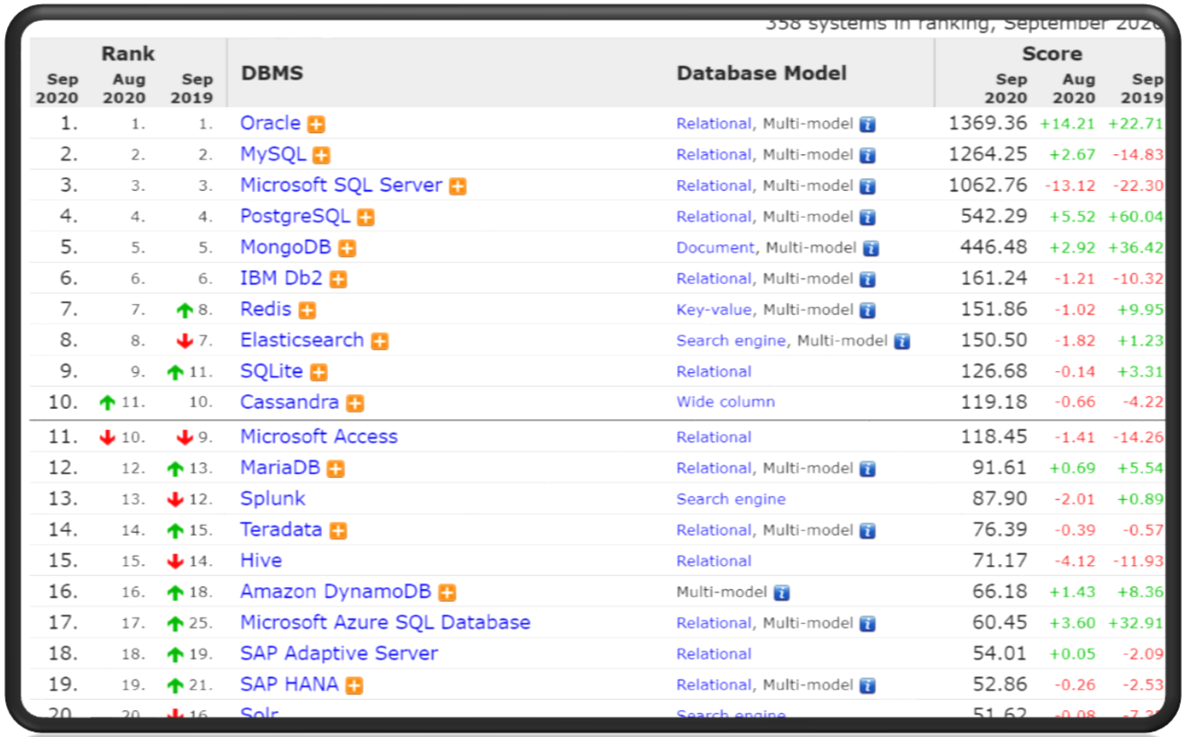
Recap of the Last Session

- ✓ What is DBMS
- ✓ Applications and Relevance
- ✓ Course Content and Delivery
- ✓ Course Outcomes
- ✓ Program Outcomes and Program Specific Outcomes
- ✓ Related the DBMS Course to the Placement, Higher Studies and Entrepreneurship

Module-1
Introduction to Databases



358 systems in ranking, September 2020

Rank			DBMS	Database Model	Score		
Sep 2020	Aug 2020	Sep 2019			Sep 2020	Aug 2020	Sep 2019
1.	1.	1.	Oracle	Relational, Multi-model	1369.36	+14.21	+22.71
2.	2.	2.	MySQL	Relational, Multi-model	1264.25	+2.67	-14.83
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1062.76	-13.12	-22.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	542.29	+5.52	+60.04
5.	5.	5.	MongoDB	Document, Multi-model	446.48	+2.92	+36.42
6.	6.	6.	IBM Db2	Relational, Multi-model	161.24	-1.21	-10.32
7.	7.	↑ 8.	Redis	Key-value, Multi-model	151.86	-1.02	+9.95
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	150.50	-1.82	+1.23
9.	9.	↑ 11.	SQLite	Relational	126.68	-0.14	+3.31
10.	↑ 11.	10.	Cassandra	Wide column	119.18	-0.66	-4.22
11.	↓ 10.	↓ 9.	Microsoft Access	Relational	118.45	-1.41	-14.26
12.	12.	↑ 13.	MariaDB	Relational, Multi-model	91.61	+0.69	+5.54
13.	13.	↓ 12.	Splunk	Search engine	87.90	-2.01	+0.89
14.	14.	↑ 15.	Teradata	Relational, Multi-model	76.39	-0.39	-0.57
15.	15.	↓ 14.	Hive	Relational	71.17	-4.12	-11.93
16.	16.	↑ 18.	Amazon DynamoDB	Multi-model	66.18	+1.43	+8.36
17.	17.	↑ 25.	Microsoft Azure SQL Database	Relational, Multi-model	60.45	+3.60	+32.91
18.	18.	↑ 19.	SAP Adaptive Server	Relational	54.01	+0.05	-2.09
19.	19.	↑ 21.	SAP HANA	Relational, Multi-model	52.86	-0.26	-2.53
20.	20.	↓ 16.	Solr	Search engine	51.62	-0.08	[illegible]

Topics of Module-1

Introduction to Databases: Introduction, Characteristics of database approach, Advantages of using the DBMS approach, History of database applications. **Overview of Database Languages and Architectures:** Data Models, Schemas, and Instances. Three schema architecture and data independence, database languages, and interfaces, The Database System environment. **Conceptual Data Modelling using Entities and Relationships:** Entity types, Entity sets, attributes, roles, and structural constraints, Weak entity types, ER diagrams, examples, Specialization and Generalization. **10 Hours**

Textbook 1:Ch 1.1 to 1.8, 2.1 to 2.6, 3.1 to 3.10

Text Books:

1. Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill

Learning Objectives

On Completion of this Module, the learner will be able to:

1. Understand the basic concepts of Database and Evolution of database
2. Understand the Data Models, Schemas, and Instances, Database system environment
3. Understand the concepts of ER Modelling
4. Apply and draw ER Diagrams for various problem statement

Quote for the Day

With data collection, 'the sooner the better' is always the best answer."

- **Marissa Mayer**

American businesswoman and investor

"You can have data without information, but you cannot have information without data."

- **Daniel Keys Moran**

American computer programmer and science fiction writer

Basic Definitions

Data: Raw facts that can be recorded/acquired which has an implicit meaning. Ex- Age, Color, name..etc

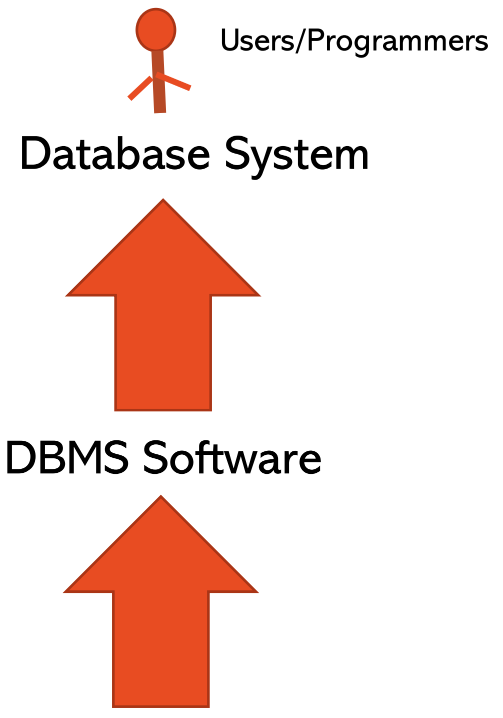
Database: A collection of related data, organized in a proper manner for effective and efficient storage and retrieval purpose.

