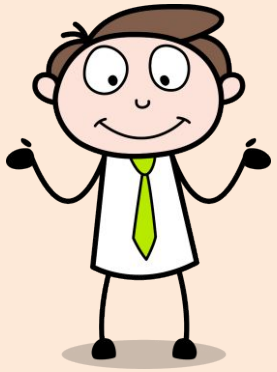
- ER model - Basic concepts, entity set & attributes, notations, Relationships and constraints, cardinality, participation, notations, weak entities, relationships of degree 3.

Data Versus Information



What is the difference between data and information??

Data: Collected facts about a topic or item



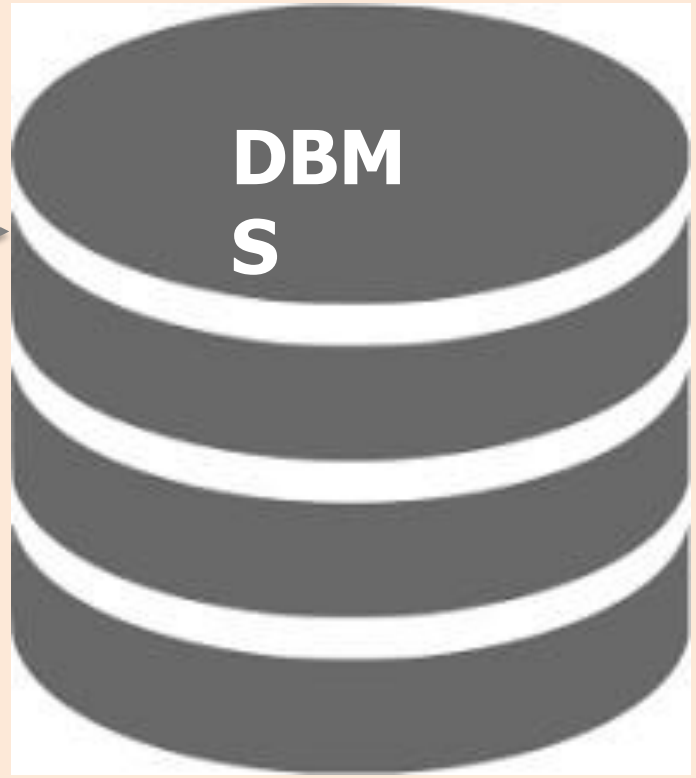
Information: The result of combining, comparing, and performing calculations on data

Data Versus Information an Example

Data in…

2021
$1,000,000
2022
$2,000,000

Student test scores were …



Information out …

… Next year's budget

… Class average or school average

DBMS Definition

- **Data Base Management System(DBMS)**
  - Is a centralized and structured set of data stored on a computer system
  - Provides facilities forretrieving, adding, modifying, and deleting the data when required
  - Provides facilities for transforming retrieved data into useful information
- A DBMS is software that controls the storage, organization, and retrieval of data

Basic Definitions

- **Database:**
  A collection of related data.
- **Data:**
  Known facts that can be recorded and have an implicit meaning.
- **Mini-world:**
  Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Data Base Management System:**
  A DBMS is software that controls the storage, organization, and retrieval of data
- **Database System:**
  The DBMS software together with the data itself. Sometimes, the applications are also included.

A simplified architecture for a database system

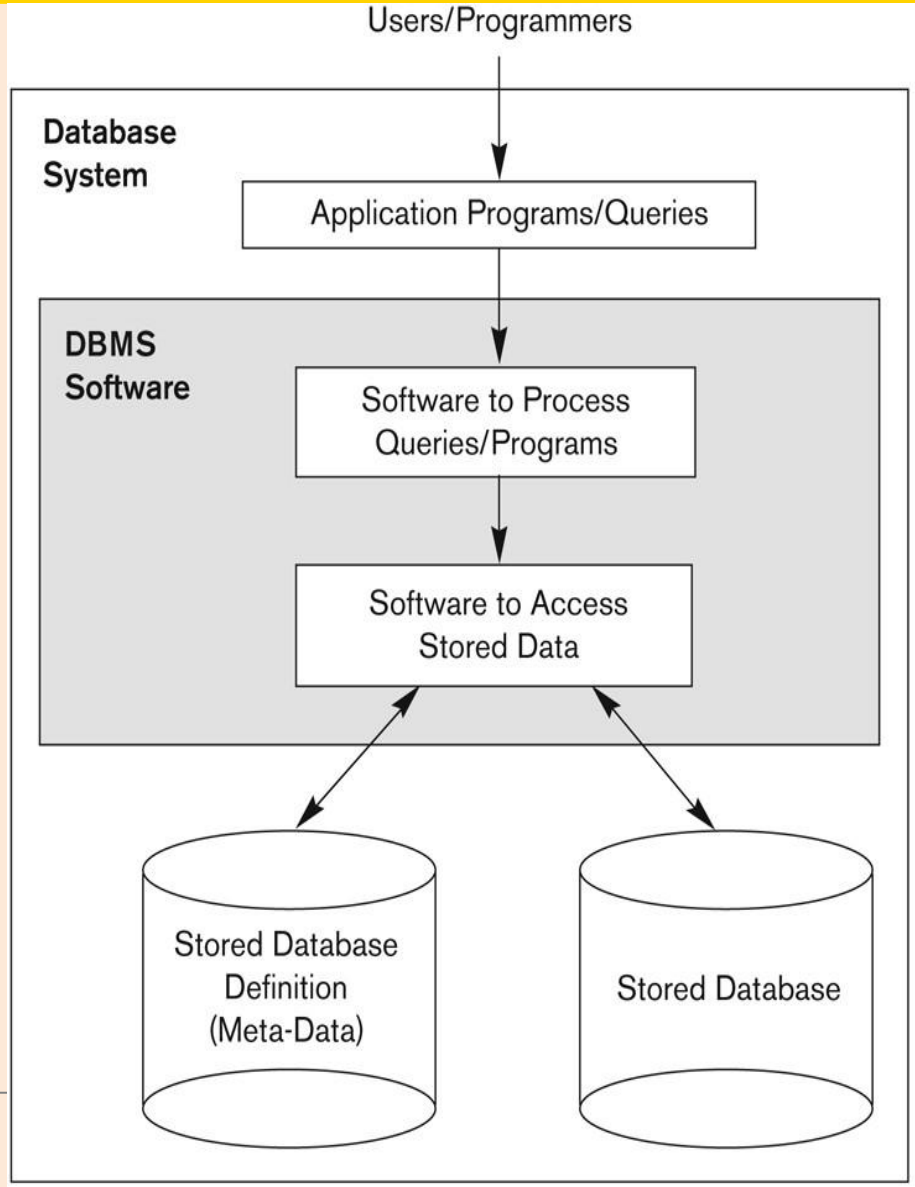


Figure 1.1
A simplified database system environment.

Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

What a DBMS Facilitates

- Define a particular database in terms of its data types, structures, and constraints
- Construct or load the initial database contents on a secondary storage medium
- Manipulating the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- Processing and sharing by a set of concurrent users and application programs , keeping all data valid and consistent

Other DBMS Functionalities

- DBMS may additionally provide:
  - **Protection** or security measures to prevent unauthorized access
  - **“Active” processing** to take internal actions on data
  - Presentation and visualization of data
  - **Maintenance** of the database and associated programs over the lifetime of the database application

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.

Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

Application Programs and DBMS

- Applications interact with a database by generating
  - Queries: that access different parts of data and formulate the result of a request
  - Transactions: that may read some data and “update” certain values or generate new data and store that in the database

CST 204 - Database Management Systems

Characteristics of Database system

Characteristics of the Database Approach

- Older approach – Programing with files
  - Grade reporting office, may keep files on students and their grades
  - Accounting office, may keep track of students fees and their payments
  - This redundancy in defining and storing results in memory wastage
- Database approach
  - Single repository is maintained and is accessed by different users
  - The main characteristics of database approach are
    - Self-describing nature of a database system
    - Insulation between programs and data, and Data abstraction
    - Support of multiple views of the data
    - Sharing of data and multi-user transaction processing

Characteristics of the 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.

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

Characteristics of the Database Approach

- **Insulation between programs and data:**
  - The structure of data files is stored in DBMS catalog separately from the access program (program-data independence).
  - Allows changing data structures and storage organization without having to change the DBMS access programs
    - E.g., ADTs

Data Item Name	Starting Position in Record	Length in Characters (bytes)
Name	1	30
Student_number	31	4
Class	35	1
Major	36	4

Figure 1.4
Internal storage format for a STUDENT record, based on the database catalog in Figure 1.3.

Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

Characteristics of the Database Approach (continued)

- 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
- 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.

Example of a multiple views of the data

TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

(a)

COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

(b)

Figure 1.5

a) User is interested in accessing and printing the transcript of each user

b) User who is interested only in checking that students have taken all the prerequisites of each course

Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

Characteristics of the Database Approach (continued)

- 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; allows hundreds of concurrent transactions to execute per second.

CST 204 - Database Management Systems

Database Users

Database Users

- Users may be divided into
  - Those who actually use and control the database content, and those who design, develop and maintain database applications (called "**Actors on the Scene**"), and
  - Those who design and develop the DBMS software and related tools, and the computer systems operators (called "**Workers Behind the Scene**").

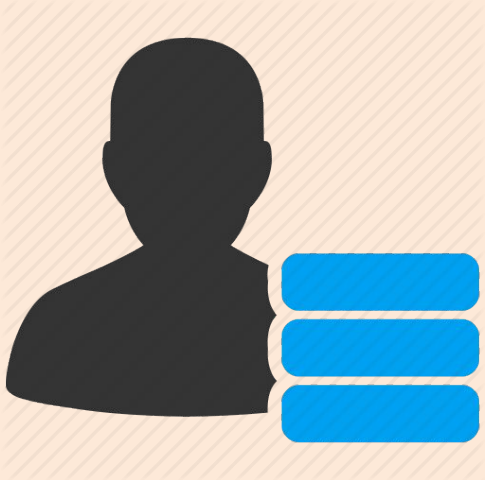
Database Users - Actors on the scene

- **Database administrators**
  - Responsible for authorising access to the database
  - for coordinating and monitoring its use
  - acquiring software and hardware resources
  - controlling its use and monitoring efficiency of operations.

- **Database designers**
  - Responsible to define the
    - content,
    - the structure,
    - the constraints,
    - and functions or transactions against the database.
  - They must communicate with the end-users and understand their needs.

Database Users - Actors on the scene

- **End-users**: They use the data forqueries, reports and some of them update the database content.
- End-users can be categorized into:
  - Casual: access database occasionally when needed
  - Naïve or parametric: they make up a large section of the end-user population.
    - They use previously well-defined functions in the form of "canned transactions" against the database.
      - Users of mobile apps mostly fall in this category
      - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
      - Social media users post and read information from websites



Database Users - Actors on the scene

- Sophisticated:
  - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
  - Many use tools in the form of software packages that work closely with the stored database.
- Stand-alone:
  - Mostly maintain personal databases using ready-to-use packaged applications.
  - An example is the user of a tax program that creates its own internal database.
  - Another example is a user that maintains a database of personal photos and videos.

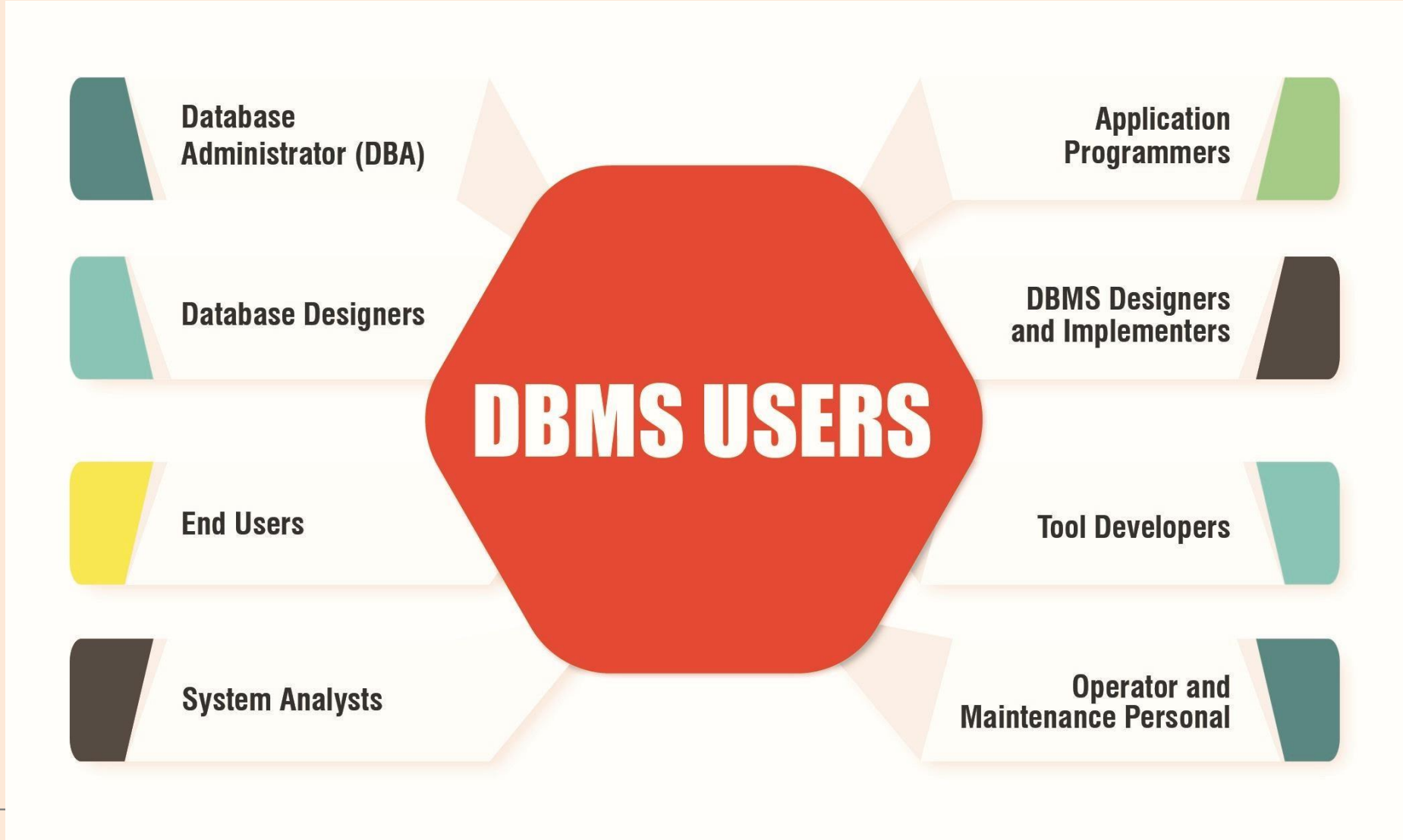
Database Users - Actors on the scene

- **System analysts and application developers**
  - System analysts: They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
  - Application programmers: Implement the specifications developed by analysts and test and debug them before deployment.
  - Business analysts: There is an increasing need for such people who can analyze vast amounts of business data and real-time data ("Big Data") for better decision making related to planning, advertising, marketing etc.

Database Users – Actors behind the Scene

- System designers and implementers:
  - Design and implement DBMS packages in the form of modules and interfaces and test and debug them.
  - The DBMS must interface with applications, language compilers, operating system components, etc.
- Tool developers:
  - Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- Operators and maintenance personnel:
  - They manage the actual running and maintenance of the database system hardware and software environment.

Database Users



CST 204 - Database Management Systems

Structured, semi-structured and unstructured data

Structured Data

- Structured data is generally tabular data that is represented by columns and rows in a database.
- Databases that hold tables in this form are called relational databases.
- The mathematical term “relation” specify to a formed set of data held as a table.
- In structured data, all row in a table has the same set of columns.
- SQL (Structured Query Language) programming language used for structured data.

id	name	age
1	Jim	28
2	Pam	26
3	Michael	42

id	subject	Teacher
1	Languages	John Jones
2	Track	Wally West
3	Swimming	Arthur Curry
4	Computers	Victor Stone

student_id	subject_id	grade
2	1	98
1	2	100
1	4	75
3	3	60
2	4	76
3	2	88

Semi-structured Data

- Semi-structured data is information that doesn't consist of Structured data (relational database) but still has some structure to it.

- Semi-structured data consist of documents held in JavaScript Object Notation (JSON) format.
- It also includes key-value stores and graph databases.

```
## Document 1 ##
{
  "customerID": "103248",
  "name":
  {
    "first": "AAA",
    "last": "BBB"
  },
  "address":
  {
    "street": "Main Street",
    "number": "101",
    "city": "Acity",
    "state": "NY"
  },
  "ccOnFile": "yes",
  "firstOrder": "02/28/2003"
}
```

Unstructured Data

- Unstructured data is information that either does notorganize in a pre- defined manner or not have a pre-defined data model.
- Unstructured information is a set of text-heavy but may contain data such as numbers, dates, and facts as well.
- Videos, audio, and binary data files might nothave a specific structure. They're assigned to as unstructured data.

