

# **Data Modeling using E.R. Model (Entity Relationship Model) & Relational Database**



**Course:-BSC CS-II**

**Subject:-Database Management System**

**Unit:-3**

# Entity-Relationship(ER) Model

- The ER model is a high-level conceptual data model. It has not been implemented in any commercial DBMS (?), but is a powerful short hand often used in database design for a first rendition of the miniworld.
- The ER model was introduced by Peter Chen in 1976, and is now the most widely used conceptual data model.

# Entity-Relationship(ER) Model[1]



# Definitions

- An entity is an object in the miniworld.
- An attribute of an entity can have a value from a value set (domain)
- Each entity belongs to some one entity type s.t. entities in one entity type have the same attributes (so each entity type is a set of similar entities).

# Definitions

- A key attribute of an entity type is one whose value uniquely identifies an entity of that type.
- A combination of attributes may form a composite key.
- If there is no applicable value for an attribute that attribute is set to a null value.

# Entity Type / Entity Set

Entity Type (Intension):

EMPLOYEE

Attributes:

Name, Age, Salary

Entity Set (Extension):

$e_1 = (\text{John Smith}, 55, 80000)$

$e_2 = (\text{Joe Doe}, 40, 20000)$

$e_3 = (\text{Jane Doe}, 27, 30000)$

.

.

.

# Attributes

- Attributes can be
  - composite / simple (atomic)
  - single-valued / multivalued
  - stored / derived
  - key / nonkey.

# Attribute Examples

Name = John Doe

Birth date = May 10, 1989

Age = 9

Degree = null

SSN = 123456789

Name = Jane Doe

Birthrate = July 11, 1960

Age = 38

Degree = B.S., M.S.