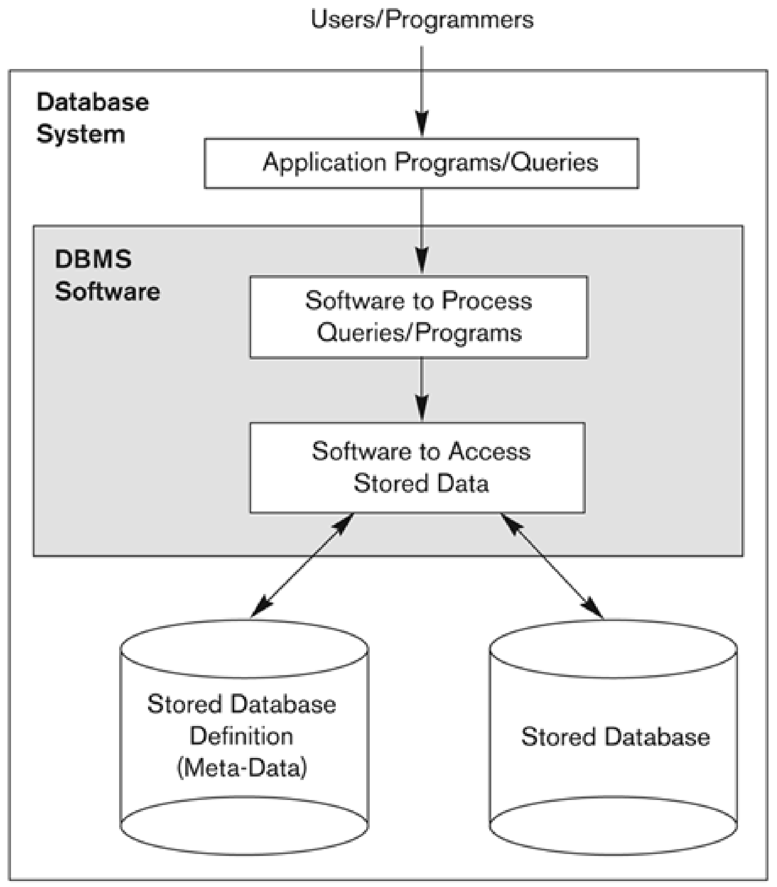
Database Management System (DBMS): A software package/ system to facilitate the creation and maintenance of a computerized database.

Mini-world (DB - Problem Statement): Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.

Basic Definitions (Cont…)

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




Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Simplified database system environment

Main Characteristics of the Database Approach

There are 5 Main characteristics of the database approach:

1. **Self-describing nature of a database system**
2. **Insulation between programs and data**
3. **Data Abstraction**
4. **Support of multiple views of the data**
5. **Sharing of data and multi-user transaction processing**

1.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.

Main Characteristics of the Database Approach (Cont…)

2. Insulation between programs and data:

- It is also called as **program-data independence**.
- Allows changing data structures and storage organization without having to change the DBMS access programs.

3. 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

4. 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.

Main Characteristics of the Database Approach (Cont…)

5. 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 using the DBMS Approach (Cont…)

- 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
- 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.
- Enforcing integrity constraints on the database.
- Drawing inferences and actions from the stored data using deductive and active rules

Historical Development of Database Technology

Early Database Applications:

- The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.
- A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model.

Relational Model based Systems:

Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.

Relational DBMS Products emerged in the early 1980s.

Historical Development of Database Technology (Cont…)

Object-oriented and emerging applications:

- Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
- Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relational* DBMSs (ORDBMSs)
- *Extended relational* systems add further capabilities (e.g. for multimedia data, XML, and other data types)

Data on the Web and E-commerce Applications:

- Web contains data in HTML (Hypertext markup language) with links among pages.
- This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language).
- Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database.
- Allow database updates through web pages

Extending Database Capabilities

New functionality is being added to DBMSs in the following areas:

- Scientific Applications
- XML (eXtensible Markup Language)
- Image Storage and Management
- Audio and Video Data Management
- Data Warehousing and Data Mining
- Spatial Data Management
- Time Series and Historical Data Management

The above gives rise to *new research and development* in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

Overview of Database Languages and Architectures (Cont…)

Data Model: A set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.

Data Model Structure and Constraints:

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

Overview of Database Languages and Architectures (Cont…)

Data Model Operations:

- These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
- Operations on the data model may include ***basic model operations***
- (Ex. generic insert, delete, update) and ***user-defined operations***
- (Ex. compute_student_gpa, update_inventory)

Categories of Data Models

Conceptual (high-level, semantic) data models:

Provide concepts that are close to the way many users perceive data. Also called ***entity-based*** or ***object-based*** data models.

Physical (low-level, internal) data models:

Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals

Implementation (representational) data models:

Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Schemas versus Instances

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.

Schemas versus Instances (Cont…)

Database State:
This includes the collection of all the data in the database.
Also called database instance (or occurrence or snapshot).
The term *instance* is also applied to individual database components,
Ex: *record instance, table instance, entity instance*
The actual data stored in a database at a ***particular moment in time***.

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.

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**.

Schema Diagram for the database

- Student and Course Information Database

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

A database that stores student and course information.

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

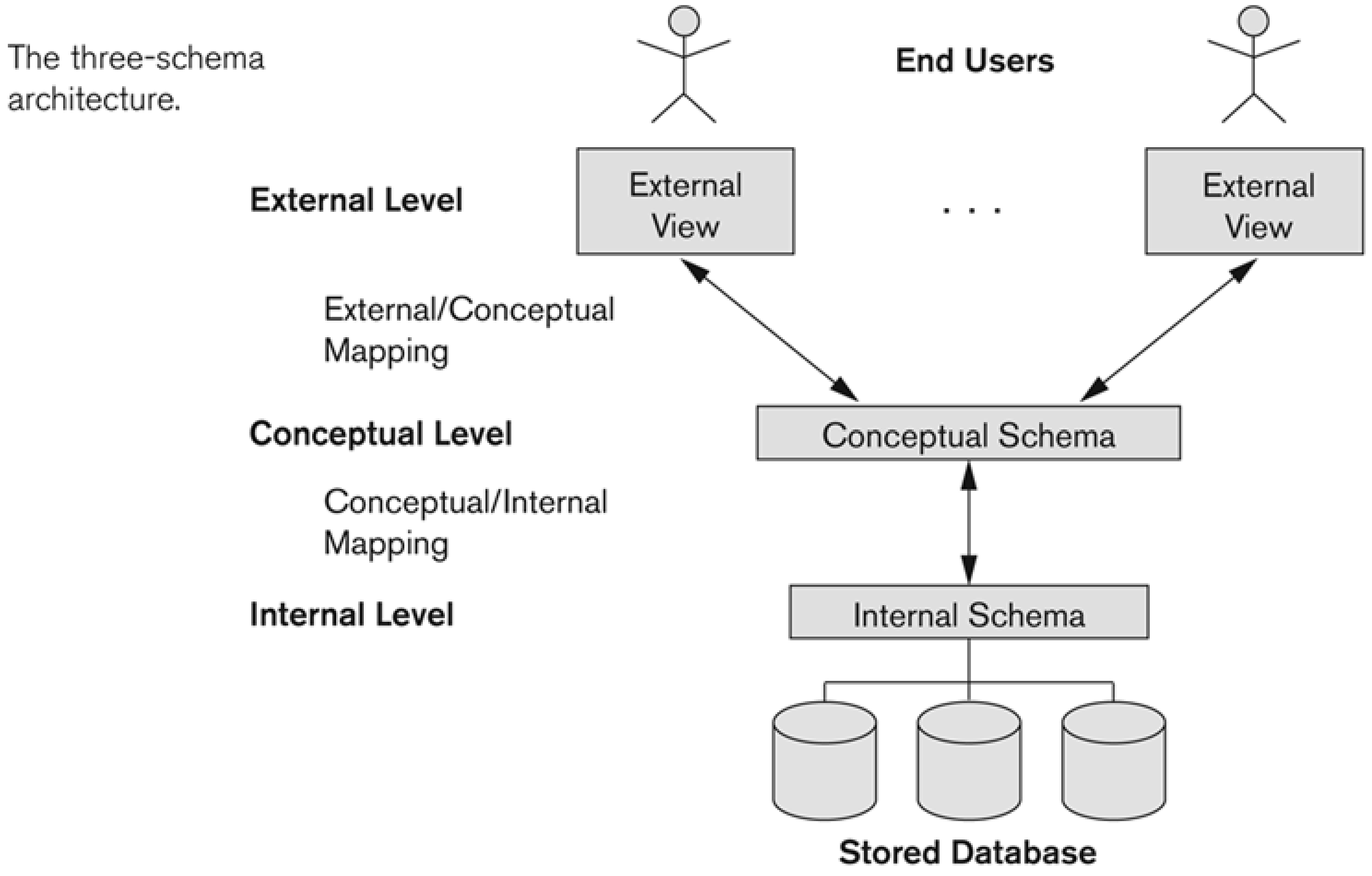
A database that stores student and course information.

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.

The Three-schema architecture



The Three-schema Architecture (Cont..)