CST 204 - Database Management Systems

Data Models

Data Models, Schemas, and Instances

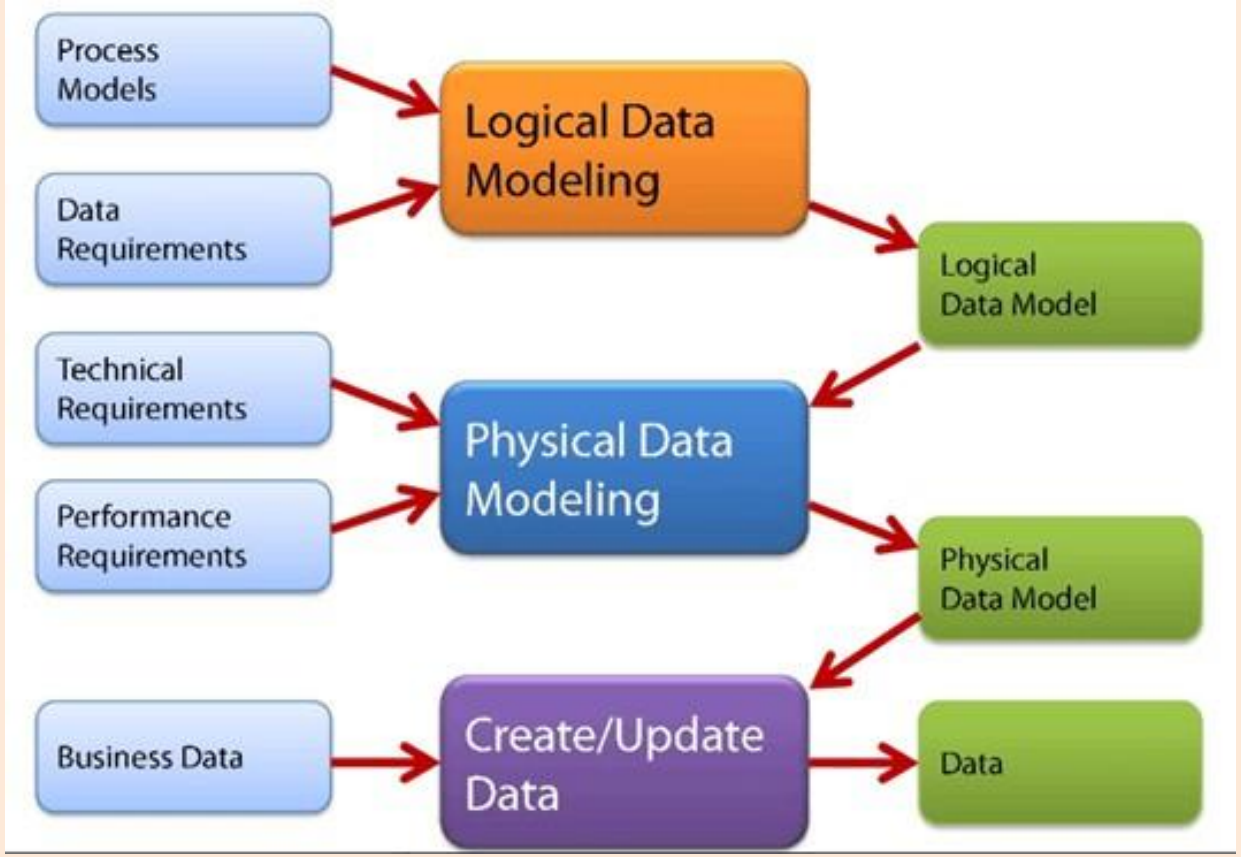
- One fundamental characteristic of the database approach is data abstraction: hiding details of data storage that are irrelevant to database users.
- A data model - a collection of concepts that can be used to describe the conceptual/logical structure of a database
  - Provides the necessary means to achieve this abstraction.
  - Structure means the data types, relationships, and constraints that should hold for the data.
  - Most data models also include a set of basic operations for specifying retrievals/updates.
  - The Data Model gives us an idea of how the final system will look after its complete implementation.

Why Data Model?

- The primary goals of using the data model are:
  - Ensures that all data objects required by the database are accurately represented.
    - The omission of data will lead to the creation of faulty reports and produce incorrect results.
  - A data model helps design the database at the conceptual, physical, and logical levels.
  - Data Model structure helps to define the relational tables, primary and foreign keys, and stored procedures.
  - It provides a clear picture of the base data and can be used by database developers to create a physical database.
  - It is also helpful to identify missing and redundant data.
  - Though the initial creation of a data model is labor and time-consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

Types of Data Models

- There are mainly three different types of data models:
  - Conceptual data models
  - Logical data models
  - Physical data models



Source: https://www.guru99.com/data-modelling-conceptual-logical.html

Types of Data Models

- Conceptual Data Model:
  - This Data Model defines WHAT the system contains.
  - This model is typically created by Business stakeholders and Data Architects.
  - The purpose is to organize, scope and define business concepts and rules.
  - It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database.
    - One such popular model is the entity/relationship model (ER model). The E/R model specializes in entities, relationships, and even attributes that are used by database designers.

Source: https://www.guru99.com/data-modelling-conceptual-logical.html

Types of Data Models

- Logical Data Model:
  - Defines HOW the system should be implemented regardless of the DBMS.
  - This model is typically created by Data Architects and Business Analysts.
  - The purpose is to develop a technical map of rules and data structures.
  - A popular representational model is a Relational model. The relational Model consists of Relational Algebra and Relational Calculus.

Source: https://www.guru99.com/data-modelling-conceptual-logical.html

Types of Data Models

- Physical Data Model:
  - This Data Model describes HOW the system will be implemented using a specific DBMS system.
  - This model is typically created by DBA and developers.
  - The purpose is the actual implementation of the database.
  - Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures.
  - Structured Query Language (SQL) is used to practically implement Relational Algebra.

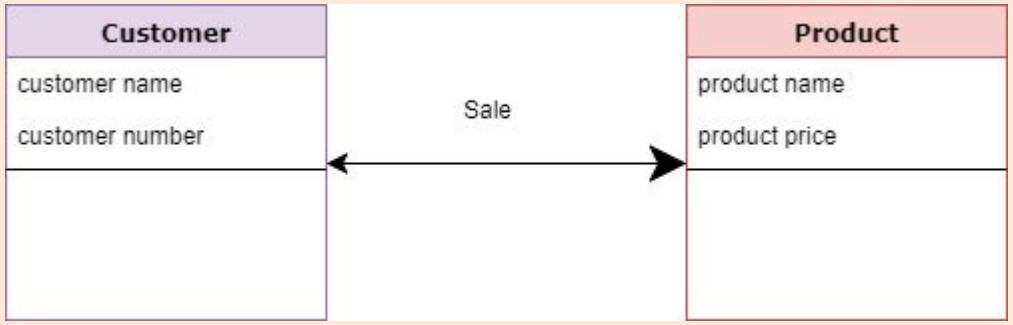
Source: https://www.guru99.com/data-modelling-conceptual-logical.html

Conceptual Data Model

- A Conceptual Data Model is an organised view of database concepts and their relationships.
- The purpose of creating a conceptual data model is to establish entities, their attributes, and relationships.
  - Entity: A real-world thing
  - Attribute: Characteristics or properties of an entity
  - Relationship: Dependency or association between two entities

Data model example:

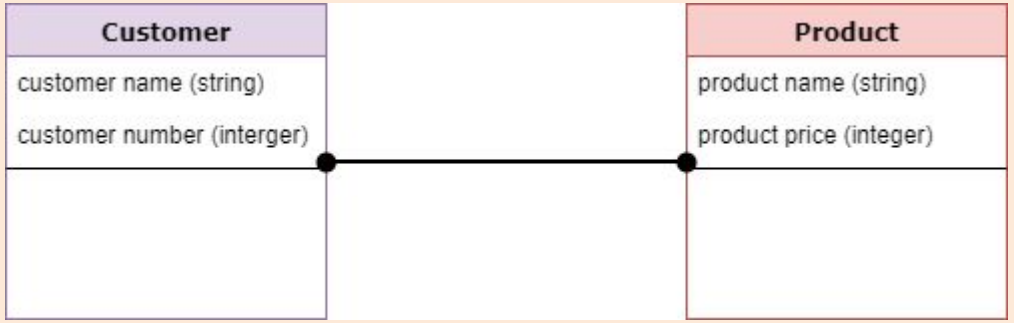
- Customer and Product are two entities. Customer number and name are attributes of the Customer entity
- Product name and price are attributes of product entity
- Sale is the relationship between the customer and product



Source: https://www.guru99.com/data-modelling-conceptual-logical.html

Logical Data Model

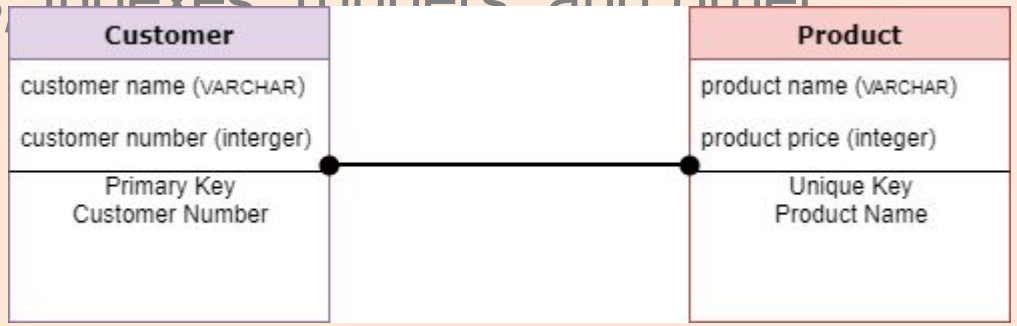
- The Logical Data Model is used to define the structure of data elements and to set relationships between them.
- The logical data model adds further information to the conceptual data model elements.
- The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model.
- However, the modeling structure remains generic.



Source: https://www.guru99.com/data-modelling-conceptual-logical.html

Physical Data Model

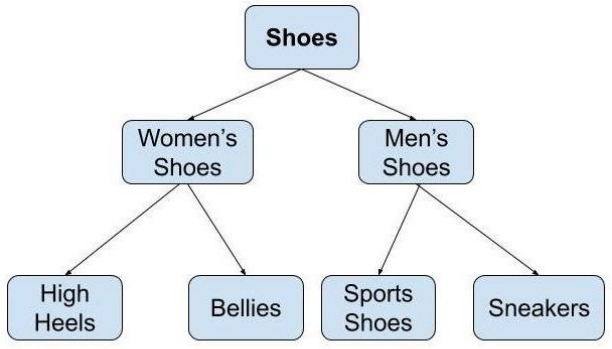
- A Physical Data Model describes a database-specific implementation of the data model.
- It offers database abstraction and helps generate the schema.
- Thisis because of therichness of meta-data offered by a Physical Data
  Model.
- The physical data model also helps in visualizing database structure by replicating database column keys, constraints, indexes, triggers, and other RDBMS features.



Source: https://www.guru99.com/data-modelling-conceptual-logical.html

Data Model

- **Hierarchical Model**
- The Hierarchical Model was the first DBMS model. This model organises the data in the hierarchical tree structure.
- A Hierarchical database model was widely used during the Mainframe Computers Era.
- It is used in applications where high performance is required such as telecommunications and banking.
- A hierarchical database is also used for Windows Registry in the Microsoft Windows operating system.
- It is useful where the following two conditions are met:
  - The data should be in a hierarchical pattern i.e. parent-child relationship must be present.
  - The data in a hierarchical pattern must be accessed through a single path only.

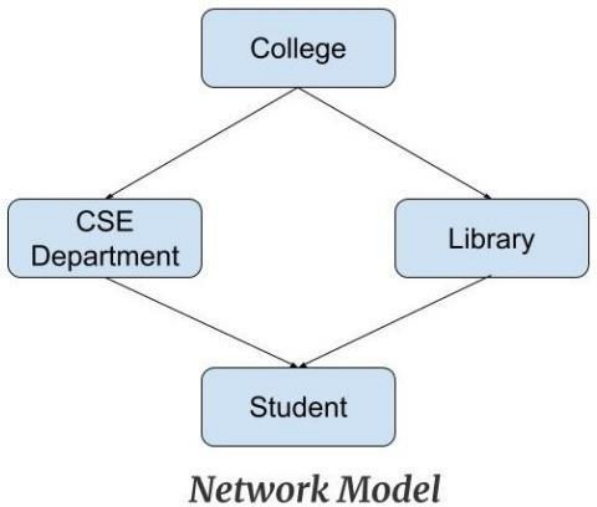


Hierarchical Model

Source: https://afteracademy.com/blog/what-is-data-model-in-dbms-and-what-are-its-types

Network Model

- This model is an extension of the hierarchical model.
- It was the most popular model before the relational model.
- This model is the same as the hierarchical model, the only difference is that a record can have more than one parent.



Source: https://afteracademy.com/blog/what-is-data-model-in-dbms-and-what-are-its-types

Data Model

Relational Model

- Relational Model is the most widely used model.
- In this model, the data is maintained in the form of a two-dimensional table.
- All the information is stored in the form of row and columns.

Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

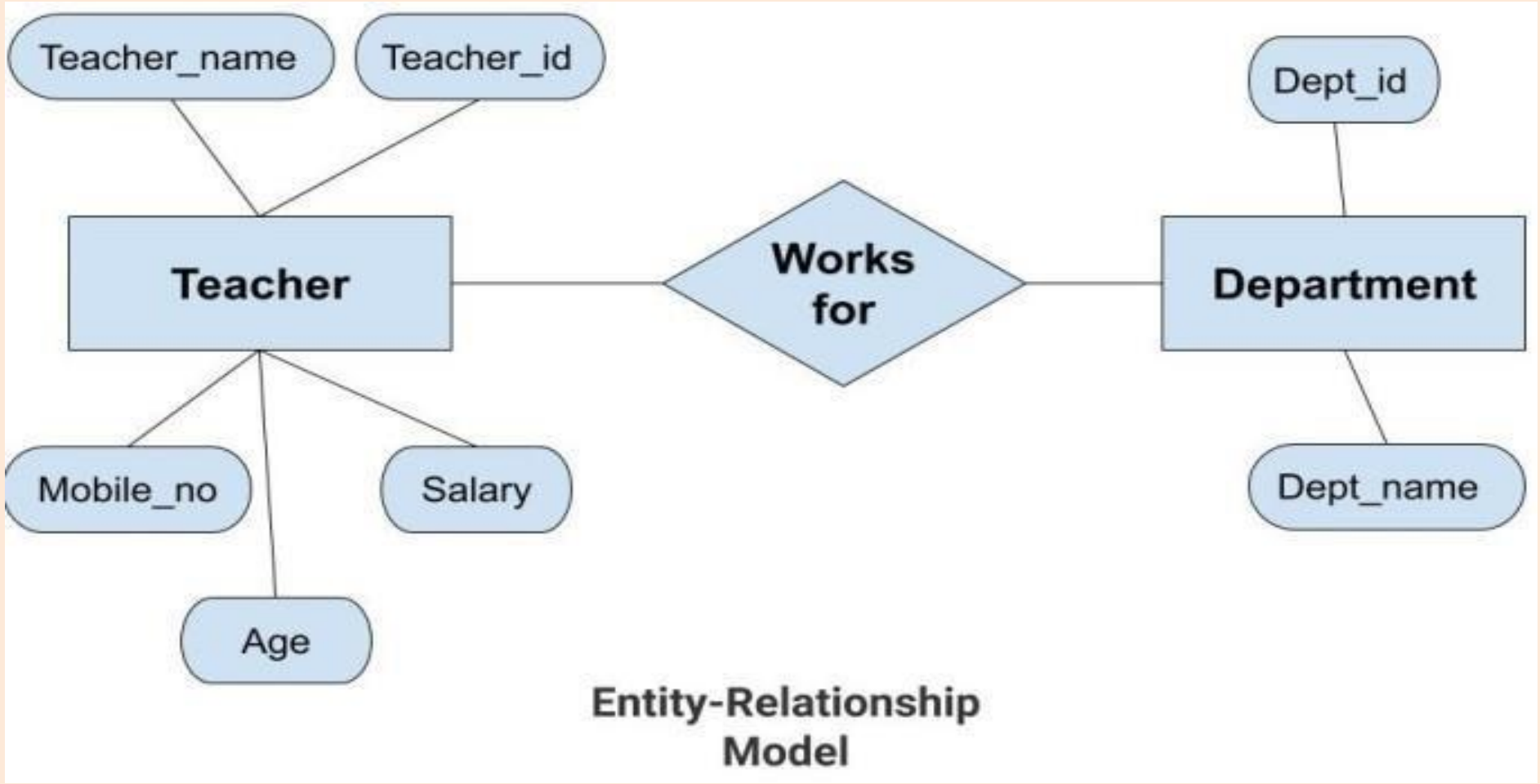
Entity-Relationship Model

- Entity-Relationship Model or simply ER Model is a high-level data model diagram.
- In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand.
- It is also very easy for the developers to understand the system by just looking at the ER diagram.
- We use the ER diagram as a visual tool to represent an ER Model.

Entity-Relationship Model

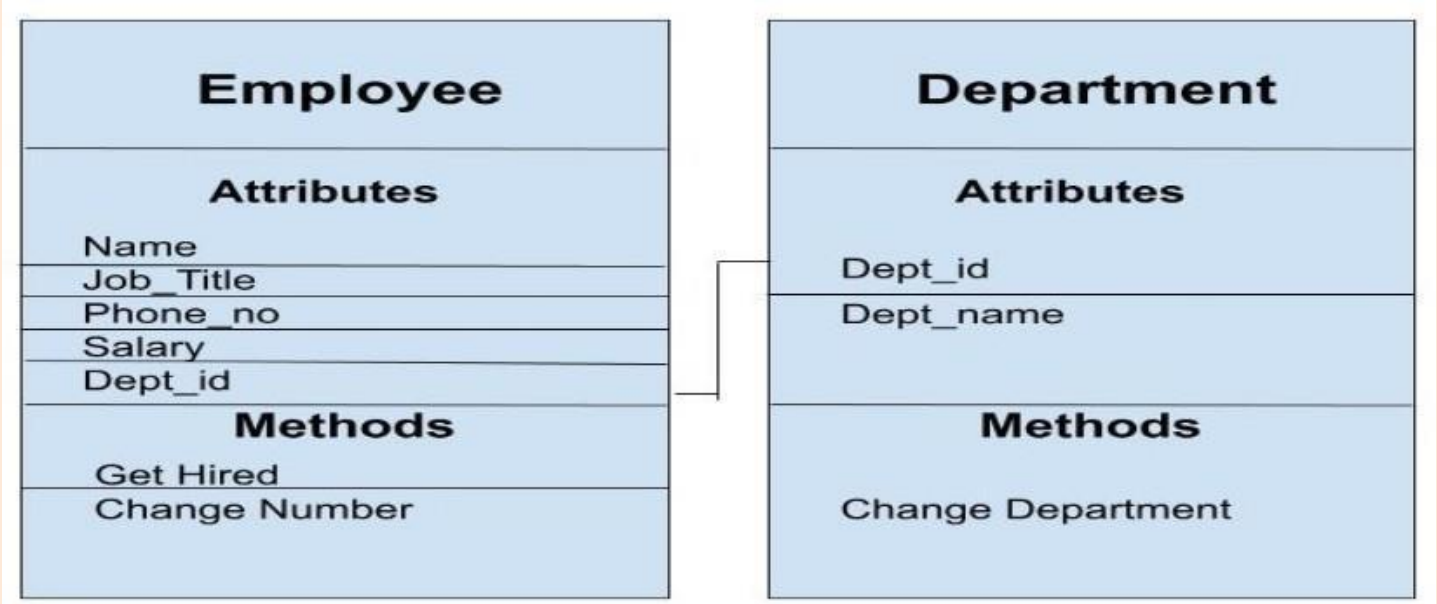
- ER diagram has the following three components:
  - **Entities**: Entity is a real-world thing. It can be a person, place, or even a concept.
    - Example: Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.
  - **Attributes**: An entity contains a real-world property called attribute. This is the characteristics of that attribute.
    - Example: The entity teacher has the property like teacher id, salary, age, etc.
  - **Relationship**: Relationship tells how two attributes are related.
    - Example: Teacher works for a department.

Data Model


Teacher_name
Teacher_id
Dept_id
Teacher
Works for
Department
Mobile_no
Salary
Dept_name
Age
Entity-Relationship Model

Object-Oriented Data Model

- The real-world problems are more closely represented through the object-oriented data model.
- In this model,both thedata and relationship arepresent in a single structure known as an object.



Object_Oriented_Model

CST 204 - Database Management Systems

Schema - Three Schema architecture

Schemas

- Database Schema
  - The description of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- Schema Diagram:
  - An illustrative display of (most aspects of) a database schema.
- Schema Construct:
  - A component of the schema or an object within the schema, e.g., STUDENT, COURSE.

Schema - Example

STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

PREREQUISITE

CourseNumber	PrerequisiteNumber

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade

Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

Instances

- Database State:
  - The actual data stored in a database at a particular moment in time. This includes the collection of all the data in the database.
  - Also called database instance (or occurrence or snapshot).

Database Schema vs. Database State

- Database State:
  - Refers to the content of a database at a moment in time.
- 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.

Database Schema vs. Database State

- Distinction
  - The database schema changes very infrequently.
  - The database state changes every time the database is updated.

- Schema is also called intension.
- State is also called extension.

Example of a database state

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.

Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

CST 204 - Database Management Systems

Three-Schema Architecture

Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - Program-data independence.
  - Support of multiple views of the data.

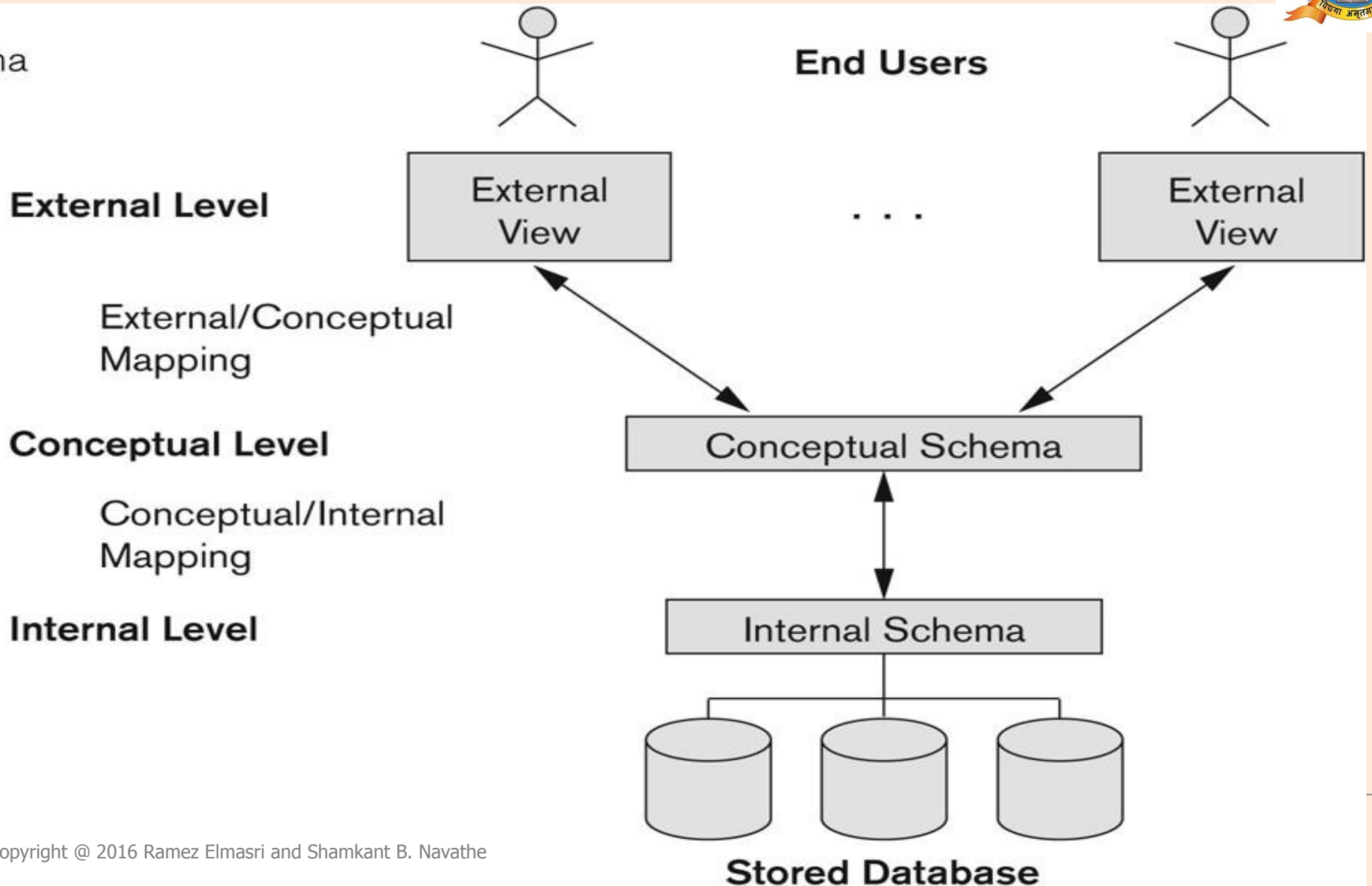
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization

Three-Schema Architecture

- Defines DBMS schemas at three levels:
  - Internal schema at the internal level to describe physical storage structures and access paths (e.g indexes).
    - Typically uses a physical data model.
  - Conceptual schema at the conceptual level to describe the structure and constraints for the whole database for a community of users.
    - Uses a conceptual or an implementation data model.
  - External schemas at the external level to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

Three-Schema Architecture

Figure 2.2
The three-schema architecture.



Copyright @ 2016 Ramez Elmasri and Shamkant B. Navathe

Three-Schema Architecture

- Mappings among schemalevels are needed to transform requests and data.
  - Programs refer to an external schema and are mapped by the DBMS to the internal schema for execution.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view.

Three-Schema Architecture – Data Independence

- **Logical Data Independence:**
  - The capacity **to change the conceptual schema** without having to change the external schemas and their associated application programs.

TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

(a)

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

GRADE_REPORT

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Smith	112	MATH2410	B
17	Smith	119	CS1310	C
8	Brown	85	MATH2410	A
8	Brown	92	CS1310	A
8	Brown	102	CS3320	B
8	Brown	135	CS3380	A

Three-Schema Architecture – Data Independence

- **Physical Data Independence:**
  - The capacity to change the internal schema without having to change the conceptual schema.
  - For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance

CST 204 - Database Management Systems

Database Languages

Database Languages

- Data Definition Language (DDL)
  - The Data Definition Language is used for specifying the database schema
- Data Manipulation Language (DML)
  - Data Manipulation Language is used for both reading and updating the database

Database Languages

- Data Definition Language (DDL)
  - Data Definition Language (DDL) statements are used to classify the database structure or schema.
  - It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints.
  - Here are the lists of tasks that come under DDL:
    - CREATE - used to create objects in the database
    - ALTER - used to alter the structure of the database
    - DROP - used to delete objects from the database
    - TRUNCATE - used to remove all records from a table, including all spaces allocated for the records are removed
    - RENAME - used to rename an object

Database Languages

- Data Manipulation Language (DML)
  - A language that offers a set of operations to support the fundamental data manipulation operations on the data held in the database.
  - Data Manipulation Language (DML) statements are used to manage data within schema objects.
  - Here are the lists of tasks that come under DML:
    - SELECT - It retrieves data from a database
    - INSERT - It inserts data into a table
    - UPDATE - It updates existing data within a table
    - DELETE - It deletes all records from a table, the space for the records remain

Database Languages

- Data Control Language (DCL)
  - DCL (Data Control Language) includes commands like GRANT and REVOKE, which are
    useful to give "rights & permissions."
  - Using DCL command, it allows or restricts the user from accessing data in database schema.
  - Commands that come under DCL:
    - Grant: This command is use to give user access privileges to a database
    - Syntax: GRANT SELECT, UPDATE ON TABLE TO SOME_USER, ANOTHER_USER
    - For example: GRANT SELECT ON Users TO 'Tom'@'localhost';
    - Revoke: It is useful to back permissions from the user.
    - Syntax:
    - REVOKE privilege_name ON object_name FROM {user_name |PUBLIC |role_name}
    - For example: REVOKE SELECT, UPDATE ON student FROM BCA, MCA;

Database Languages

- Transaction Control Language (TCL)
  - Transaction Control statements are used to run the changes made by DML statements.
  - It allows statements to be grouped into logical transactions.
  - Commands that come under DCL:
    - COMMIT - It saves the work done
    - SAVEPOINT - It identifies a point in a transaction to which you can later roll back
    - ROLLBACK - It restores the database to original since the last COMMIT
    - SET TRANSACTION - It changes the transaction options like isolation level and what rollback segment to use

Database Languages

- Storage Definition Language (SDL)
  - 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.
  - The mappings between the two schemas may be specified in either one of these languages.
  - In most relational DBMSs today, there is no specific language that performs the role of SDL.
  - Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to the storage of files.
  - These permit the DBA staff to control indexing choices and mapping of data to storage.

Database Languages

- View Definition Language (VDL)
  - To specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.
  - In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries
    - Taking an example CREATE TABLE f (x int, y varchar(25));. Here, you're defining
    - SDL: the storage for the row int, varchar(25).
    - DDL: the schema: f(x,y) a relational tuple with two slots.
    - VDL: the "view" into it, which is just a relationship by the same name as the table.

CST 204 - Database Management Systems

DBMS Components

Typical DBMS Component Modules

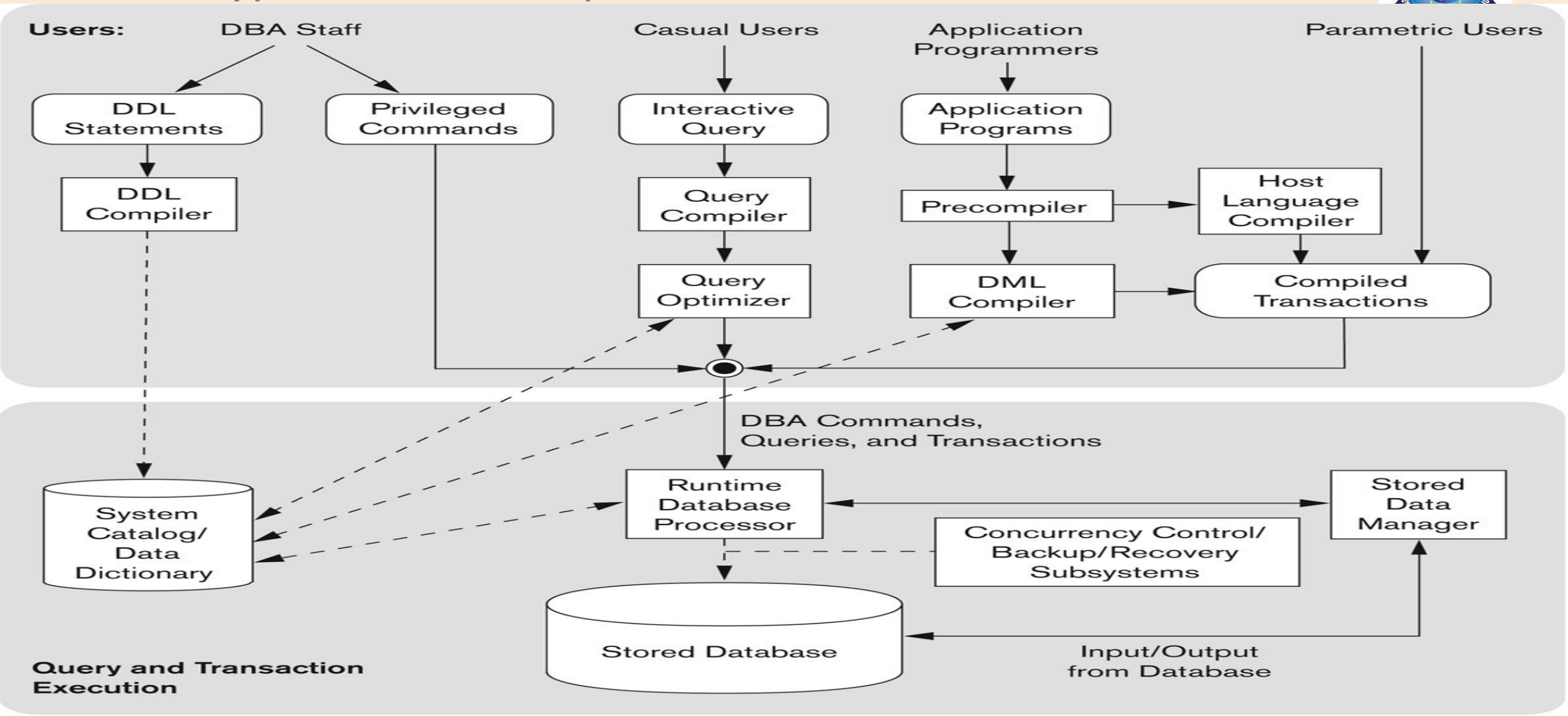


Figure 2.3
Component modules of a DBMS and their interactions.

DBMS component modules

- The database and the DBMS catalog are usually stored on disk.
  - So access is controlled primarily by the operating system (OS)
- A higher-level **stored data manager** module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.
- The **stored data manager**
  - Use basic OS services for carrying out low-level data transfer between the disk and computer's main storage,
  - it controls other aspects of data transfer, like handling buffers in main memory.
- Once the data is in main memory buffers, it can be processed by other DBMS modules, as well as by application programs.

DDL compiler

- processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.
- The catalog includes
  - names of files, data items, storage details of each file, mapping information among schemas, and constraints etc.
- DBMS software modules then look up the catalog information as needed.

DBMS component modules

The run-time database processor

- handles database accesses at run time
- it receives retrieval or update operations and carries them out on the database.
- Access to disk goes through the stored data manager.

Query compiler

- handles high-level queries that are entered interactively.
- It parses, analyzes, and compiles or interprets a query by creating database access code, and then generates calls to the run-time processor for executing the code.

DBMS component modules

Pre-compiler

- extracts DML commands from an application program written in a host programming language.
- These commands are sent to the **DML compiler** for compilation into object code for database access.
- The rest of the program is sent to the host language compiler.
- The object codes for the DML commands and the rest of the program are linked, forming a canned transaction whose executable code includes calls to the runtime database processor.

DBMS component modules

- The DBMS interacts with the operating system when disk accesses—to the database or the catalog—are needed.
- If the computer system is shared by many users,
  - the OS will schedule DBMS disk access requests and DBMS processing along with other processes.
- The DBMS also interfaces with compilers for general-purpose host programming languages.

CST 204 - Database Management Systems

Database architectures and classification

Centralized and Client-Server DBMS Architectures

Centralized DBMS:

- Combines everything into a single system including- DBMS software, hardware, application programs, and user interface processing software.
- User can still connect through a remote terminal – however, all processing is done at a centralized site.

A Physical Centralized Architecture

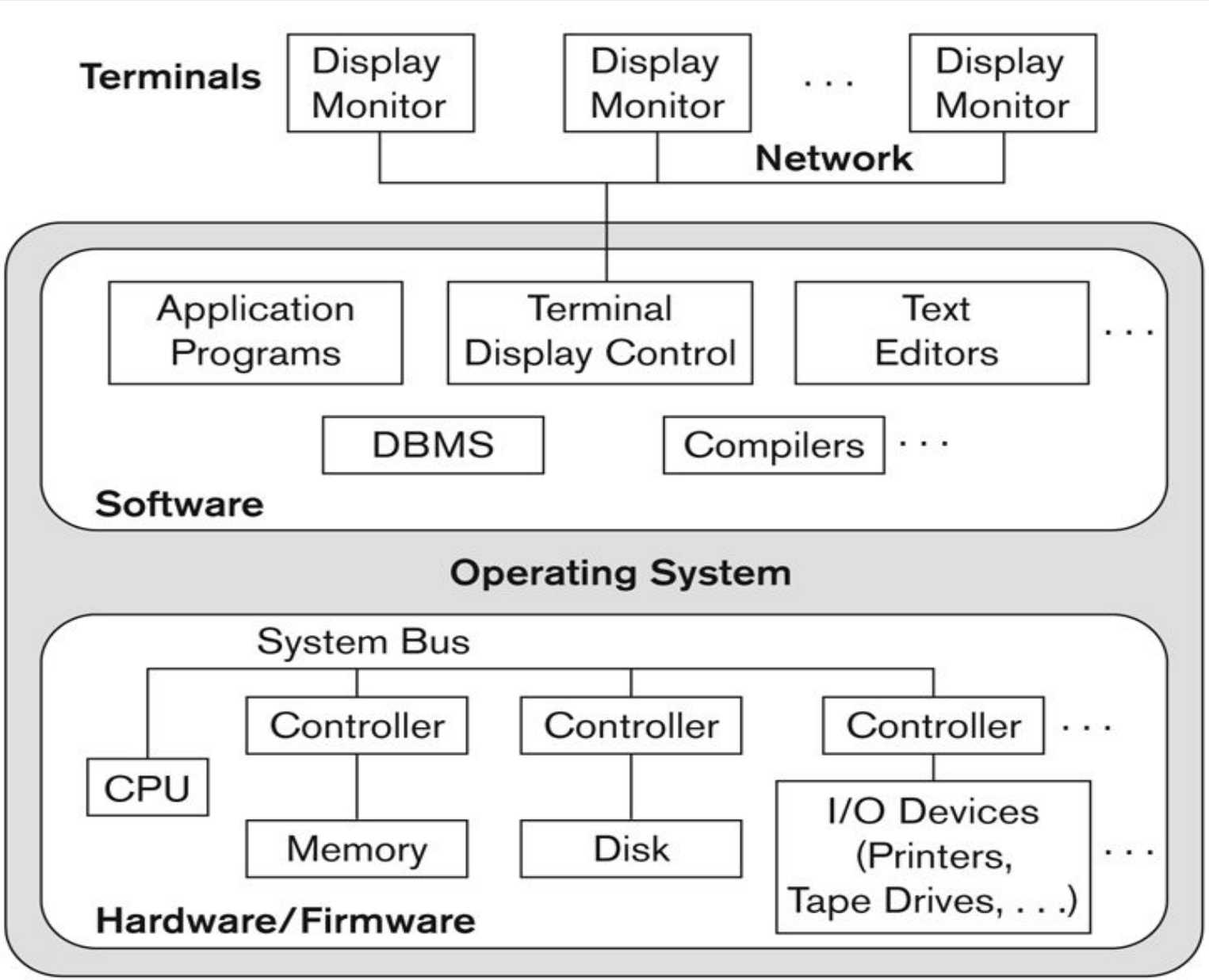


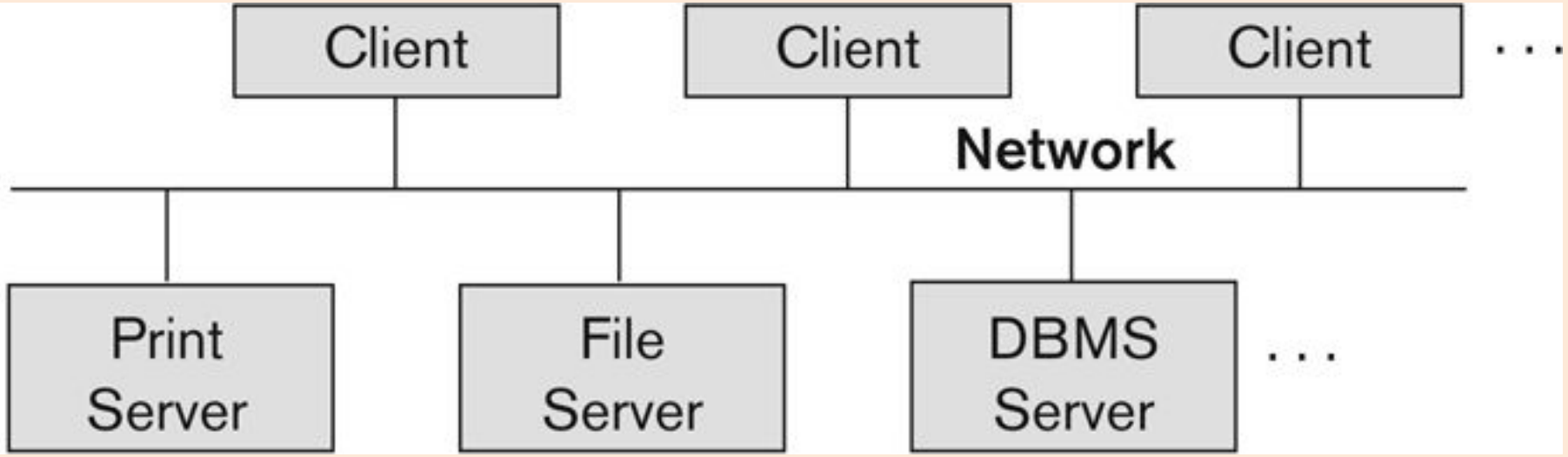
Figure 2.4
A physical centralized architecture.