Mappings among schema levels 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 (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

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

Data Independence (continued)

- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are **unchanged**.
  - Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

- **Data Definition Language (DDL):**
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - In many DBMSs, the DDL is also used to define internal and external schemas (views).
  - In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
    - SDL is typically realized via DBMS commands provided to the DBA and database designers

- **Data Manipulation Language (DML):**
  - Used to specify database retrievals and updates
  - DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
    - A library of functions can also be provided to access the DBMS from a programming language
  - Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

Types of DML

- **High Level or Non-procedural Language:**
  - For example, the SQL relational language
  - Are "set"-oriented and specify what data to retrieve rather than how to retrieve it.
  - Also called **declarative** languages.
- **Low Level or Procedural Language:**
  - Retrieve data one record-at-a-time;
  - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - **Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach**: e.g. JDBC for Java, ODBC for other programming languages
  - **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components

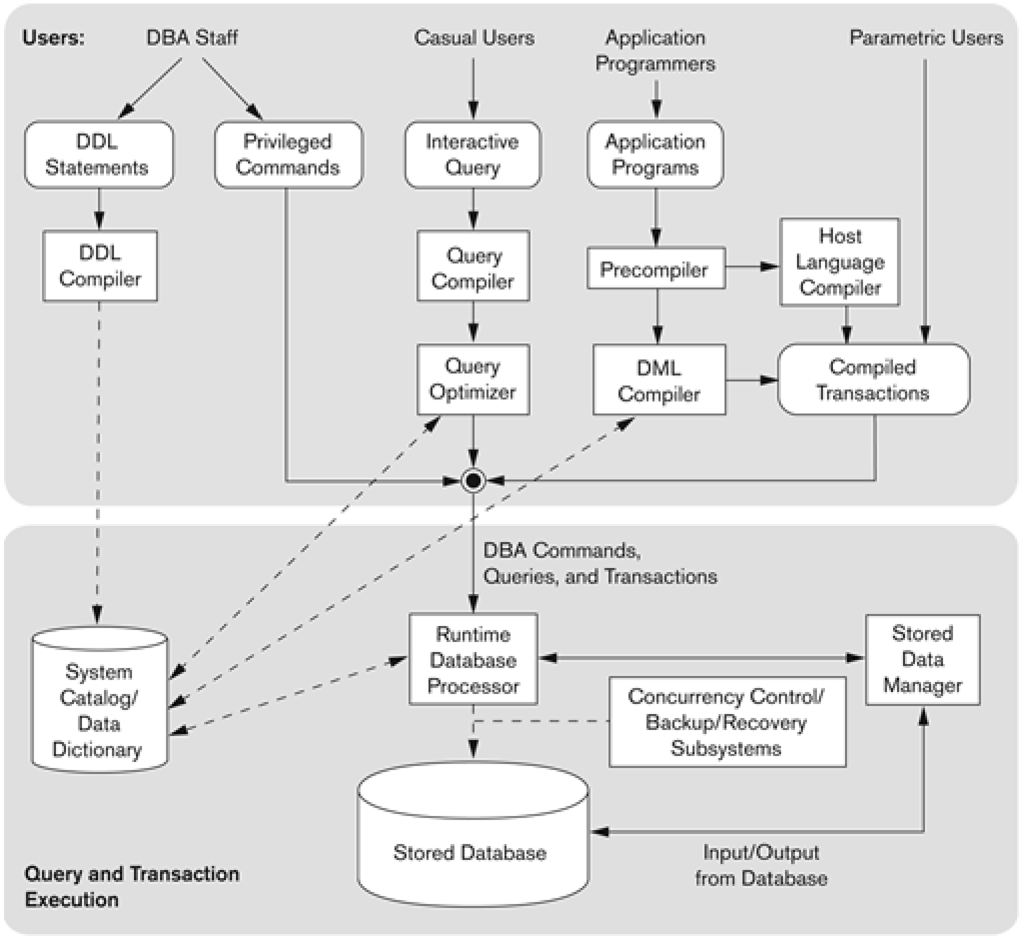
User-Friendly DBMS Interfaces

- Menu-based, popular for browsing on the web
- Forms-based, designed for naïve users
- Graphics-based
  - (Point and Click, Drag and Drop, etc.)
- Natural language: requests in written English
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

Other DBMS Interfaces

- Speech as Input and Output
- Web Browser as an interface
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

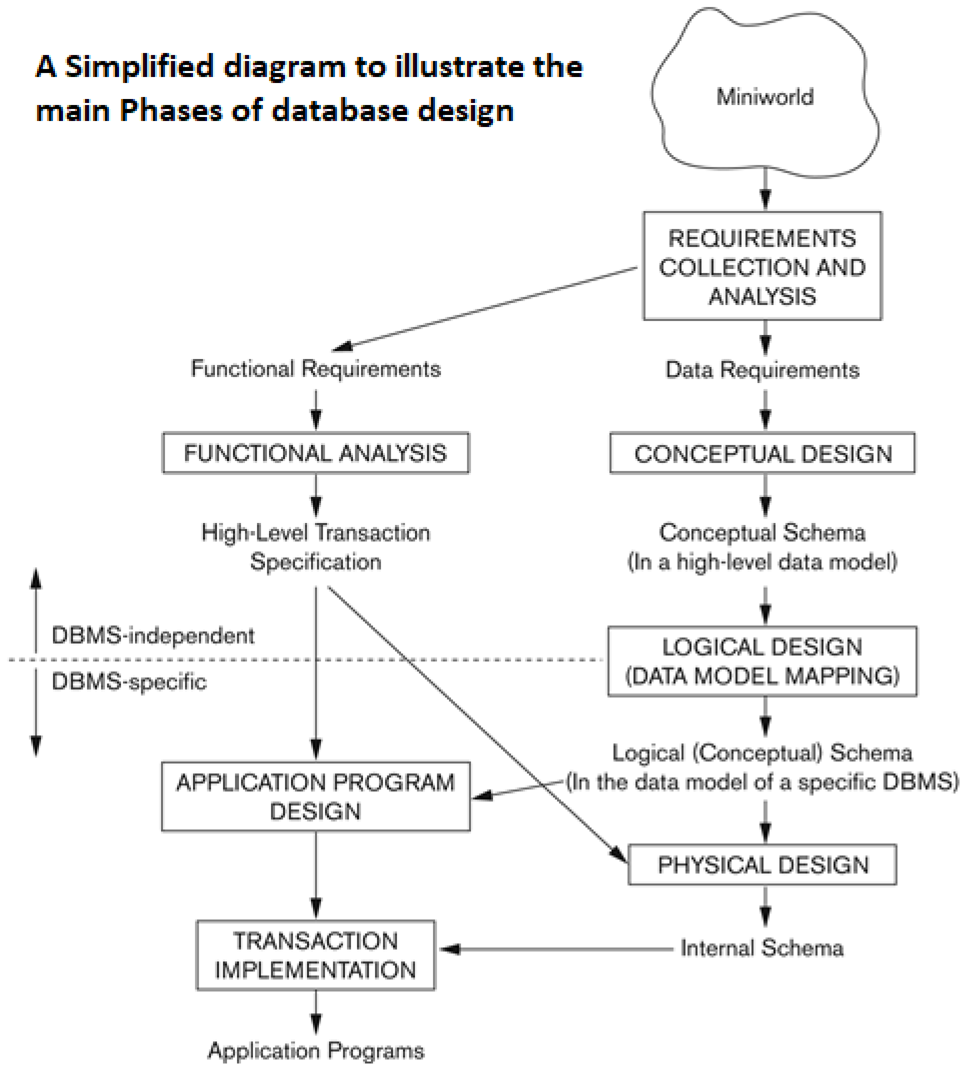
The Database System environment.



Component Modules of a DBMS and Their Interactions

Overview of Database Design Process

A Simplified diagram to illustrate the main Phases of database design


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
  - The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
  - Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

- 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* a number of DEPENDENTs.
  - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

ER Model Concepts

- Entities and Attributes
  - Entities are specific objects or things in the mini-world that are represented in the database.
    - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
  - Attributes are properties used to describe an entity.
    - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Sex, BirthDate
  - A specific entity will have a value for each of its attributes.
    - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘
  - Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, subrange, enumerated type, …

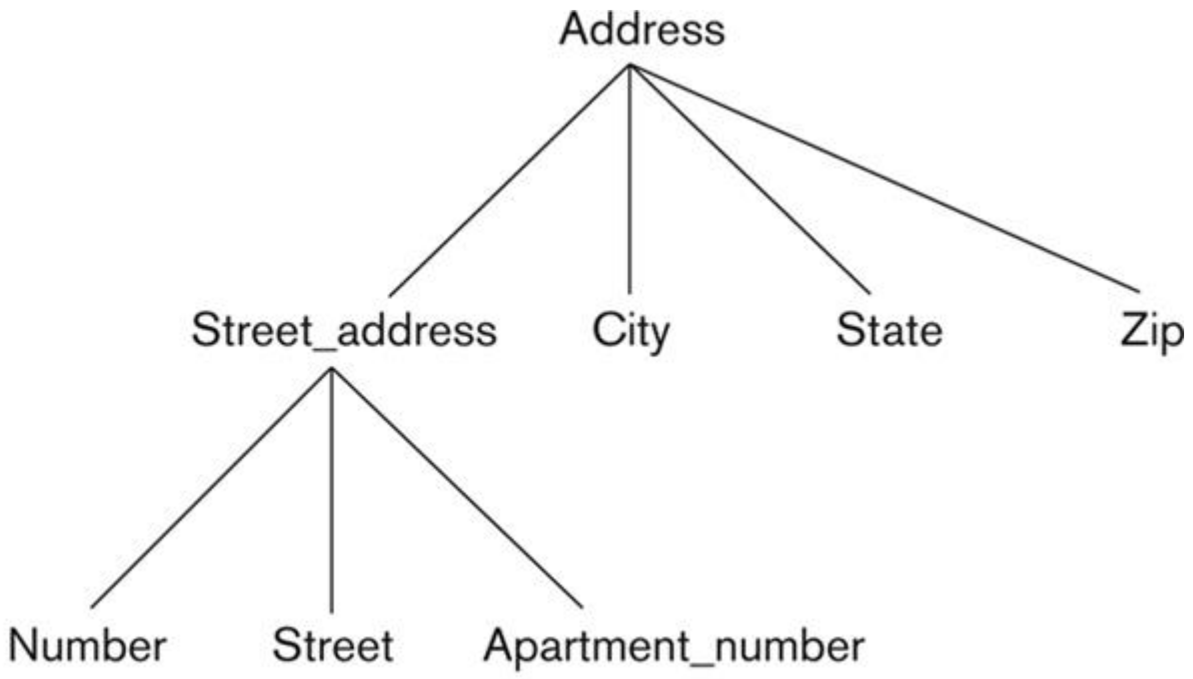
Types of Attributes (1)

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

Types of Attributes (2)

- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.
  - For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
  - Multiple PreviousDegrees values can exist
  - Each has four subcomponent attributes:
    - College, Year, Degree, Field

Example of a composite attribute



A hierarchy of composite attributes.

Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the entity type EMPLOYEE and PROJECT.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
  - For example, SSN of EMPLOYEE.

Entity Types and Key Attributes (Cont..)

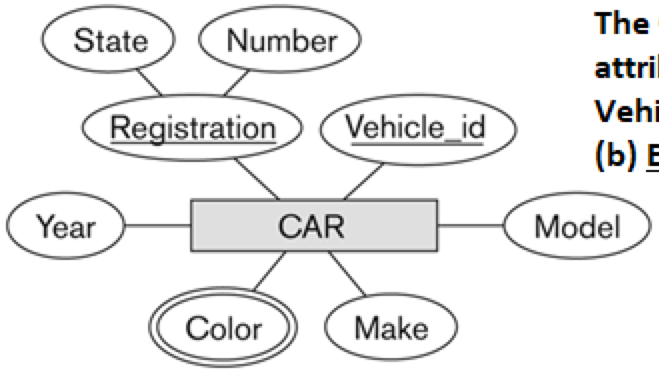
- A key attribute may be composite.
  - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State), aka license plate number.
- Each key is <u>underlined</u>

Displaying an Entity type

- In ER diagrams, an entity type is displayed in a rectangular box
- Attributes are displayed in ovals
  - Each attribute is connected to its entity type
  - Components of a composite attribute are connected to the oval representing the composite attribute
  - Each key attribute is underlined
  - Multivalued attributes displayed in double ovals
- See CAR example on next slide

Entity Type CAR with two keys and a corresponding Entity Set

(a)



The CAR entity type with two key attributes, Registration and Vehicle_id (a) ER Diagram
(b) Entity Set with three entities

(b)

CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR$_1$
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR$_2$
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR$_3$
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

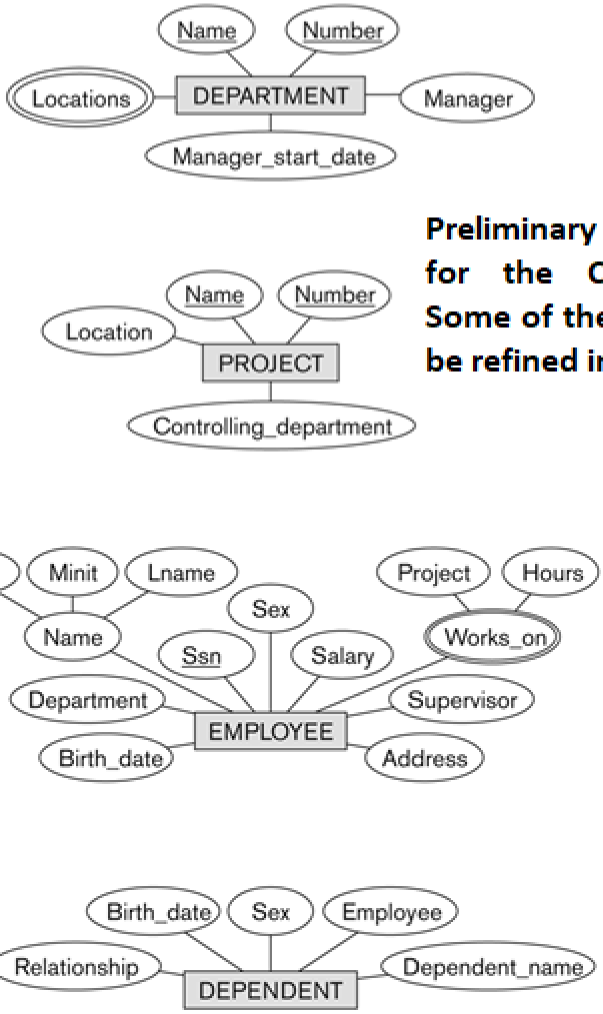
Entity Set

- Each entity type will have a collection of entities stored in the database
    - Called the **entity set**
- Previous slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- Entity set is the current *state* of the entities of that type that are stored in the database

Initial Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- Their initial design is shown on the following slide
- The initial attributes shown are derived from the requirements description

Initial Design of Entity Types:
EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT



Preliminary design of entity types for the COMPANY database. Some of the shown aatibutes will be refined into relationships

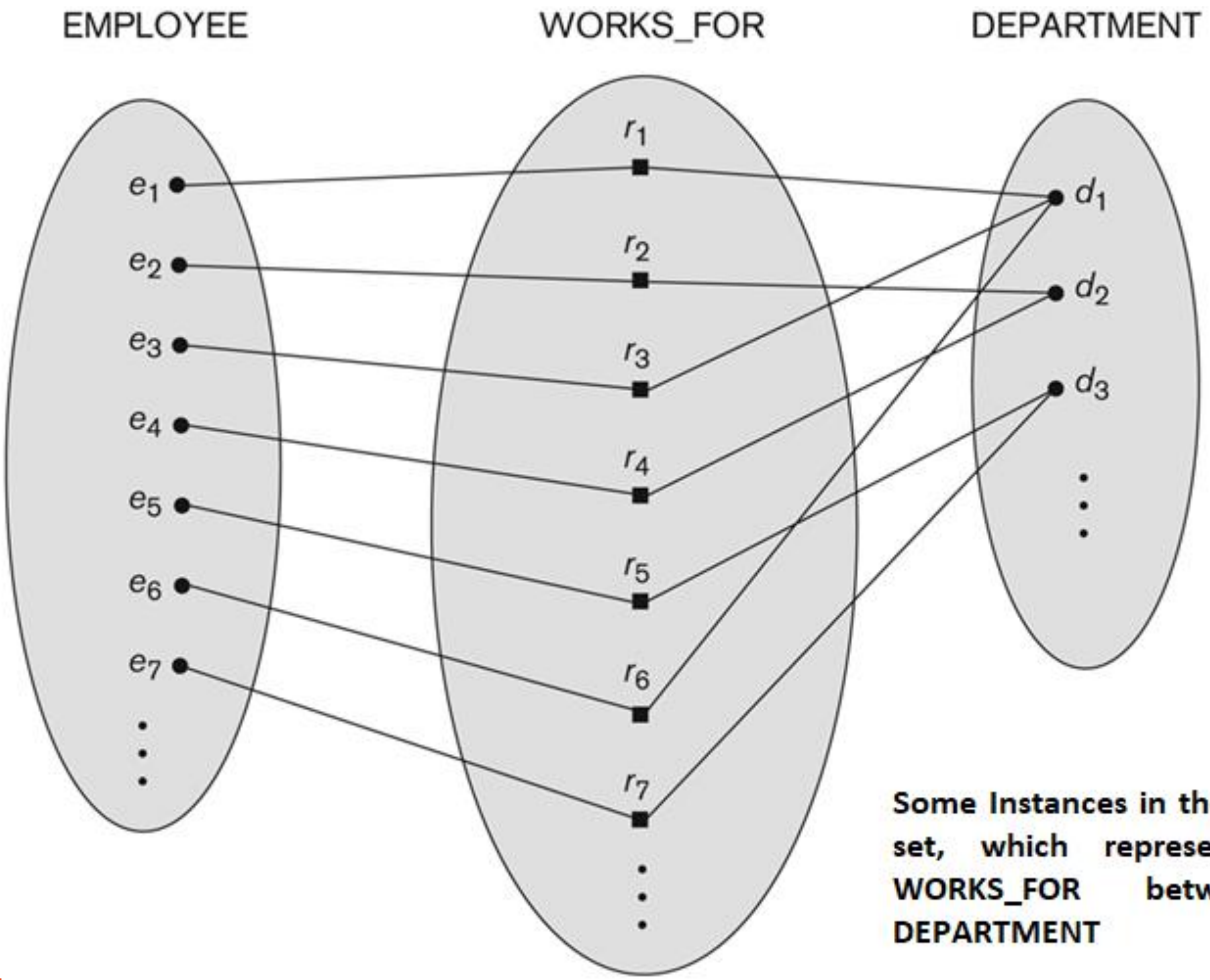
Refining the initial design by introducing **relationships**

- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - Entities (and their entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (and their relationship types and relationship sets)
- We introduce relationship concepts next

Relationships and Relationship Types

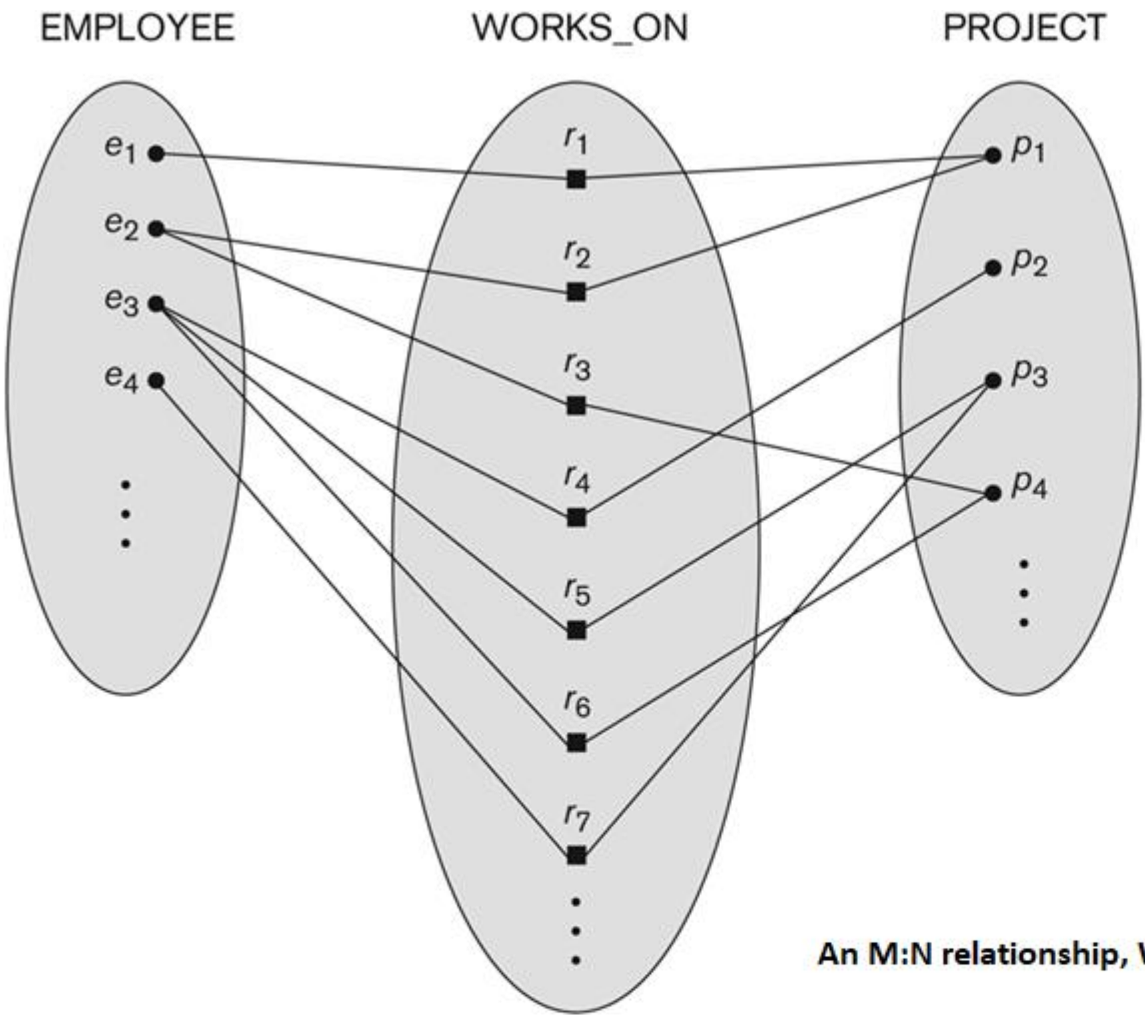
- A **relationship** relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith *works on* the ProductX PROJECT, or EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a **relationship type**.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types.
  - Both MANAGES and WORKS_ON are *binary* relationships.

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT



Some Instances in the WORKS_FOR Relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT



An M:N relationship, WORKS_ON

Relationship type vs. relationship set

- Relationship Type:
  - Is the schema description of a relationship
  - Identifies the relationship name and the participating entity types
  - Also identifies certain relationship constraints
- Relationship Set:
  - The current set of relationship instances represented in the database
  - The current *state* of a relationship type

Relationship type vs. relationship set

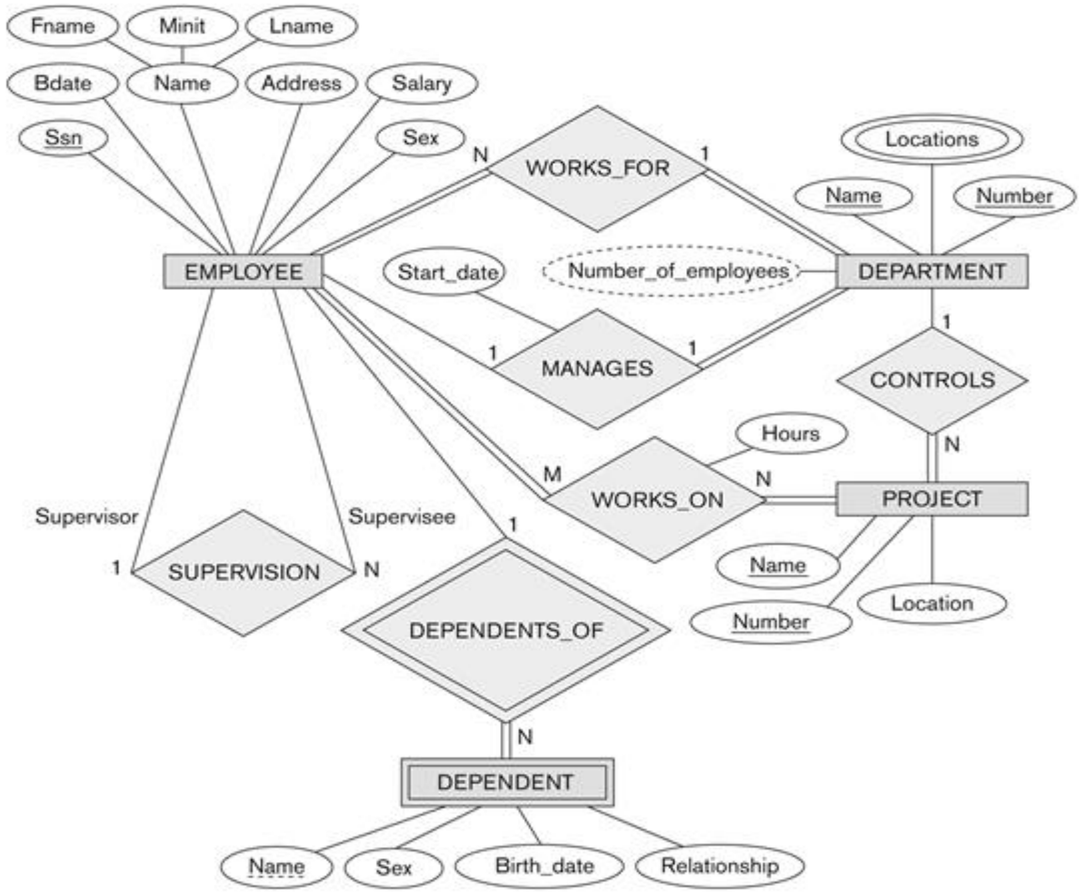
- Previous figures displayed the relationship sets
- Each instance in the set relates individual participating entities – one from each participating entity type
- In ER diagrams, we represent the *relationship type* as follows:
  - Diamond-shaped box is used to display a relationship type
  - Connected to the participating entity types via straight lines

Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships(degree 2)
- Listed below with their participating entity types:
  - WORKS_FOR (between EMPLOYEE, DEPARTMENT)
  - MANAGES (also between EMPLOYEE, DEPARTMENT)
  - CONTROLS (between DEPARTMENT, PROJECT)
  - WORKS_ON (between EMPLOYEE, PROJECT)
  - SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
  - DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF



An ER Schema Diagram for the COMPANY database.