Basic 2-tier Client-Server Architectures

- Specialized Servers with Specialized functions
  - Print server
  - File server
  - DBMS server
  - Web server
  - Email server
- Clients can access the specialized servers as needed

Logical two-tier client server architecture

Figure 2.5
Logical two-tier client/server architecture.



Clients

- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network.
  - (LAN: local area network, wireless network, etc.)

DBMS Server

- Provides database query and transaction services to the clients
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as:
  - ODBC: Open Database Connectivity standard
  - JDBC: for Java programming access
- Client and server must install appropriate client module and server module software for ODBC or JDBC

Two Tier Client-Server Architecture

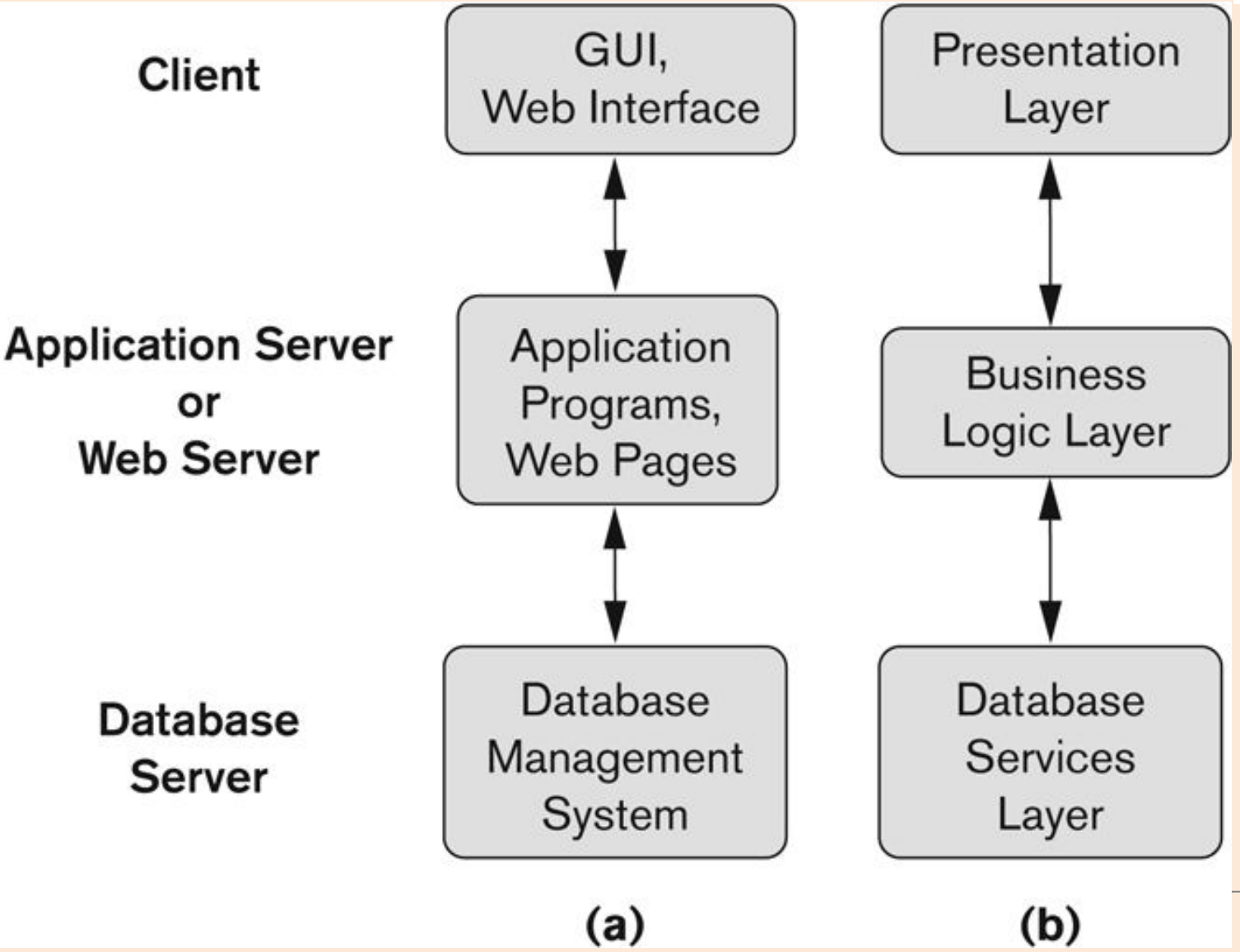
- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data.
- Other variations of clients are possible: e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc.

Three Tier Client-Server Architecture

- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
  - Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
  - Acts like a channel for sending partially processed data between the database server and the client.
- Three-tier Architecture Can Enhance Security:
  - Database server only accessible via a middle-tier
  - Clients cannot directly access the database server

Three-tier client-server architecture

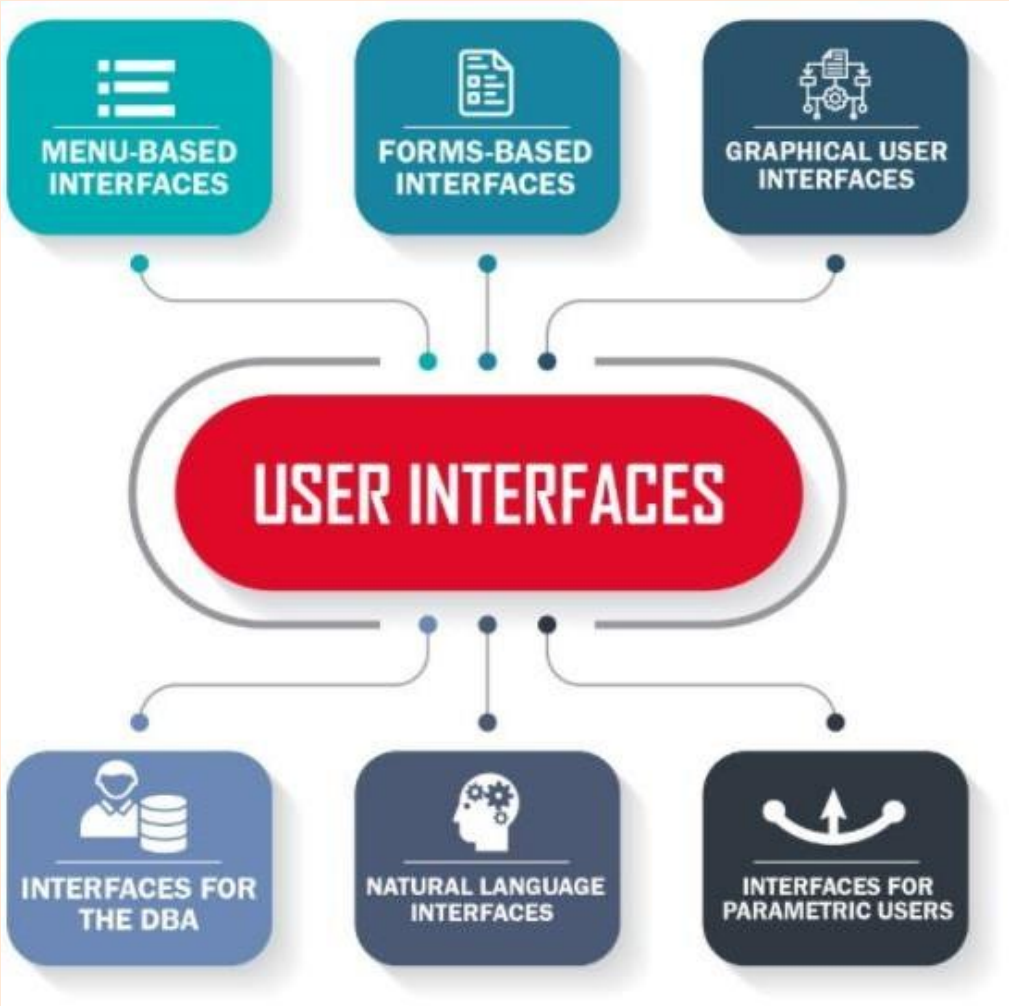
Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



CST 204 - Database Management Systems

DBMS Interfaces

DBMS Interfaces



Source: https://tutorials.ducatindia.com/

DBMS Interfaces

- A database management system (DBMS) interface is a user interface that allows for the ability to input queries to a database without using the query language itself.
  - Menu-Based Interfaces for Web Clients or Browsing
  - Forms-Based Interfaces
  - Graphical User Interface
  - Natural language Interfaces
  - Speech Input and Output
  - Interfaces for DBA

CST 204 - Database Management Systems

ER Diagram: Entity Relationship Diagram Model

What is ER Diagram?

- ER Diagram stands for Entity Relationship Diagram, also known as ER Diagram
- Diagram that displays the relationship of entity sets stored in a database.
- ER diagrams help to explain the logical structure of databases.
- ER diagrams are created based on three basic concepts:

entities, attributes, and relationships.

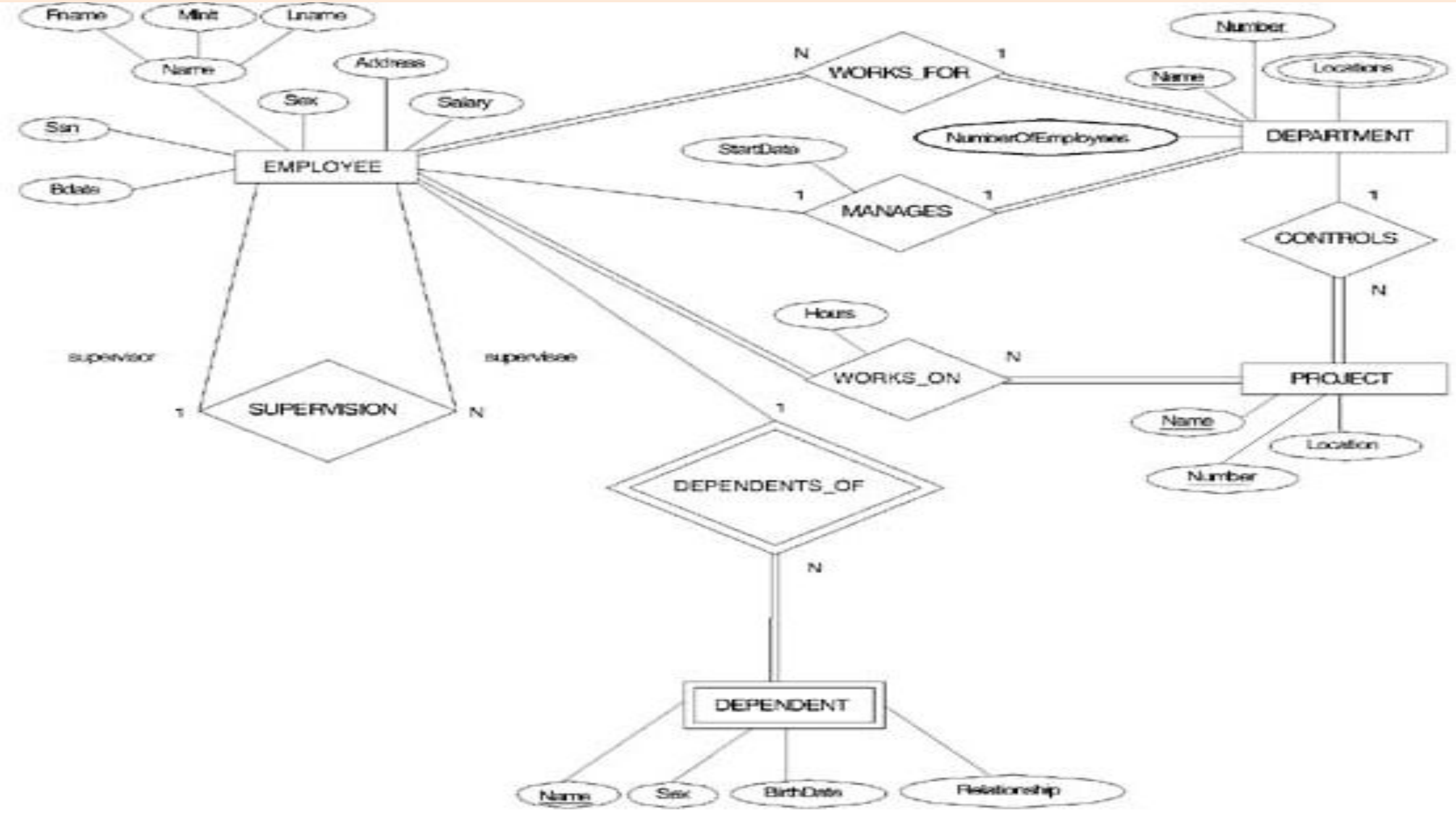
Example COMPANY Database

Requirements of the Company:

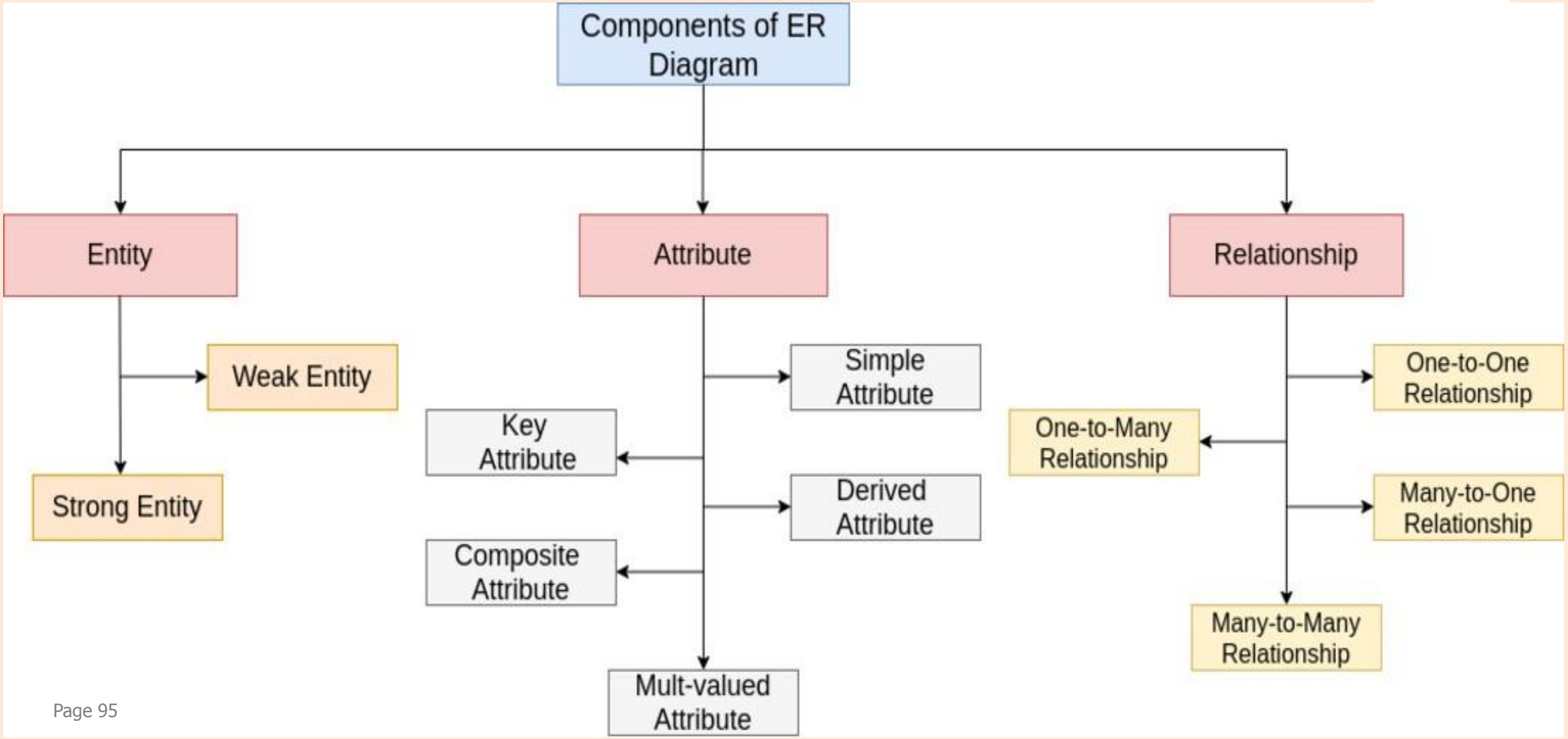
- The company is organized into DEPARTMENTs.
  - Each department has a name, number and an employee who *manages* the department.
  - We keep track of the start date of the department manager.
- Each department *controls* a number of PROJECTs.
  - Each project has a name, number and is located at a single location.

Example COMPANY Database (Cont.)

- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
- Each employee *works for* one department but may *work on* several projects.
- We keep track of the number of hours per week that an employee currently works on each project.
- We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* several DEPENDENTs.
- For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.


Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
Hours
N
WORKS_ON
PROJECT
Name
Location
Number
supervisor
supervisee
1
SUPERVISION
N
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

Components of the ER Diagram



Components of the ER Diagram - Entity

- Entities are categories of things that are important for a business and about which information must be kept.
- Entities contain facts and information that the business must know and remember.
- Some examples of entities might include the following:
  - PERSON: Agent, insured, employee, customer
  - PLACE: State, country, municipality
  - THING: Inventory item, vehicle, product
  - CONCEPT: Policy, risk, coverage, job
  - ORGANIZATION: Agency, department
  - EVENT: Service request, claim, election

ERD Notation for Entity

Components of the ER Diagram - Entity

- **Tangible Entity**: Tangible Entities are those entities that exist in the real world physically. Example: Person, car, etc.

- **Intangible Entity:** Intangible Entities are those entities that exist only logically and have no physical existence. Example: Bank Account, etc.

ERD Notation for Entity

Entity Type, Entity set

- The entity type is a collection of the entity having similar attributes.
- An entity type in an ER diagram is defined by a name and a set of attributes
- We use a rectangle to represent an entity type in the E-R diagram, not entity.
- The collection of same type of entities that is their attributes are same is called entity set
- We can say that entity type is a superset of the entity set as all the entities are included in the entity type

Entity Type, Entity set

Student → **Entity Type**

Roll_no	Student_name	Age	Mobile_no
1	Andrew	18	7089117222
2	Angel	19	8709054568
3	Priya	20	9864257315
4	Analisa	21	9847852156

→ **E 1**

→ **E 2**

ENTITY SET

E 1
E 2

Entity Type, Entity set

Student → Entity Type

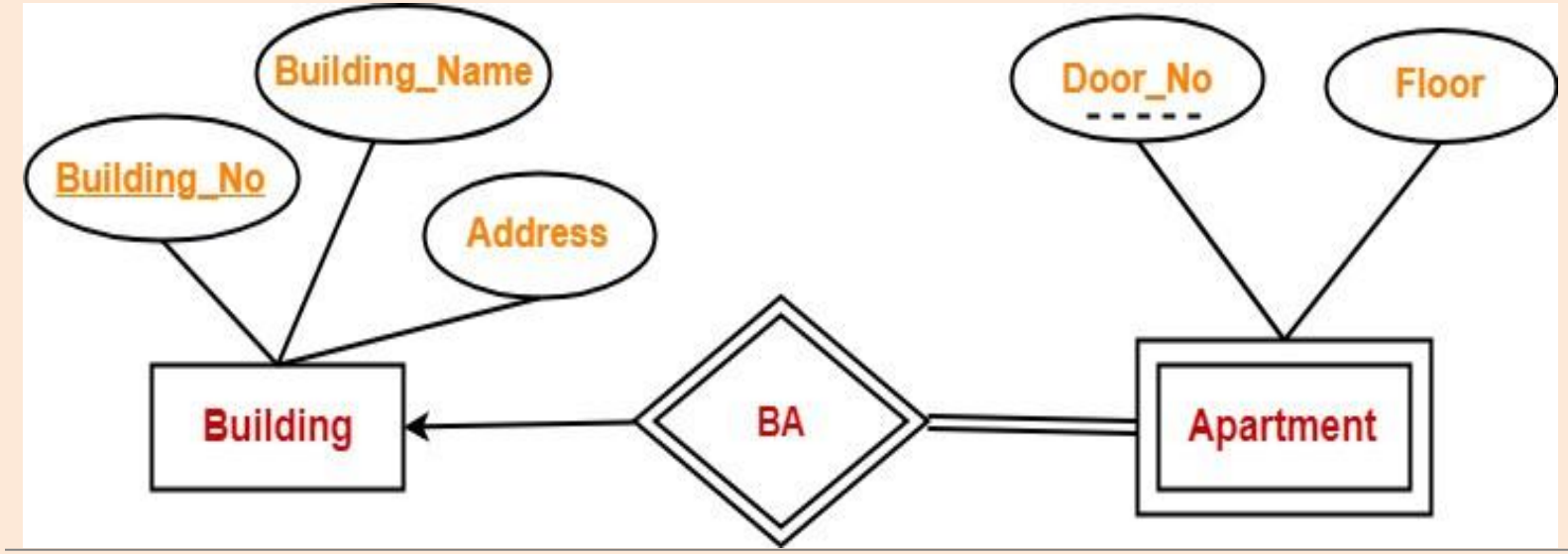
Roll_no	Student_name	Age	Mobile_no	
1	Andrew	18	7089117222	
2	Angel	19	8709054568	→ E 1
3	Priya	20	9864257315	→ E 3
4	Analisa	21	9847852156	→ E 2

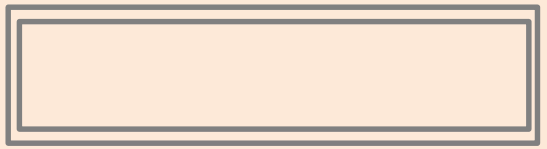
ENTITY SET

E 1
E 2
E 3

Components of the ER Diagram - Weak Entity

- Weak entities depend on some other entity type.
- They don't have primary keys and have no meaning in the diagram without their parent entity.
- It contains a partial key called as discriminator which helps in identifying a group of entities from the entity set

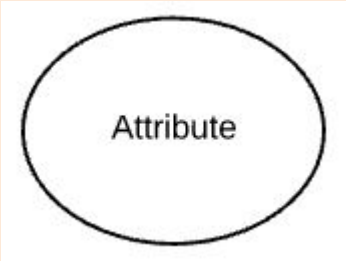




ERD Notation for Weak Entity

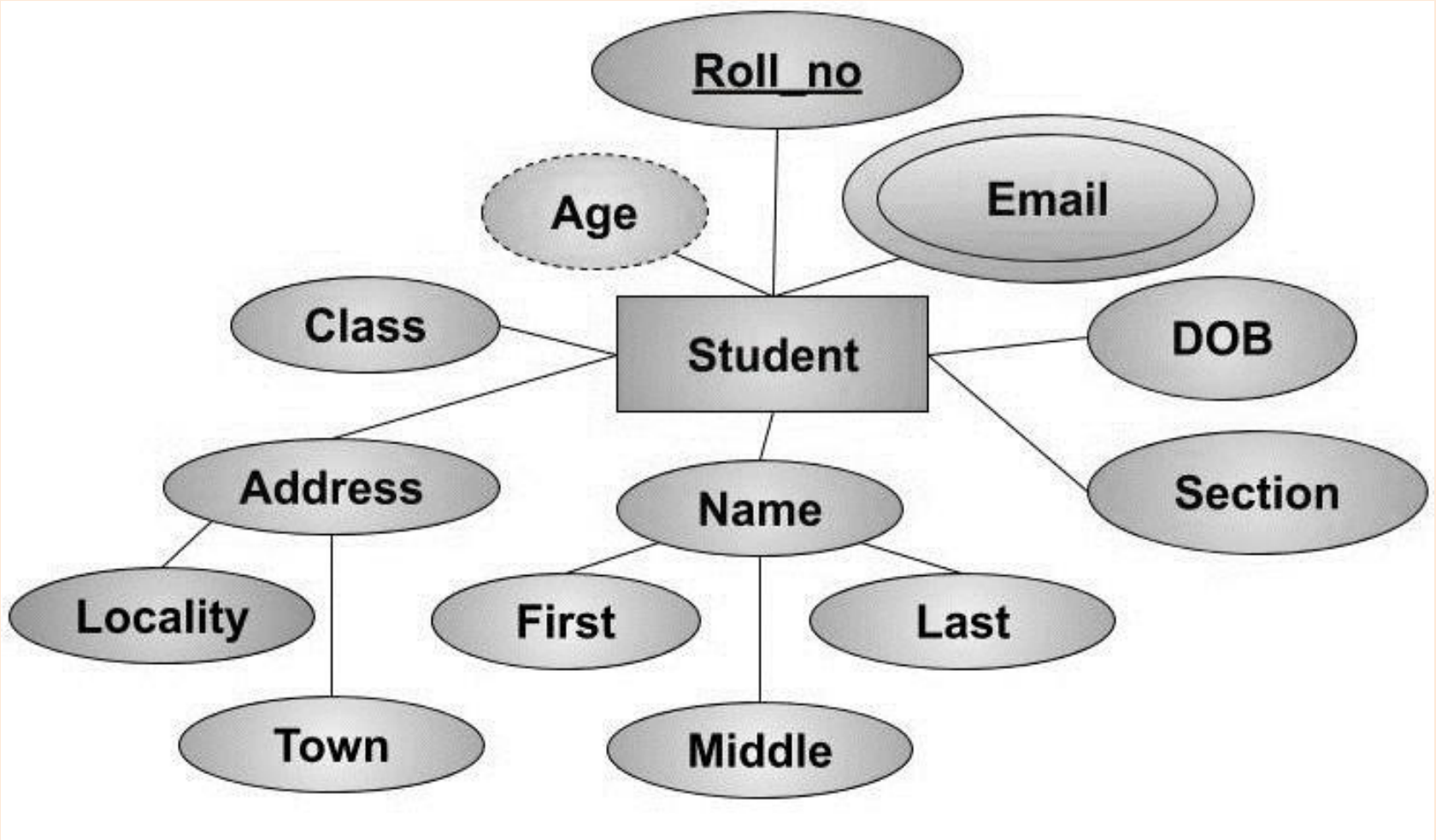
Components of the ER Diagram - Attribute

- A real-world property of an entity type is called an attribute. This is the characteristics of an entity.
- Each attribute can take only a set of permitted values. This is called the domain of that attribute.
  - For example, we define the roll_no of the 'Student' by a numeric value. So, the permitted values are only integers, and hence, 'integer' is the domain of attribute 'roll_no'.
- Each attribute is represented by a separate column in a relational table.



ERD Notation for attribute

Components of the ER Diagram - Attribute



- Various oval symbols represent various types of the attribute.

Components of the ER Diagram - Attribute

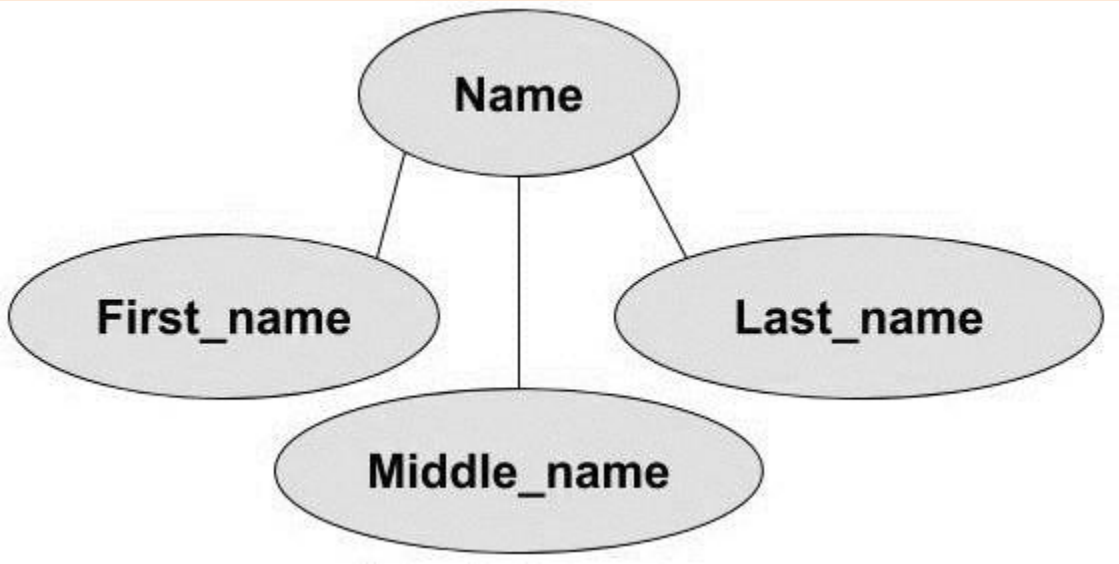
- There are many types of attributes which are as follows:
  - Simple Attribute & Composite Attribute
  - Single Valued Attribute & Multi-valued Attribute
  - Stored Attribute & Derived Attribute
  - Key Attribute & Non-key Attribute

Simple Attribute & Composite Attribute

- A simple attribute contains an atomic value that cannot be further divided. It is simply represented by an oval.
  - For example, Roll_no of Student, Age of Student. It is represented in the E-R diagram as:

Simple Attribute & Composite Attribute

- Composite Attribute - can be further divided into other attributes.
- We represent this in E-R diagram by connecting an oval with oval



- In the table, we will have three columns i.e. First_name, Middle_name, and Last_name.

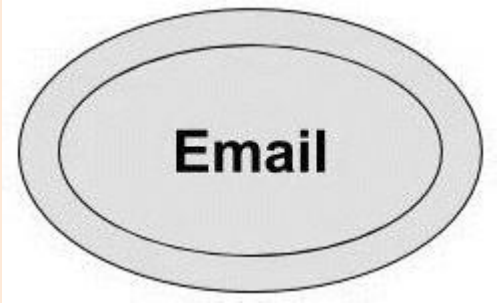
Single Valued Attribute & Multi-valued Attribute

- Single-valued Attribute has only one value
- It is represented by a simple oval.
- Some simple attribute can also be a single-valued attribute.
  - For example, the Section of 'Student' is a simple attribute as it can't be further divided. Also, it is a single-valued attribute because a student can't study in more than one section.

For example, Section of a Student.

Single Valued Attribute & Multi-valued Attribute

- Multivalued Attribute has more than one value
- It is represented by a double oval.
- *Some Composite keys can also be a multivalued attribute.*
  - For example, the address attribute of a student can further be divided into 'Locality', 'Town' and 'PIN'. So, we call it a composite attribute.
- Also, the address of a student can have more than one value.
  - A Student can have more than one Address i.e Permanent Address and Temporary Address. So, it can also be called a multivalued attribute.



For example, A student can have more than one e-mail address.

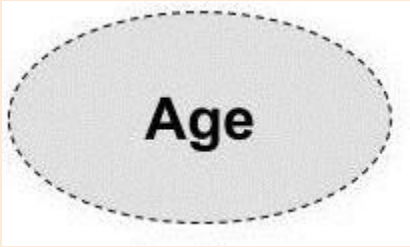
Stored Attribute & Derived Attribute

- Stored Attribute - The value of this attribute should be provided by the user.
- This attribute is stored and can be used for finding the value of other attributes.
- It cannot be derived from any other attribute.
- It is also represented by an oval.
  - For Example, The Date of Birth of 'Student' has to be provided by the student.

Example: Date of Birth(DOB) of Student.

Stored Attribute & Derived Attribute

- Derived Attribute- The value of this attribute is derived from some other attributes.
- We know the value of some other attribute(stored attribute)and from stored attribute, we are deriving the value of this attribute(derived attribute).
- This is done mainly because the value for such attribute keeps on changing. It is represented by a dashed oval.



For example, The value of age attribute is derived from the DOB(date of birth) attribute.

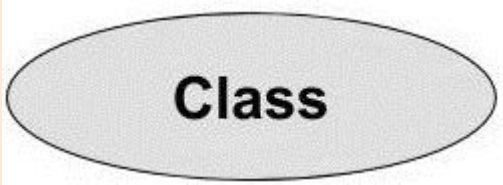
Key Attribute & Non-key Attribute

- A key attribute is used to uniquely identify the entities of an entity type.
- In a relational table, it represents the primary key of the table.
- It is represented by oval and the text in it is underlined.
- Even if all the other attributes of an entity are the same but the key attribute will always be different.

Example: We have Roll_no as the key attribute of the 'Student' because two students can never has same roll number.

Key Attribute & Non-key Attribute

- All the other attributes other than the key attribute are the non-key attributes.
- Two or more entities can have the same value for this attribute.
- For example, the Class attribute would have the same value for all those students who are studying in the same class.



Example: Class, Section, Age, Name etc, are the non-key attributes.

Null Valued Attributes

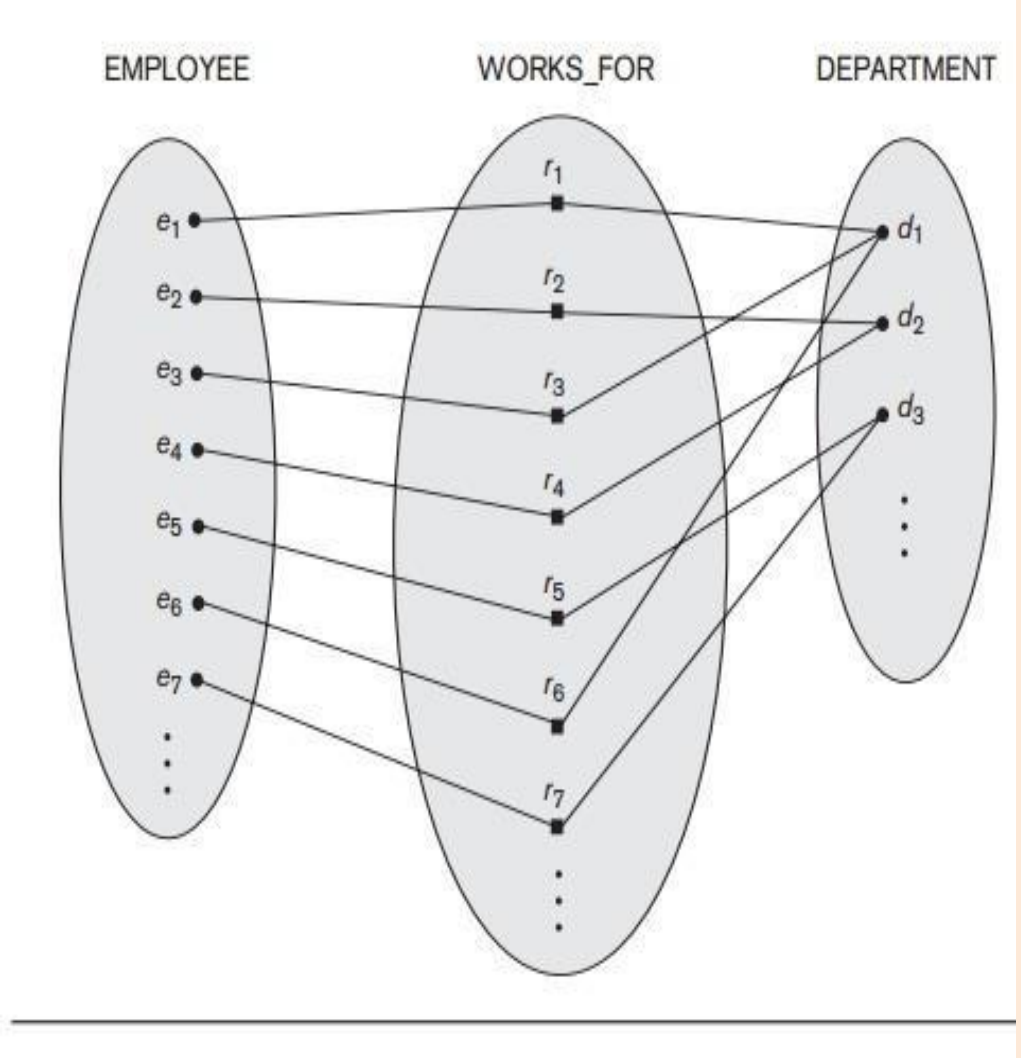
- The attributes which can have a null value called null valued attributes.
- Example: Mobile_noattributes of a personmay notbe havingmobile phones.