Recursive Relationship Type

- An relationship type whose with the same participating entity type in **distinct roles**
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
  - supervisor (or boss) role
  - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
  - One employee in *supervisor* role
  - One employee in *supervisee* role

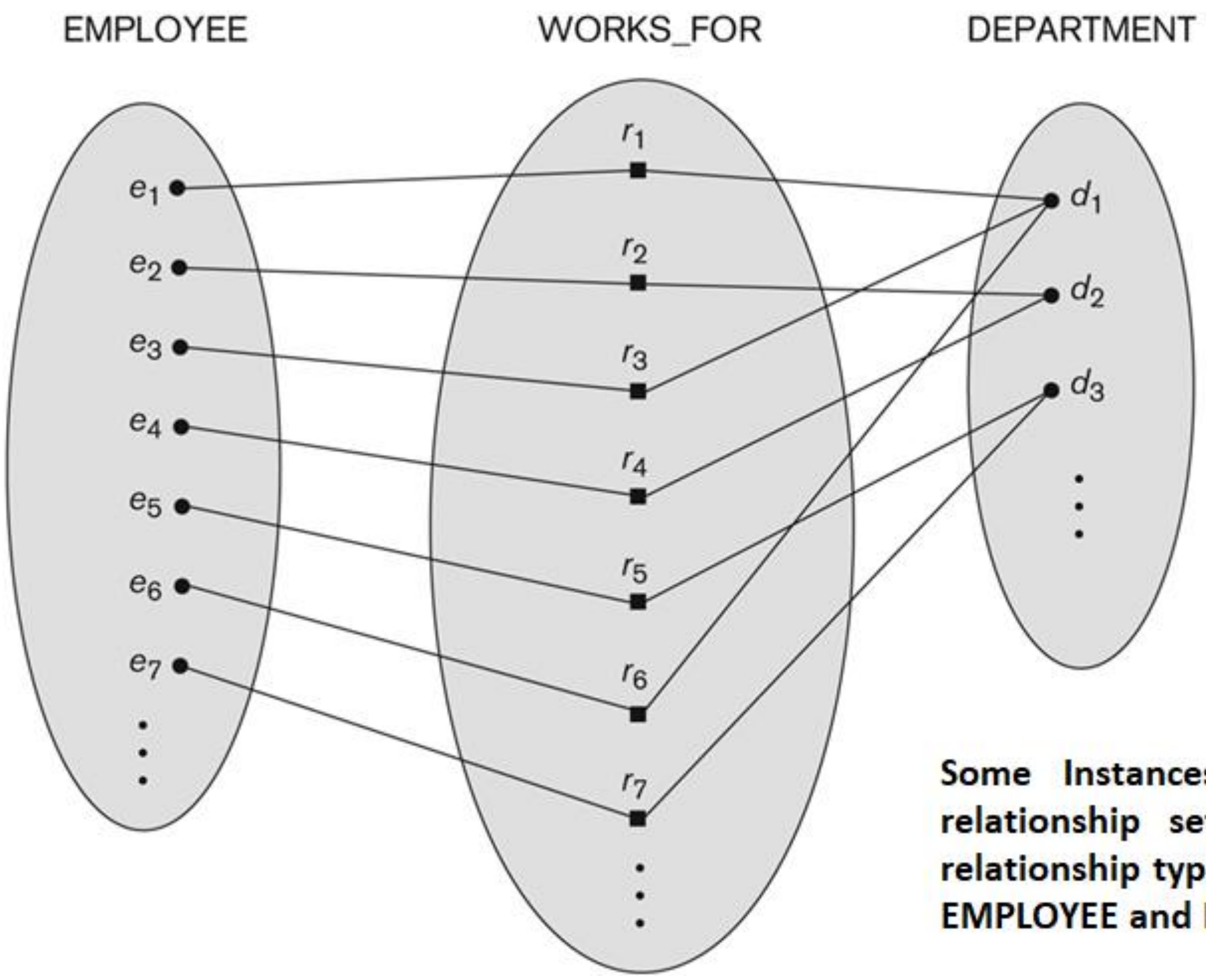
Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type
- **Example:**
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Constraints on Relationships

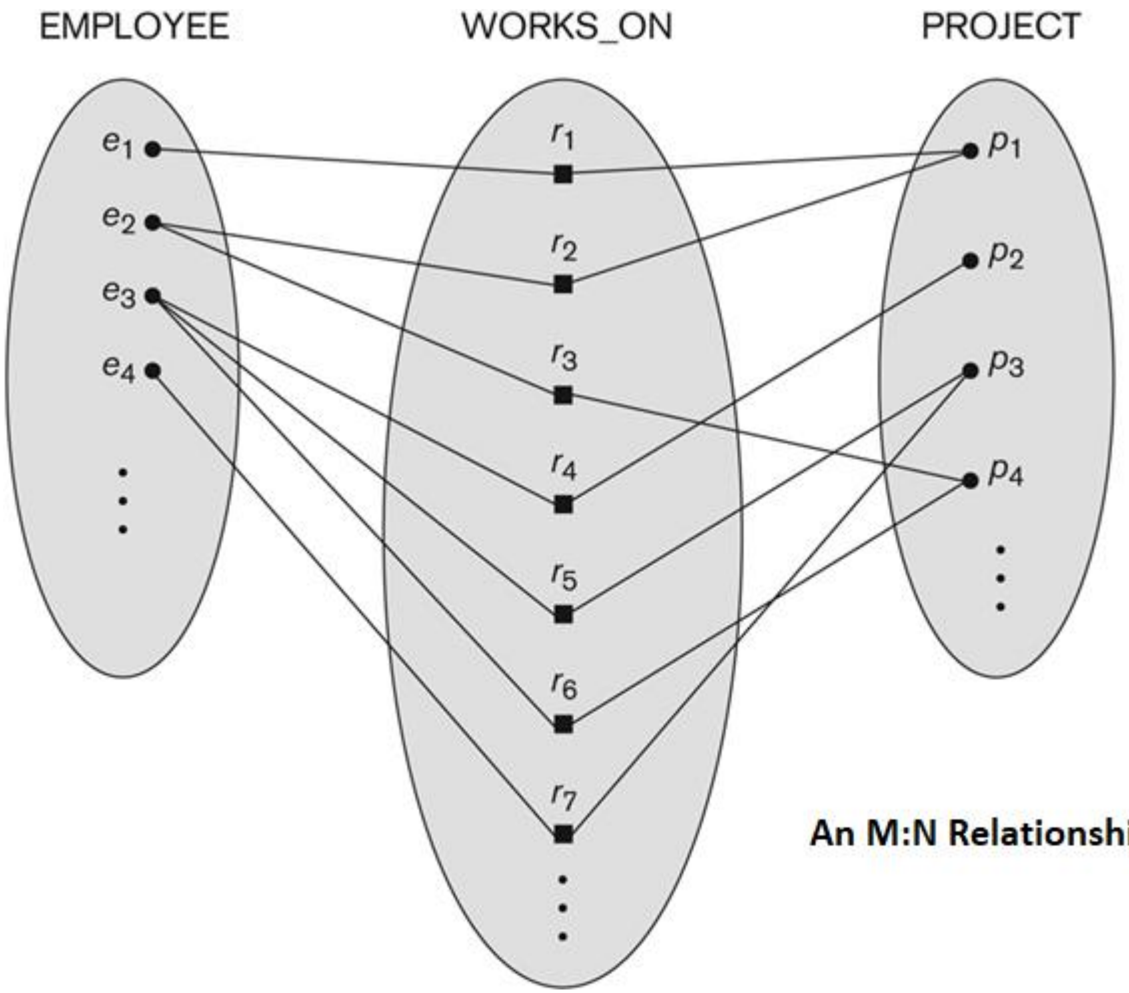
- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Cardinality Ratio (specifies *maximum* participation)
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many (M:N)
  - Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory participation, existence-dependent)