Relationship Type and Relationship Set:

- A relationship type represents the association between entity types.
- For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course.
- In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.
- Each relationship has
  - Name
  - Degree
  - Cardinality ratio

Relationship Type and Relationship Set:

- A set of relationships of same type is known as relationship set.
  - For example, consider a relationship type WORKS_FOR between two entity types EMPLOYEE and DEPARTMENT
- which associates each employee with the department for which the employee works in the corresponding entity set.
- Each relationship instance in the relationship set WORKS_FOR associates one EMPLOYEE entity and one DEPARTMENT entity.

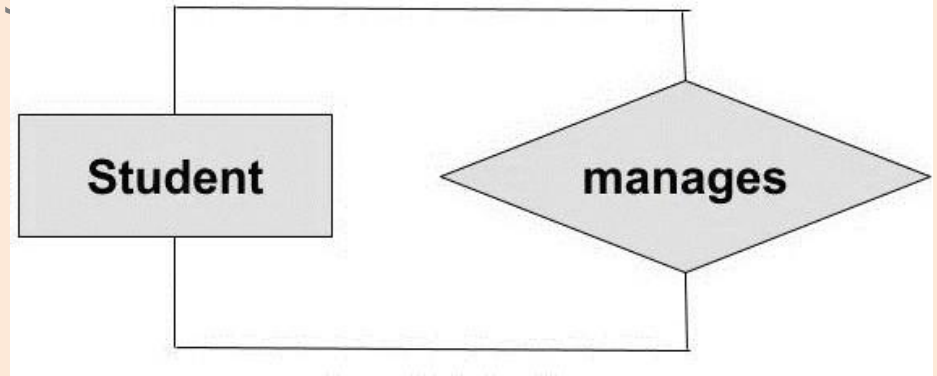


Degree of Relationship

- The degree of a relationship is the number of entity types that participate(associate) in a relationship.
- Based on the number of entity types that are connected we have the following degree of relationships:
  - Unary
  - Binary
  - Ternary
  - N-ary

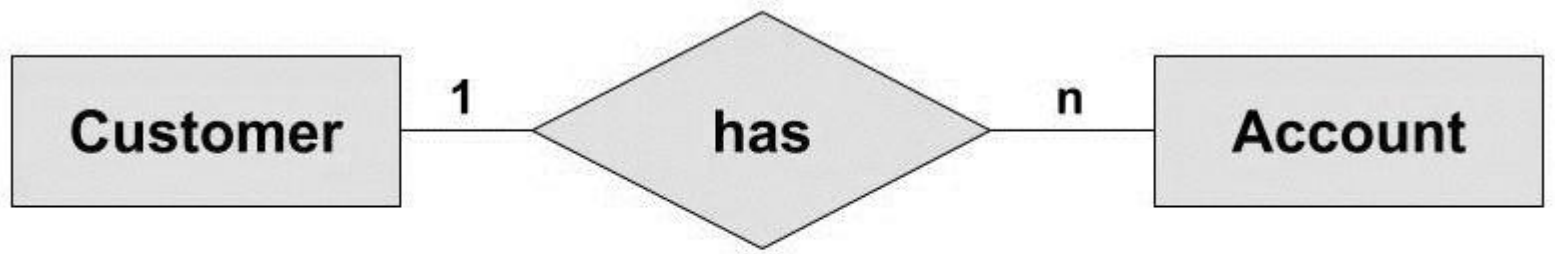
Degree of Relationship - Unary (degree 1)

- A unary relationship exists when both the participating entity type are the same.
  - For example, Suppose in a classroom, we have many students who belong to a particular club-like dance club, basketball club etc. and some of them are club leads.
  - So, a particular group of student is managed by their respective club lead.
  - Here, the group is formed from students and also, the club leads are chosen from students.
  - So, the 'Student' is the only entity participating here. We can represent this relationship using the E-R diagram as follows:



Degree of Relationship - Binary (degree 2)

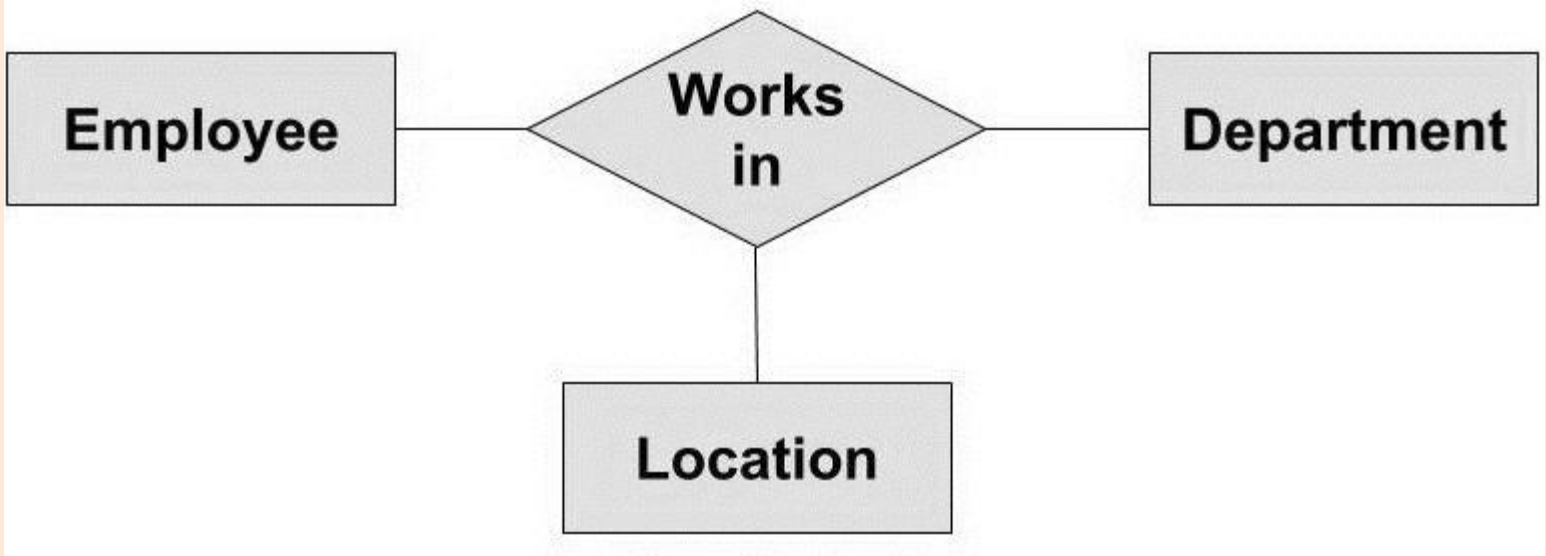
- A binary relationship exists when exactly two entity type participates.
- This is the most common degree of relationship.
- It is easy to deal with such relationship as these can be easily converted into relational tables.



Binary Relationship
(degree 2)

Degree of Relationship - Ternary(degree 3)

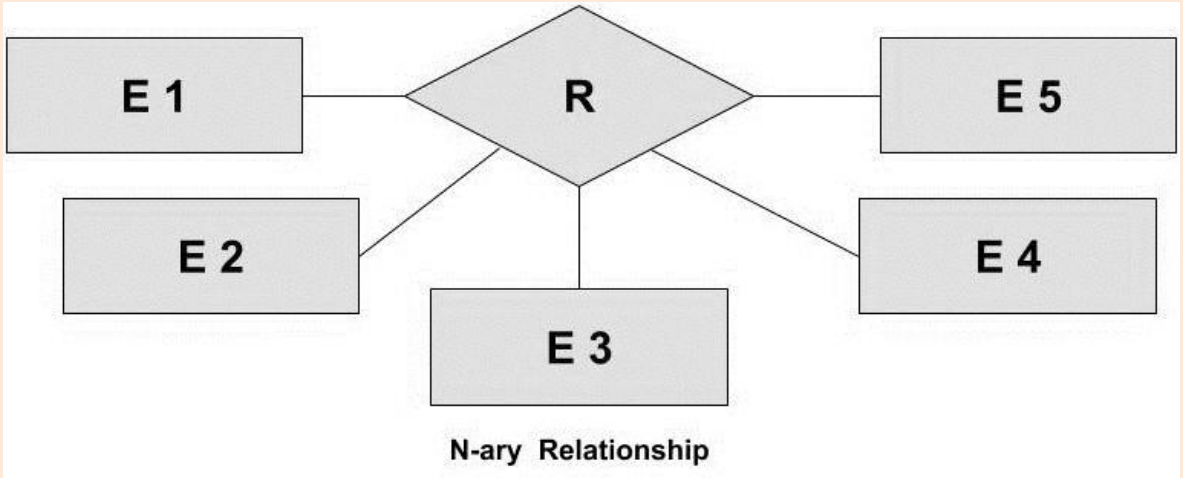
- A ternary relationship exists when exactly three entity type participates.
- As thenumber of entity increases in therelationship, it becomes complex to convert them into relational tables.

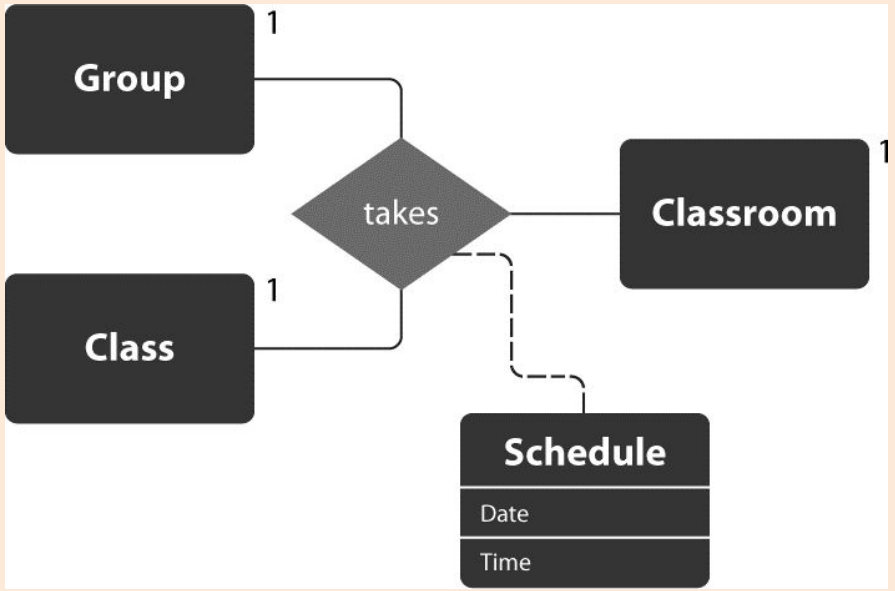


Degree of Relationship - N-ary (n degree)

- An N-ary relationship exists when ‘n’ number of entities are participating.
- So, any number of entities can participate in a relationship.
- There is no limitation to the maximum number of entities that can participate.
- But, relations with a higher degree are not common. This is because the conversion of higher degree relations to relational tables gets complex
- We are making an E-R model because it can be easily be converted into any other model for implementing the database.
- But, this benefit is not available if we use higher degree relations.

Degree of Relationship - N-ary (n degree)

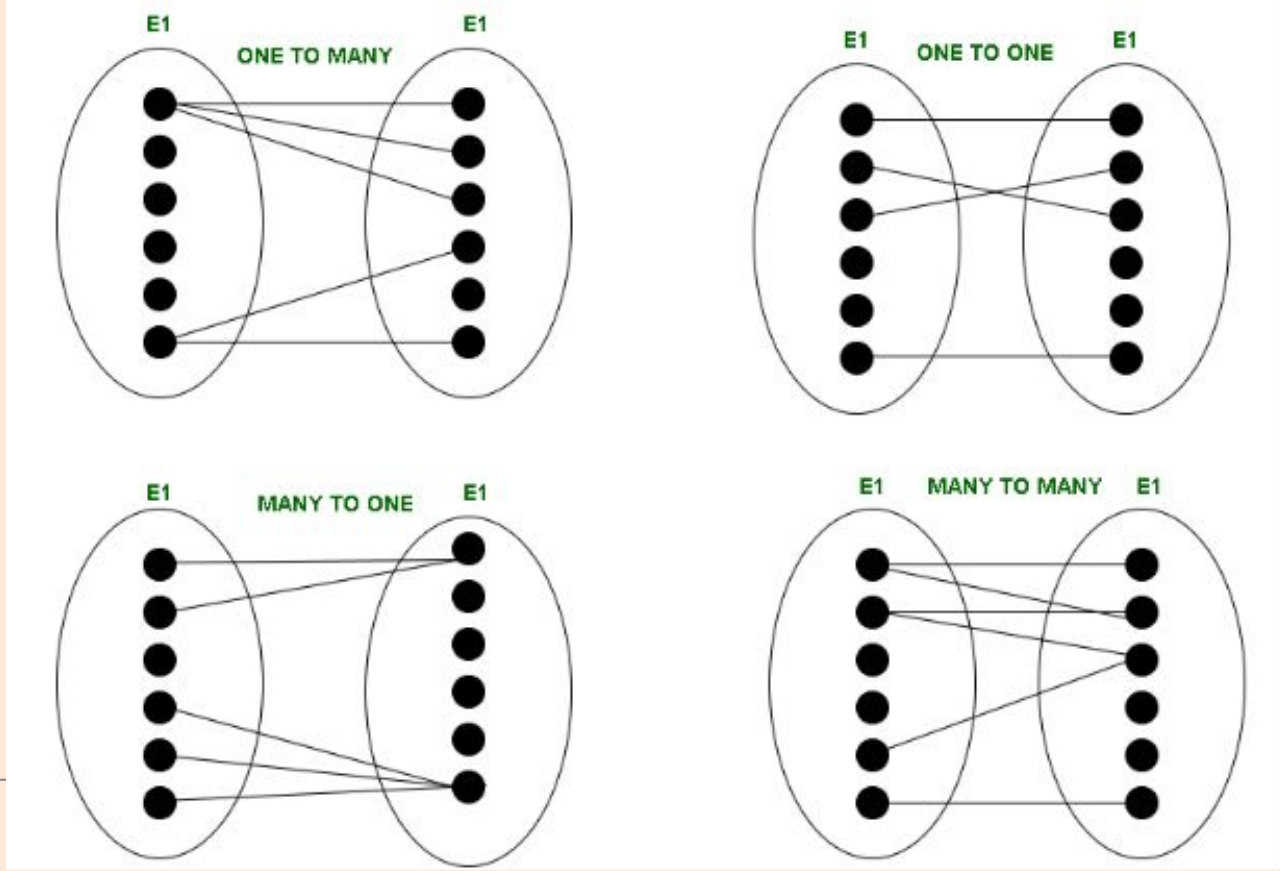




Database diagram that shows how we want to store the time and date and in which classroom a group takes classes.

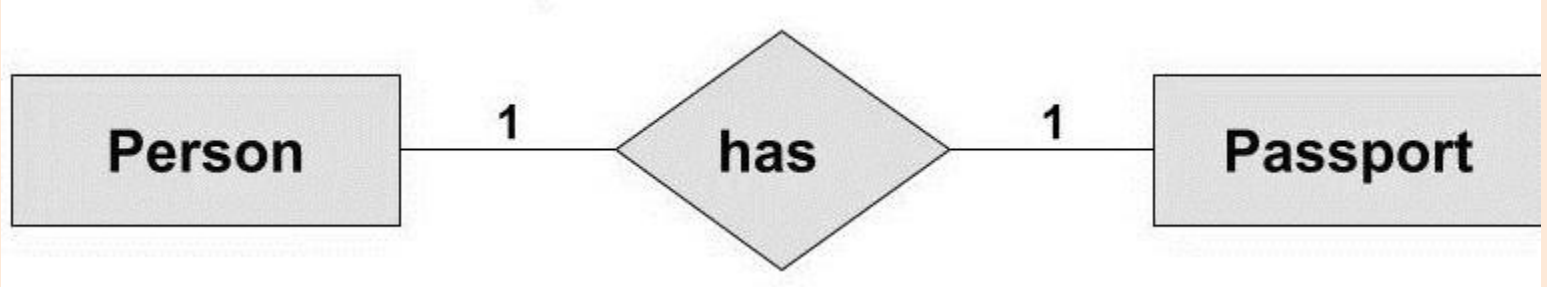
Cardinality Ratio

- A relationship where two entities are participating is called a binary relationship.
- Cardinality is the number of instance of an entity from a relation that can be associated with the relation.
- There are four types Cardinality
  - One to One
  - One to Many
  - Many to One
  - Many to Many



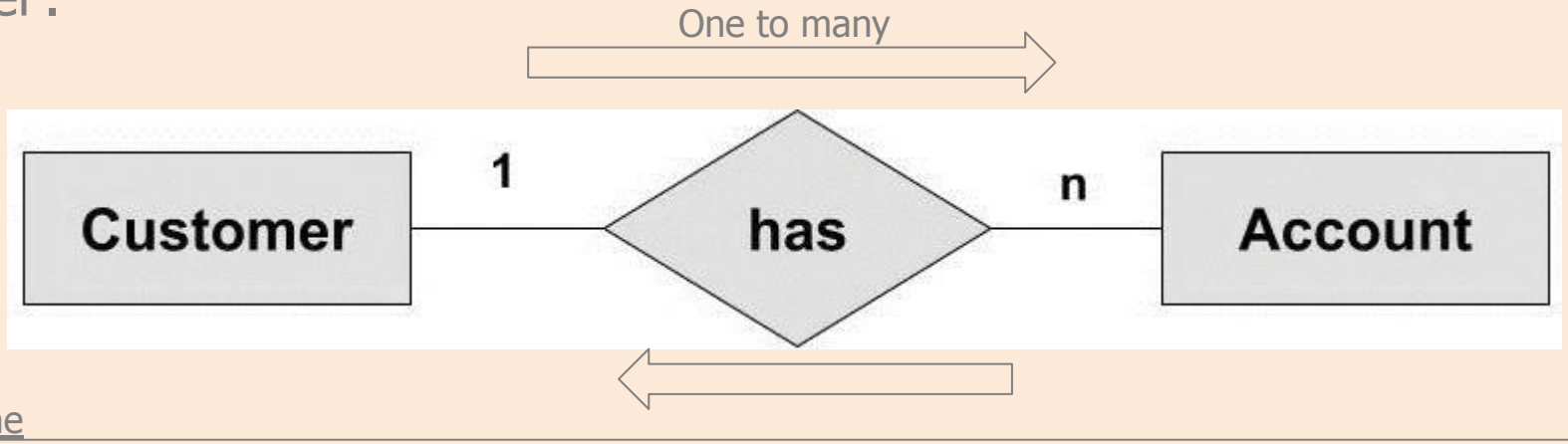
Cardinality Ratio - One to One

- Such a relationship exists when each record of one table is related to only one record of the other table.
  - For example, If there aretwo entities 'Person' (Id, Name, Age, Address)and 'Passport'(Passport_id, Passport_no).
  - So, each person can have only one passport and each passport belongs to only one person.