Many-to-one (N:1) Relationship



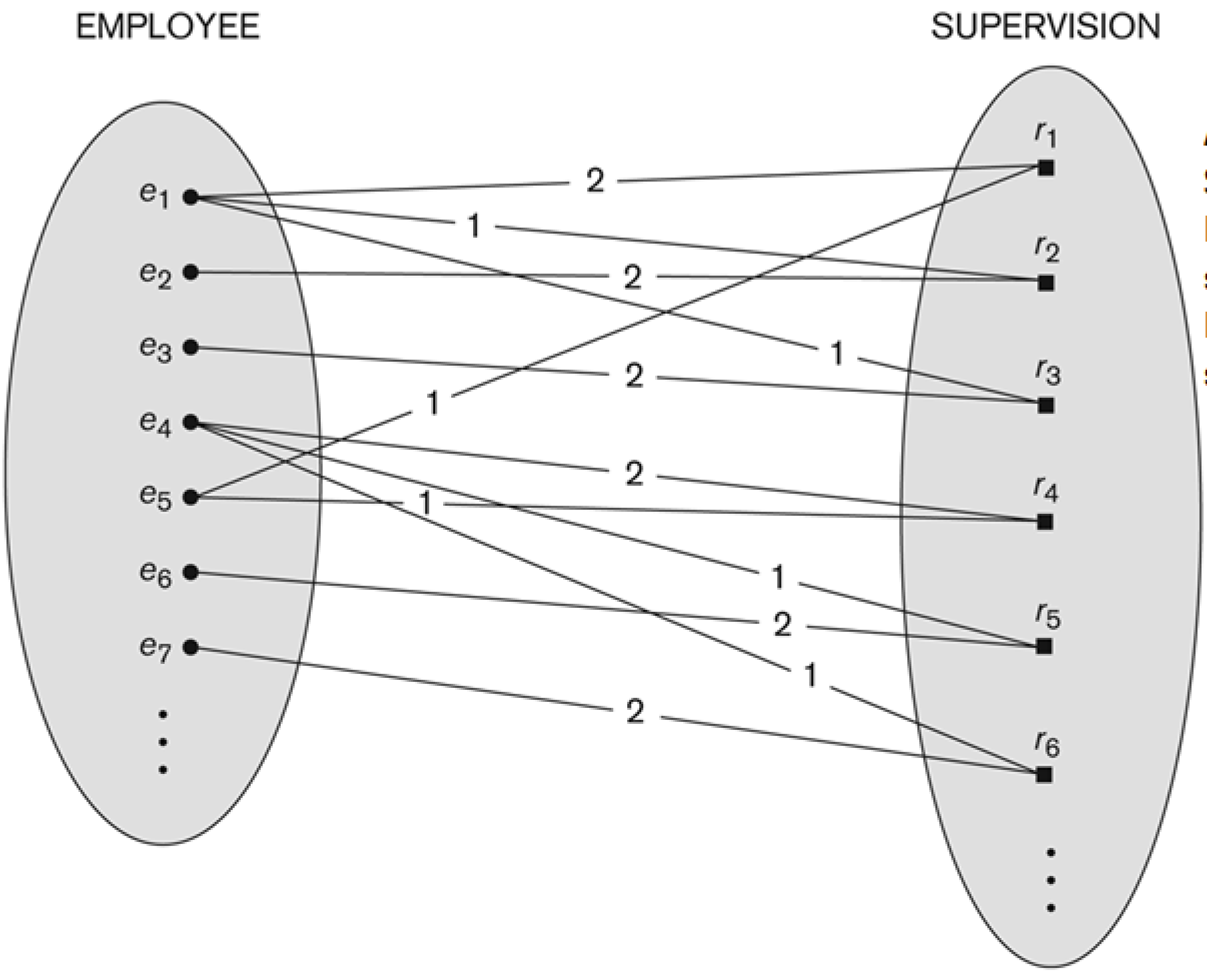
Some Instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT

Many-to-many (M:N) Relationship



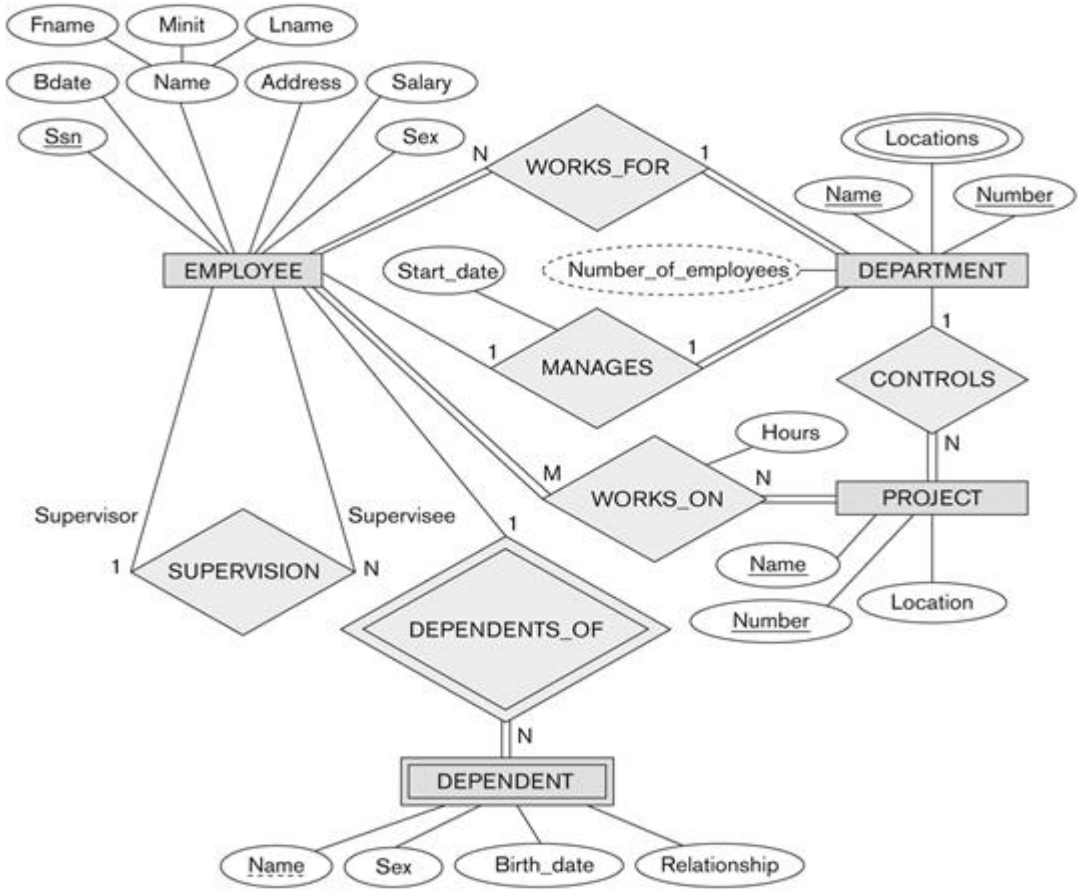
An M:N Relationship, WORKS_ON

A Recursive Relationship Supervision`



A Recursive relationship SUPERVISION between EMPLOYEE in the supervisor role(1) and EMPLOYEE in the subordinate role(2)

Recursive Relationship Type is: SUPERVISION (participation role names are shown)



An ER Schema Diagram for the COMPANY database.