Cardinality Ratio - One-to-Many or Many-to-One Relationship

- Such a relationship exists when each record of one table can be related to one or more than one record of the other table.
- This relationship is the most common relationship found.
- A one-to-many relationship can also be said as a many-to-one relationship depending upon the way we view it.
  - For example, If there are two entity type 'Customer' and 'Account' then each 'Customer' can have more than one 'Account' but each 'Account' is held by only one 'Customer'.

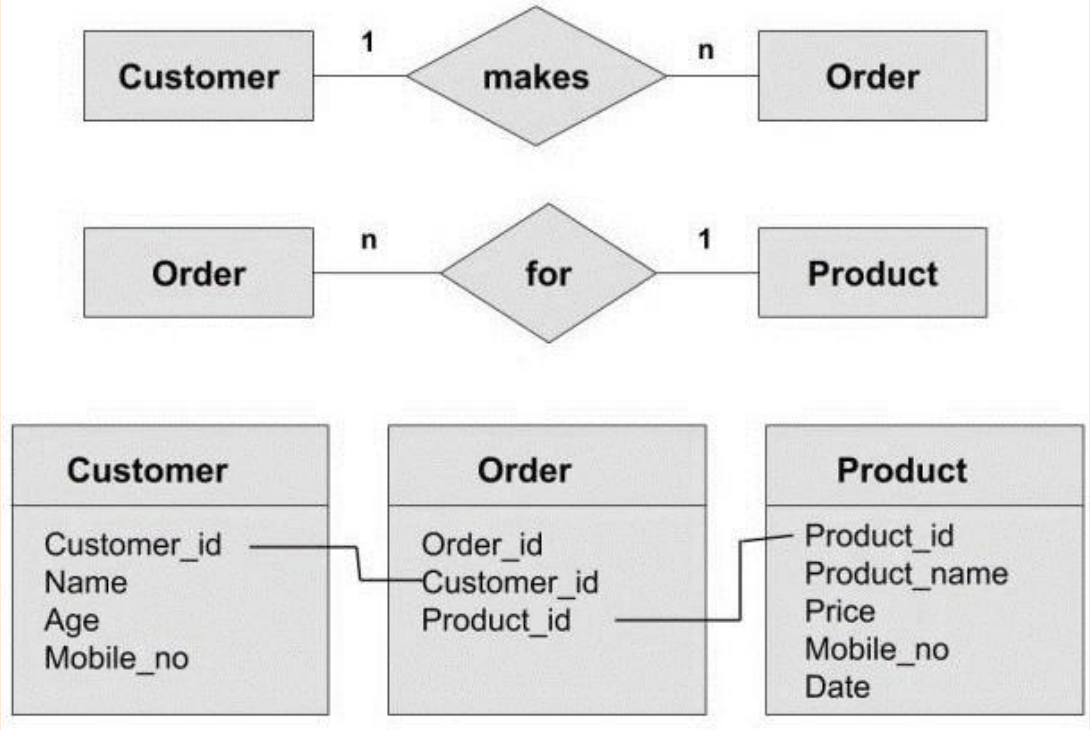


Cardinality Ratio - Many-to-Many Relationship

- Such a relationship exists when each record of the first table can be related to one or more than one record of the second table and a single record of the second table can be related to one or more than one record of the first table.
  - Example: If there are two entity type 'Customer' and 'Product' then each customer can buy more than one product and a product can be bought by many different customers.

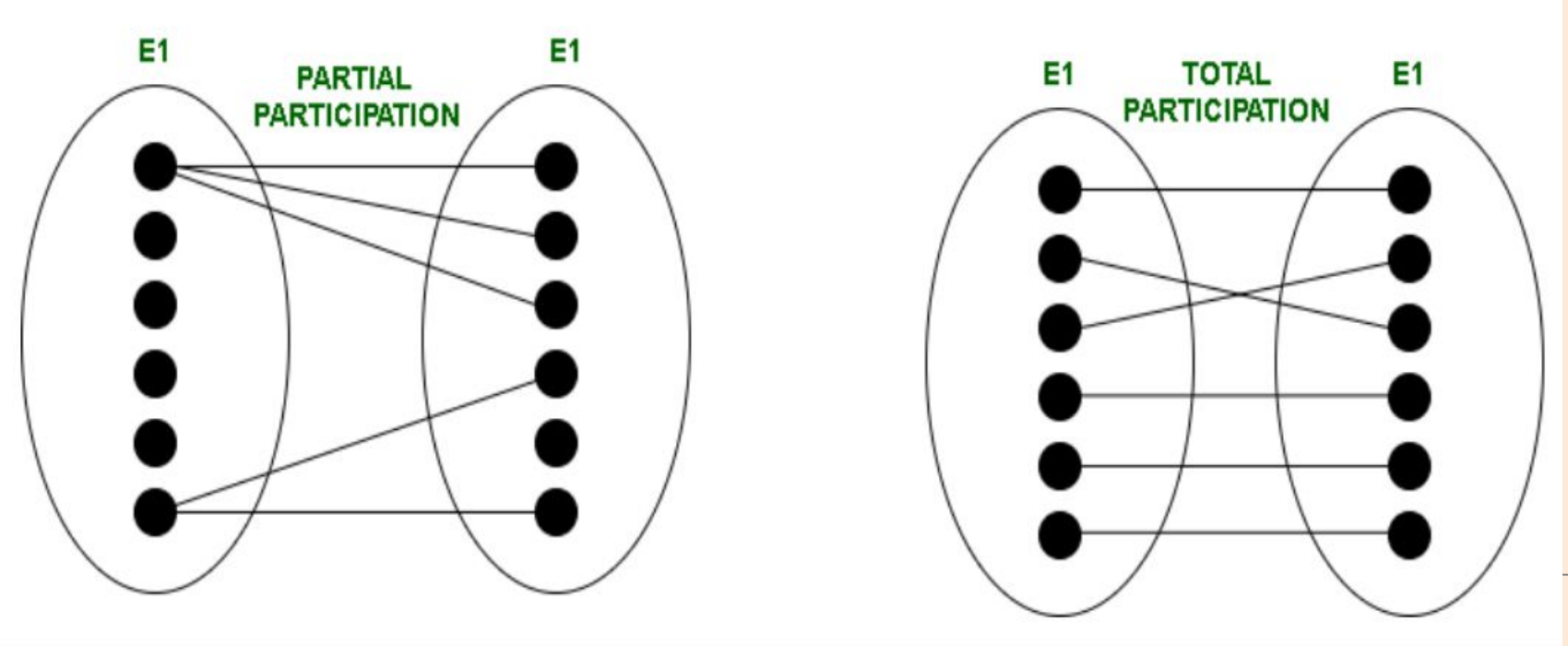
Cardinality Ratio - Many-to-Many Relationship

- A many-to-many relationship can be seen as a two one-to-many relationship which is linked by a 'linking table' or 'associate table'.
- The linking table links two tables by having fields which are the primary key of the other two tables.



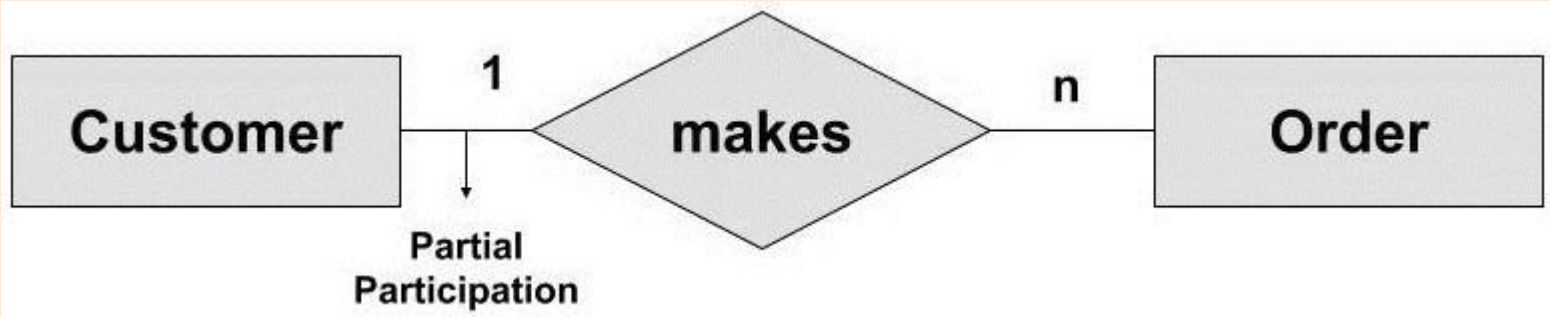
Participation Constraints

- The relationship can be between two strong entity or a strong entity and a weak entity.
- Depending upon the type of entity participating in the relationship, the participation can be partial or total.



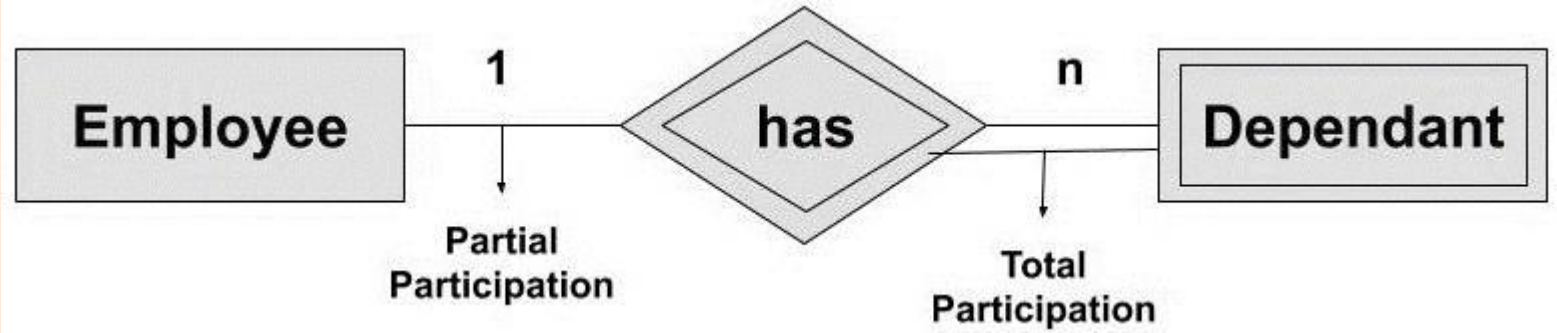
Participation Constraints - Partial Participation

- Partial Participation: Partial Participation exists when all the entity of an entity type is not associated with one or the other entity of another entity type.
- This is represented by joining the relationship with the entity type with the help of one line.
  - Example: We have two entity type 'Customer' and 'Order'. Then there can be 'Customer' who have not done any order. So, here there is partial participation of the entity in the relationship.

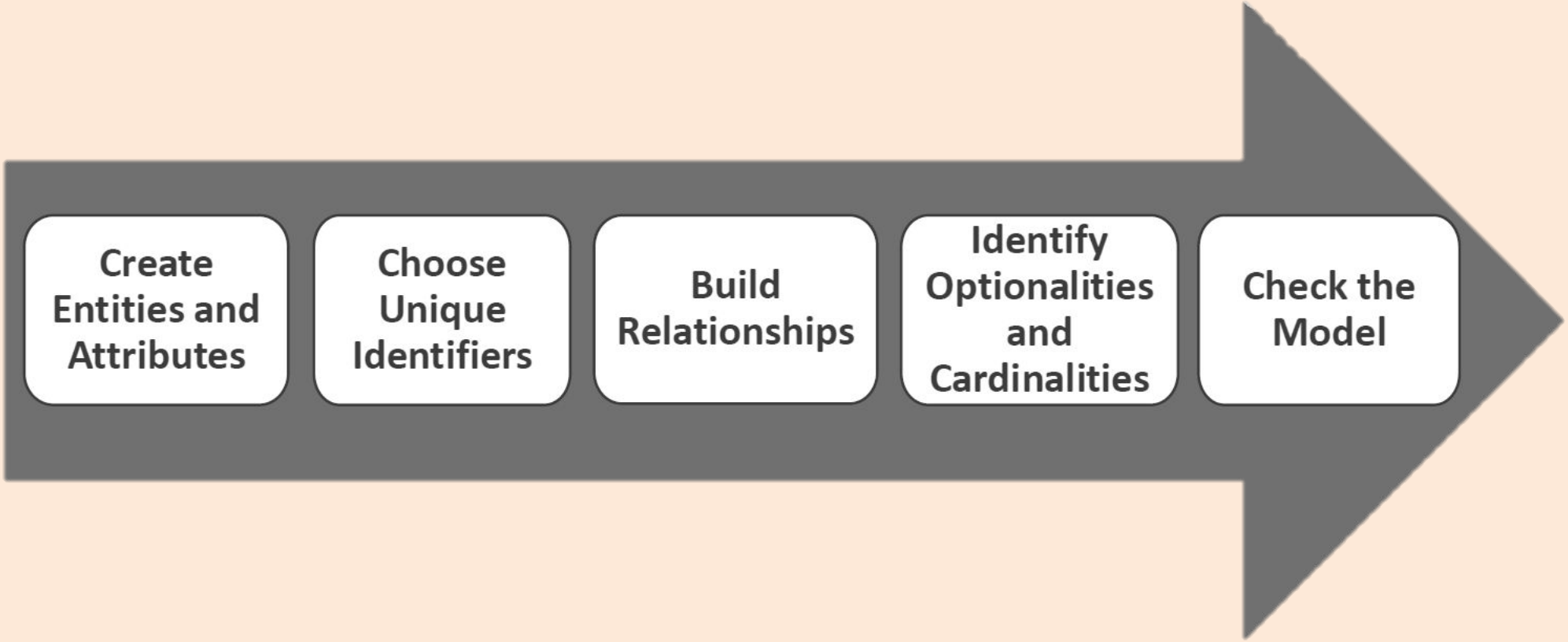


Participation Constraints - Total Participation

- Total Participation exists when all theentity of an entity type is associated with one or the other entity of another entity type.
- This is represented by joining the relationship with the entity type with the help of a double parallel line.
- Such a relationship usually exist between a strong entity and a weak entity.

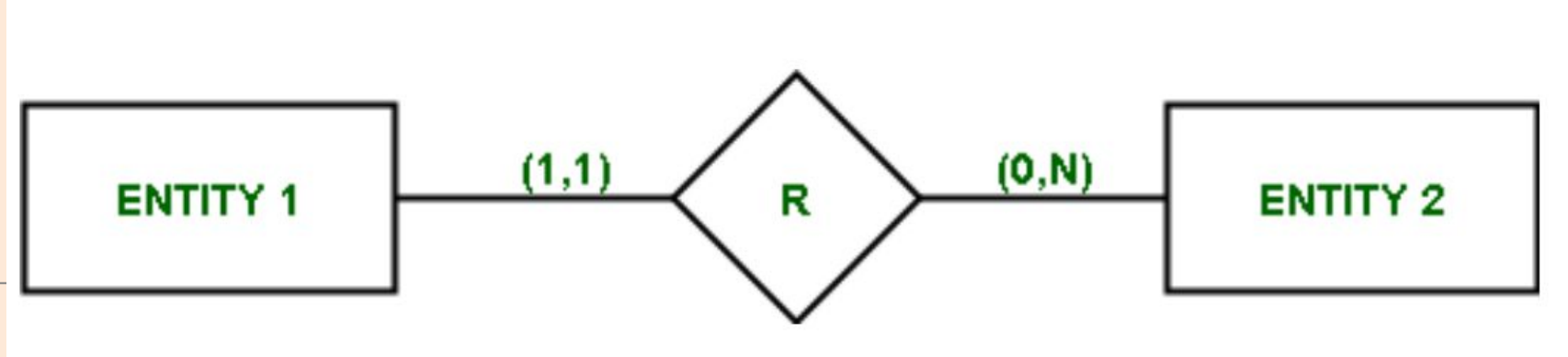


Steps to Build an ERD



Structural Constraints:

- Structural Constraints are also called Structural properties of a database management system (DBMS).
- Cardinality Ratios and Participation Constraints taken together are called Structural Constraints.
- The Structural constraints are represented by **Min-Max notation**.
- This is a pair of numbers(m, n) that appear on the connecting line between the entities and their relationships.



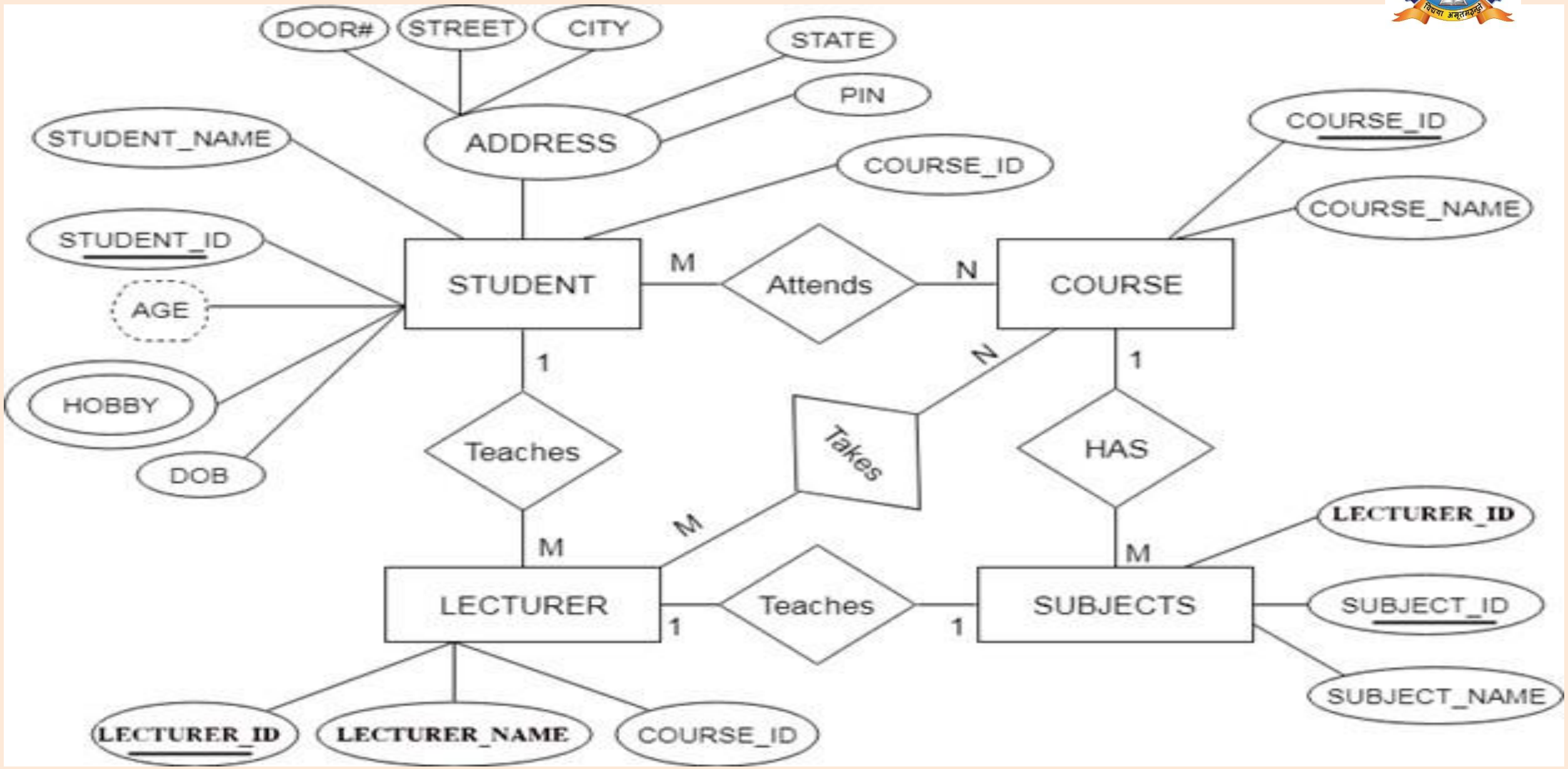
Min-Max notation

- The minimum number of times an entity can appear in a relation is represented by m whereas, the maximum time it is available is denoted by n.
- If m is 0 it signifies that the entity is participating in the relation partially, whereas, if m is either greater than or equal to 1, it denotes total participation of the entity.