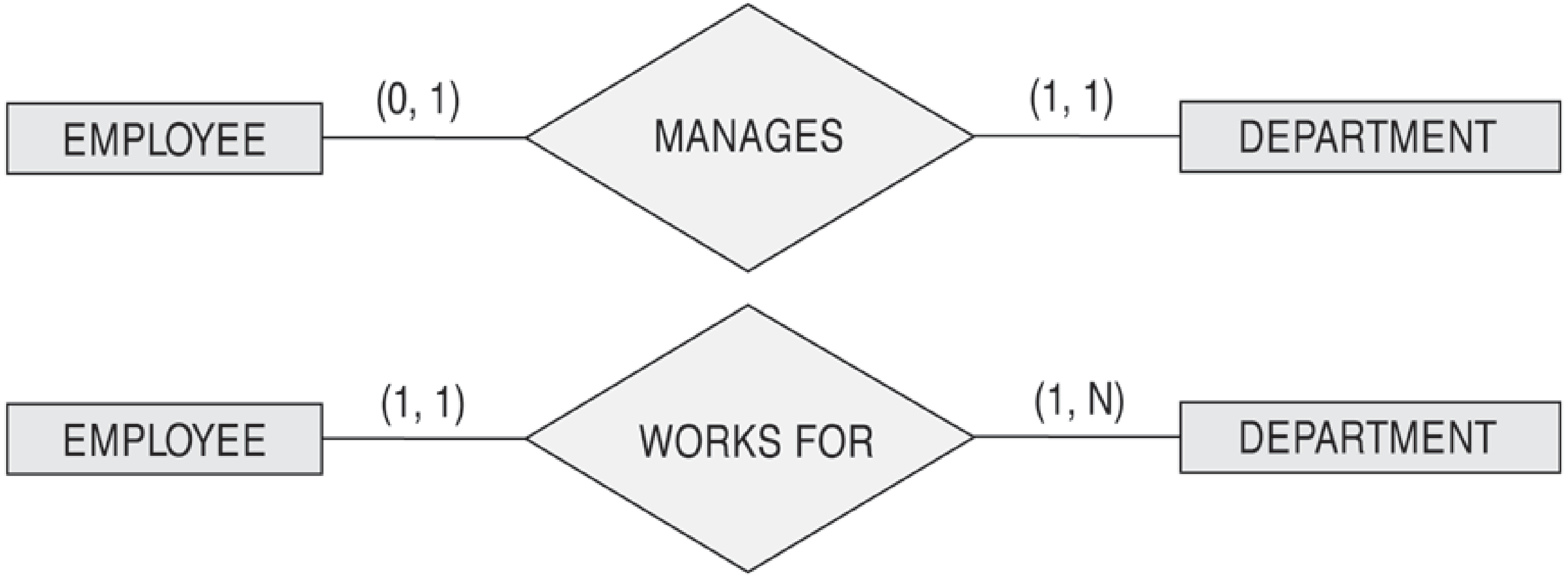
Notation for Constraints on Relationships

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): total (called existence dependency) or partial.
  - Total shown by double line, partial by single line.
- NOTE: These are easy to specify for Binary Relationship Types.

Alternative (min, max) notation for relationship structural constraints:

- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
- Default(no constraint): min=0, max=n (signifying no limit)
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints
- Examples:
  - A department has exactly one manager and an employee can manage at most one department.
    - Specify (0,1) for participation of EMPLOYEE in MANAGES
    - Specify (1,1) for participation of DEPARTMENT in MANAGES
  - An employee can work for exactly one department but a department can have any number of employees.
    - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
    - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

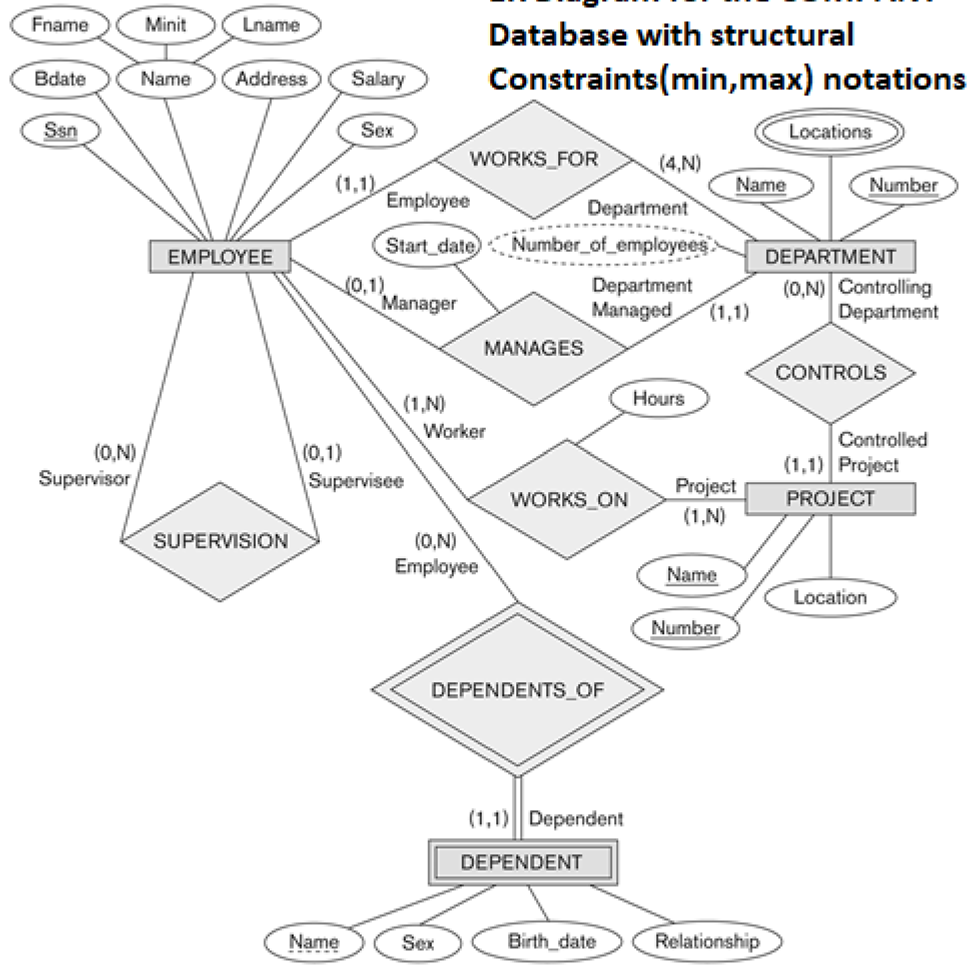
The (min,max) notation for relationship constraints



Read the min,max numbers next to the entity type and looking **away from** the entity type

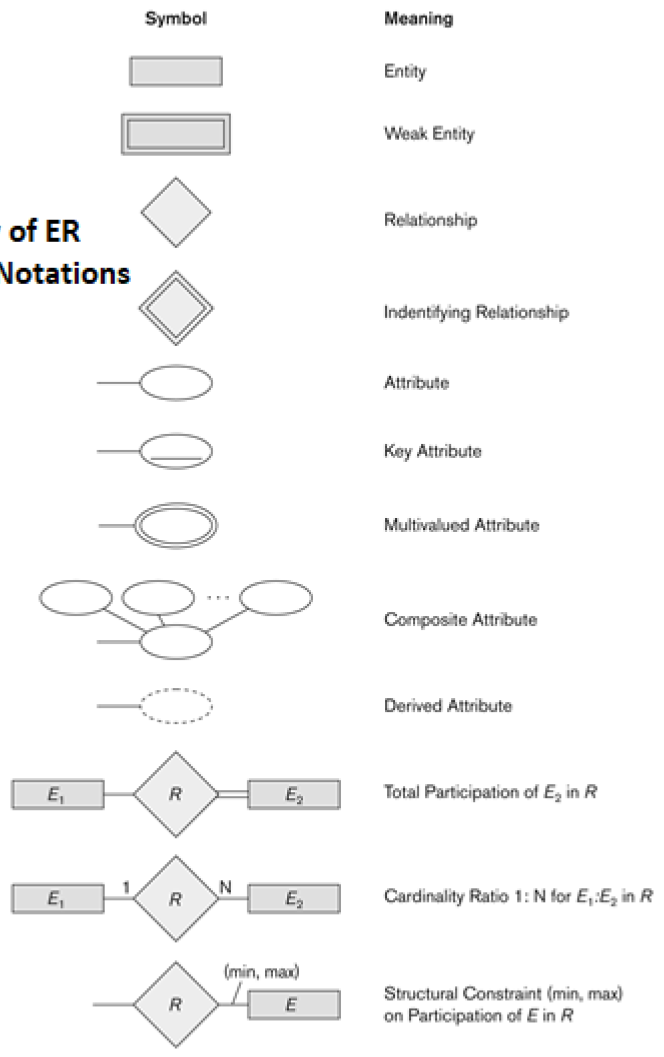
COMPANY ER Schema Diagram using (min, max) notation

ER Diagram for the COMPANY Database with structural Constraints(min,max) notations



Summary of notation for ER diagrams

Summary of ER Diagram Notations


Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
. . .
Composite Attribute
Derived Attribute
E_1
R
E_2
Total Participation of E_2 in R
E_1
1
R
N
E_2
Cardinality Ratio 1: N for E_1:E_2 in R
R
(min, max)
E
Structural Constraint (min, max) on Participation of E in R

Conclusions

We have understood the below concepts:

1. Why Data Analytics, What Data Analytics
2. Who is Data Analyst, Data Analyst skill set
3. Statistics and Types
4. Data Cleaning and Data Manipulation
5. Data Visualization
6. Bonus: Machine Learning
7. Analytics Life Cycle
8. Roles of Data Analyst and Salary Ranges

Important Questions asked in VTU Semester End Exam (SEE)

Application Oriented Questions

GATE Questions

Market Share of different Databases

Contact Details:

Dr.Manjunath T N
Professor and Dean – ER
Department of Information Science and Engg
BMS Institute of Technology and Management
Mobile: +91-9900130748
E-Mail: manju.tn@bmsit.in / dean_externalrelations@bmsit.in