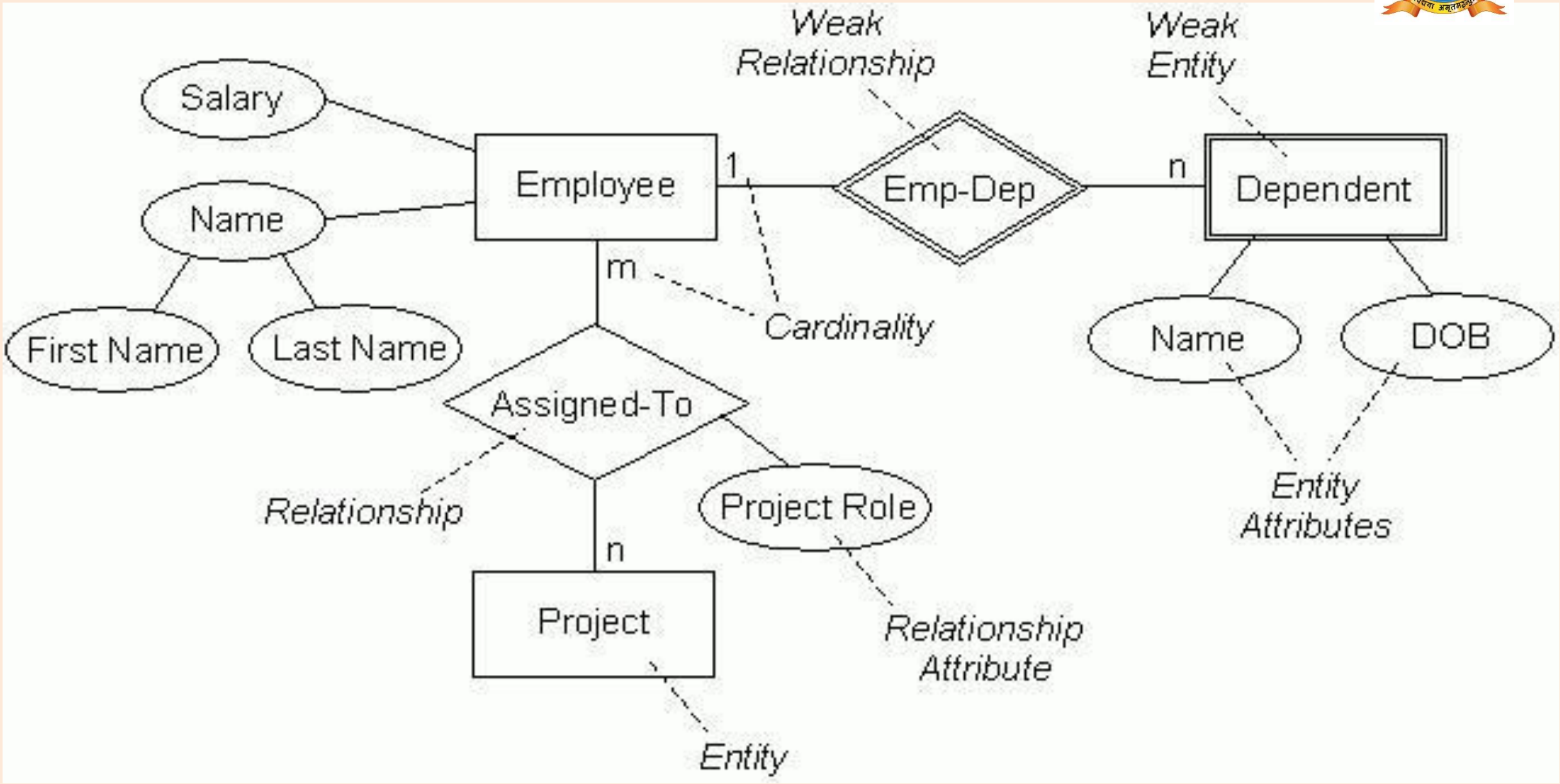
CST 204 - Database Management Systems

ER Diagram Examples

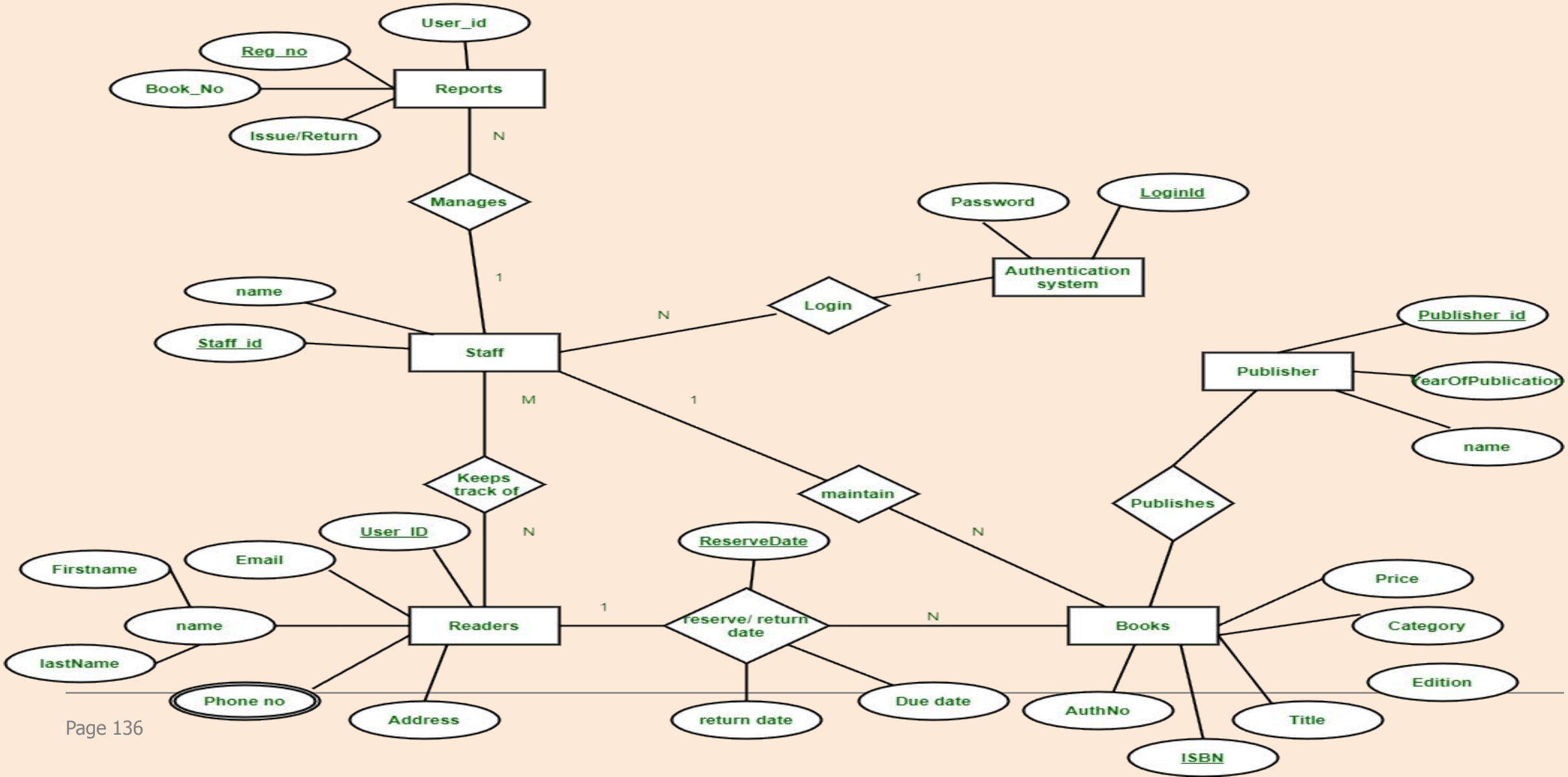
ER Model Case Studies



ER Model Case Studies



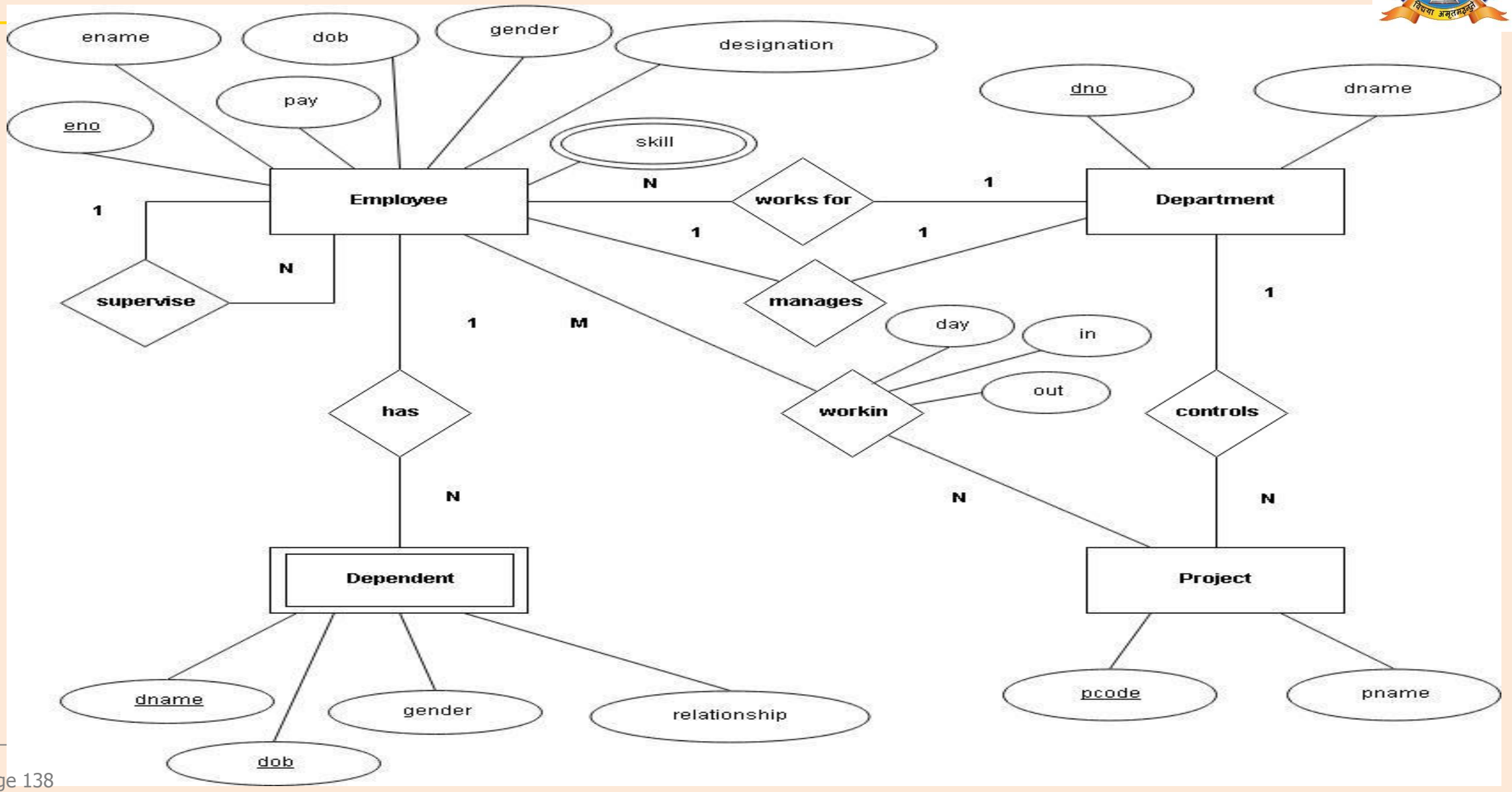
ER Model Case Studies – Library Management System



Problem Definition

- The company organized into departments and departments have employees working in it.
- Attributes of Department are dno, dname. Attributes of Employee include eno, name, dob,gender, doj, designation, basic_pay, panno, skills. Skills are multi-valued attribute.
- The Department has a manager managing it. There are also supervisors in Department who supervises a set of employees.
- Each Department enrolls a number of projects. Attributes of Project are pcode, pname. A project is enrolled by a department. An employee can work on any number of projects on a given day. The Date of employee work in-time and out-time has to keep track.
- The Company also maintains details of the dependents of each employees. Attributes of dependent include dname, dob, gender and relationship with the employee.
- Model an entity-relationship diagram for the above scenario.

Solution

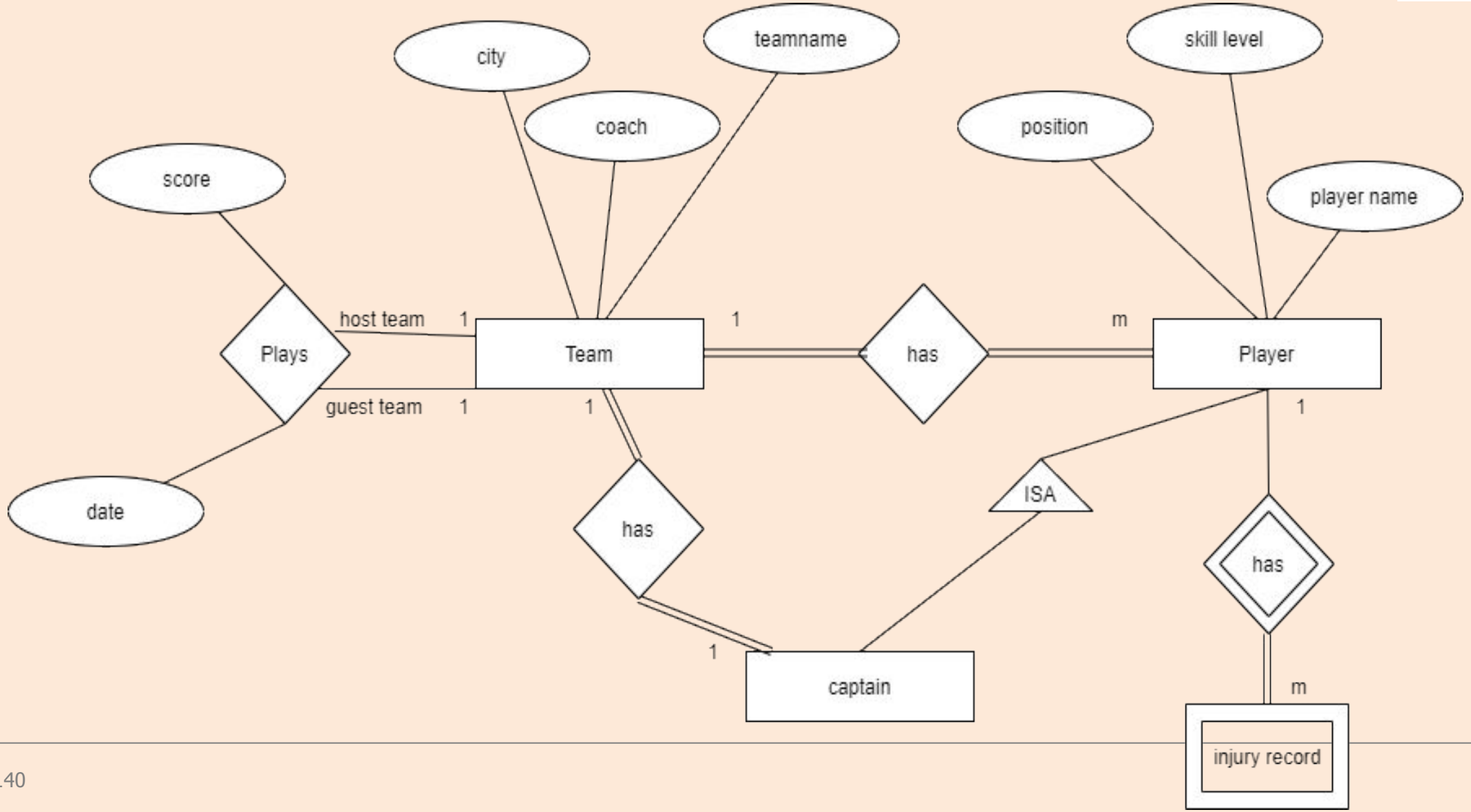

ename
dob
gender
designation
pay
eno
skill
dno
dname
Employee
N
works for
1
Department
1
1
1
N
supervise
manages
1
1
M
day
in
out
has
workin
controls
N
N
N
Dependent
Project
dname
gender
relationship
dob
pcode
pname

ER Model Case Studies

- Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):
  - the NHL has many teams,
  - each team has a name, a city, a coach, a captain, and a set of players,
  - each player belongs to only one team,
  - each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,
  - a team captain is also a player,
  - a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 2017) and a score (such as 4 to 2).

  Construct a clean and concise ER diagram for the NHL database using the Chen notation as in your textbook. List your assumptions and clearly indicate the cardinality mappings as well as any role indicators in your ER diagram.

ER Model Case Studies



Refrences

- https://afteracademy.com/blog/what-are-the-different-types-of-relationships-in-dbms