SSN = 987654321

Name = John Doe

Birth date = May 10

Birth year = 1989

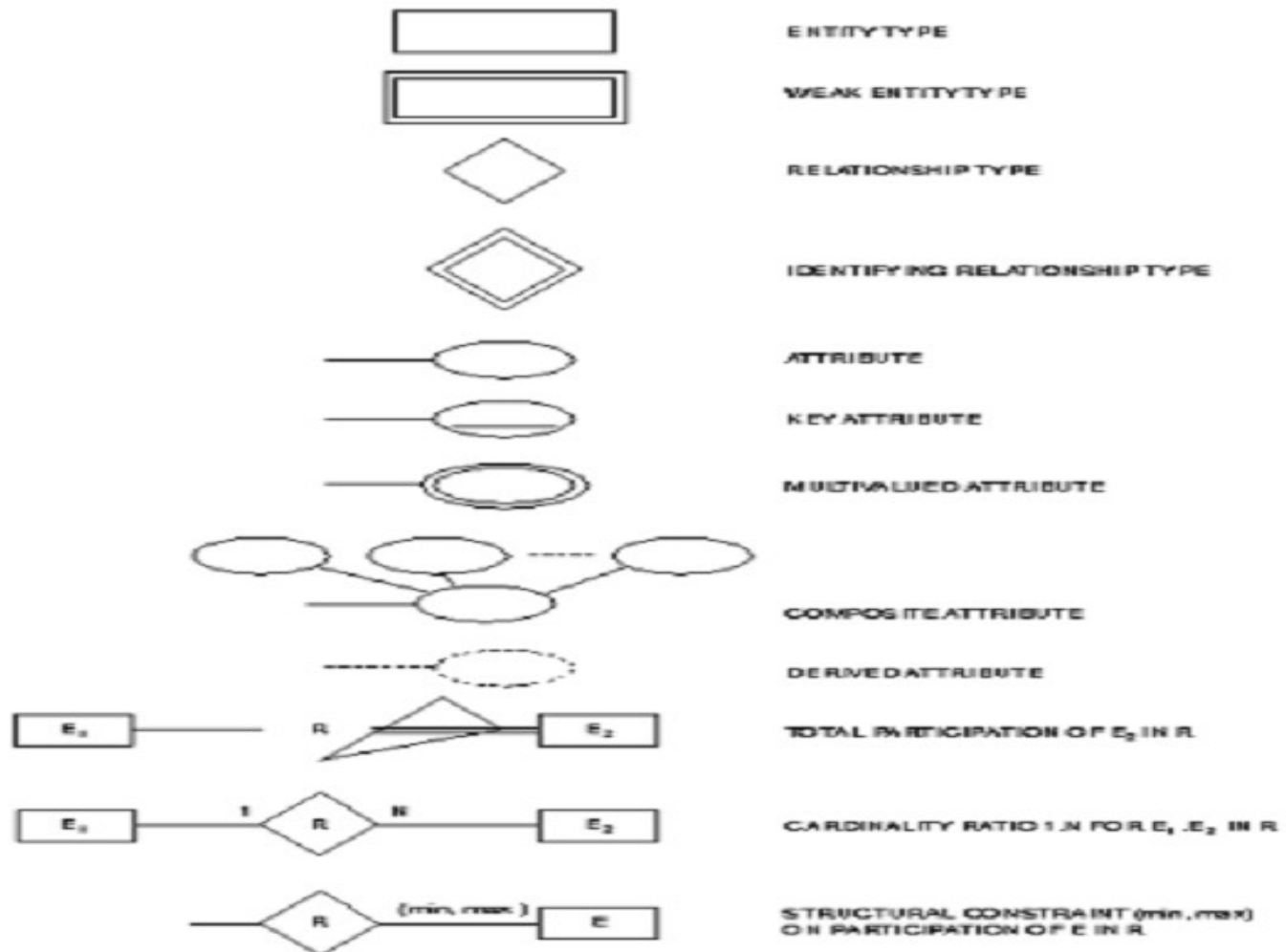
Age = 9

Degree = null

SSN = 123456789

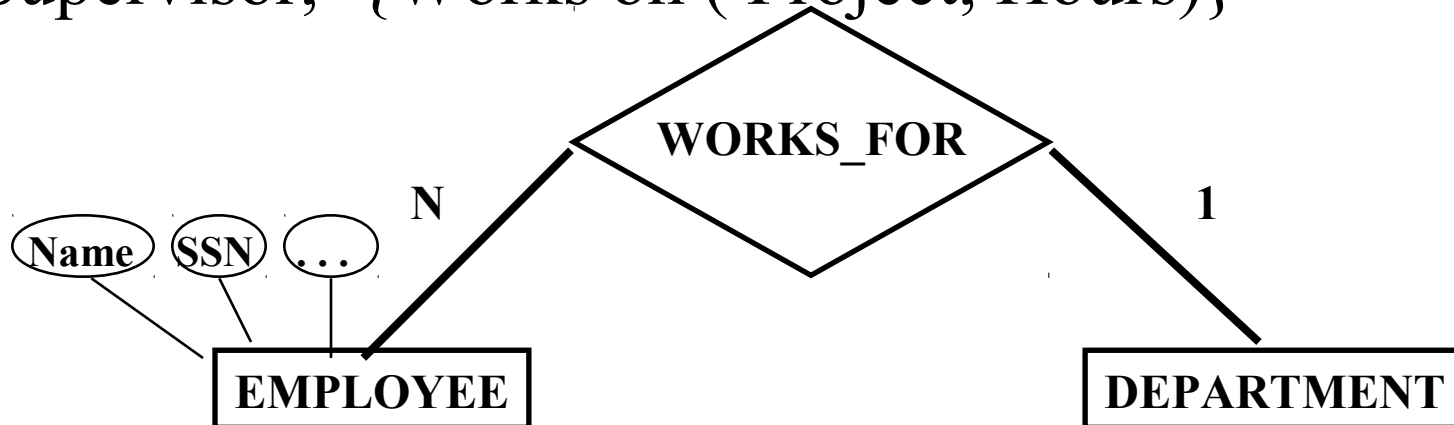


# Symbols



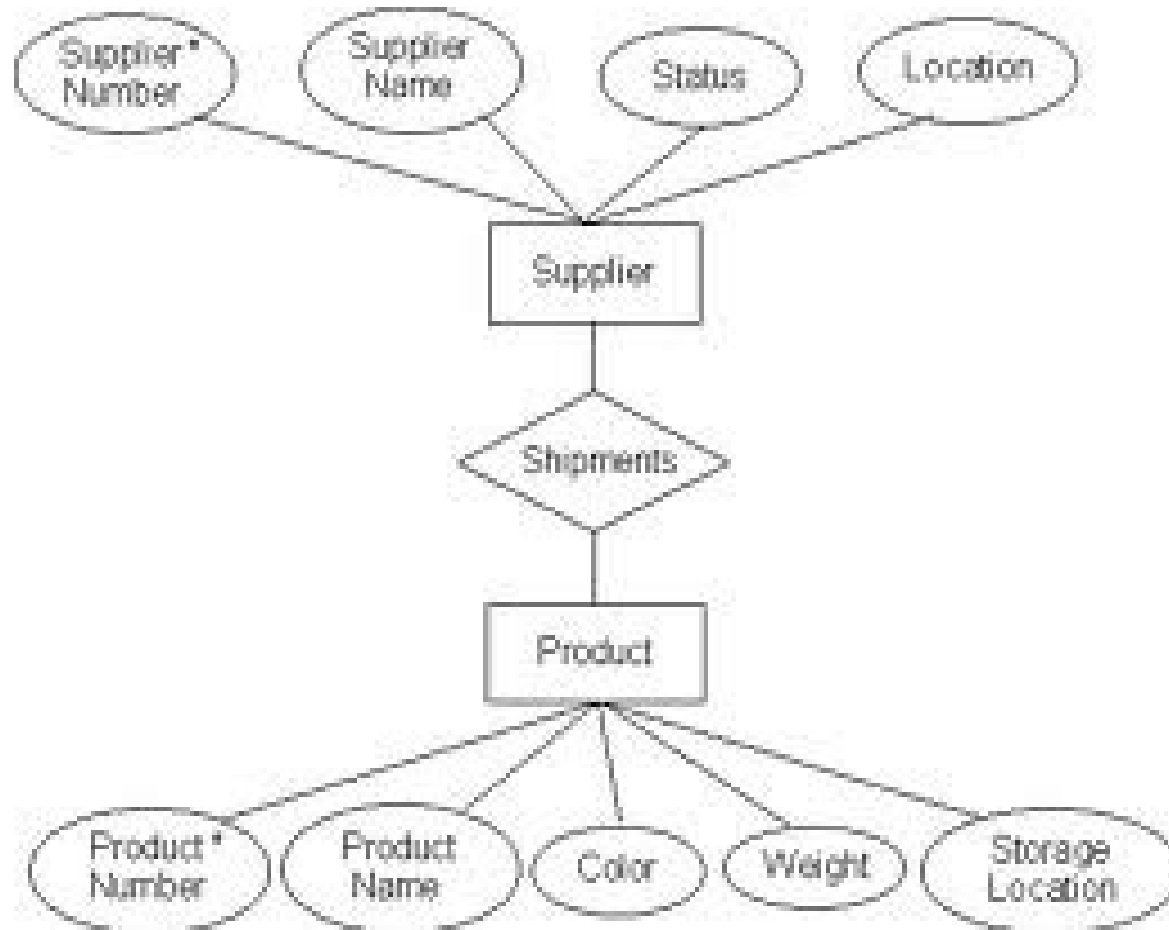
# EMPLOYEE

Name, SSN, Sex, Address, Salary, Birth date, Department,  
Supervisor, {Works on ( Project, Hours)}



Relationship instances of **WORKS\_FOR**:  
 $\{(KV, CS), (Pan, EE), \dots\}$

# ER Diagram for COMPANY Database[2]



# Relationship Type

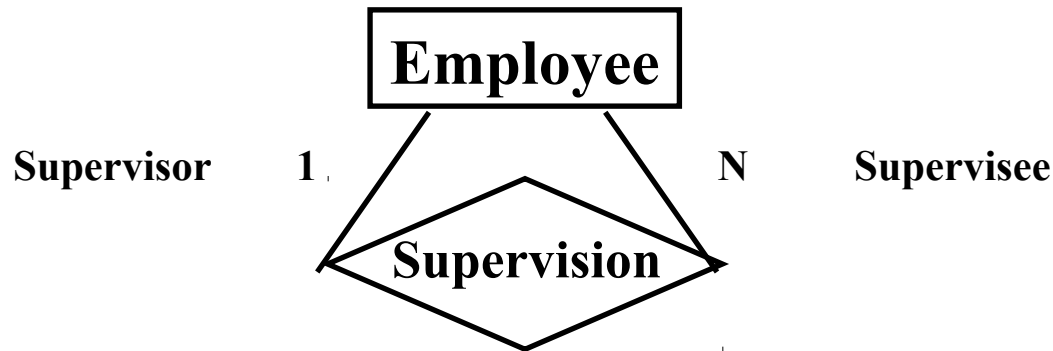
- A relationship type  $R$  among  $n$  entity types  $E_1, \dots, E_n$  is a set of relationship instances  $r_i$ , where each  $r_i$  associates  $n$  entities  $(e_1, \dots, e_n)$ , s.t. each  $e_j \hat{=} E_j$ . Informally, a relationship instance is an association of entities, with exactly one entity from each participating entity type.

# Relationship Type

- The degree  $n$  of a relationship type is the number of participating entity types.
- In the ER model relationships are **explicitly** represented.

# Entity Roles

- Each entity type in a relationship type plays a particular role that is described by a role name. Role names are especially important in recursive relationship types where the same entity participates in more than one role:



# Weak Entity Type

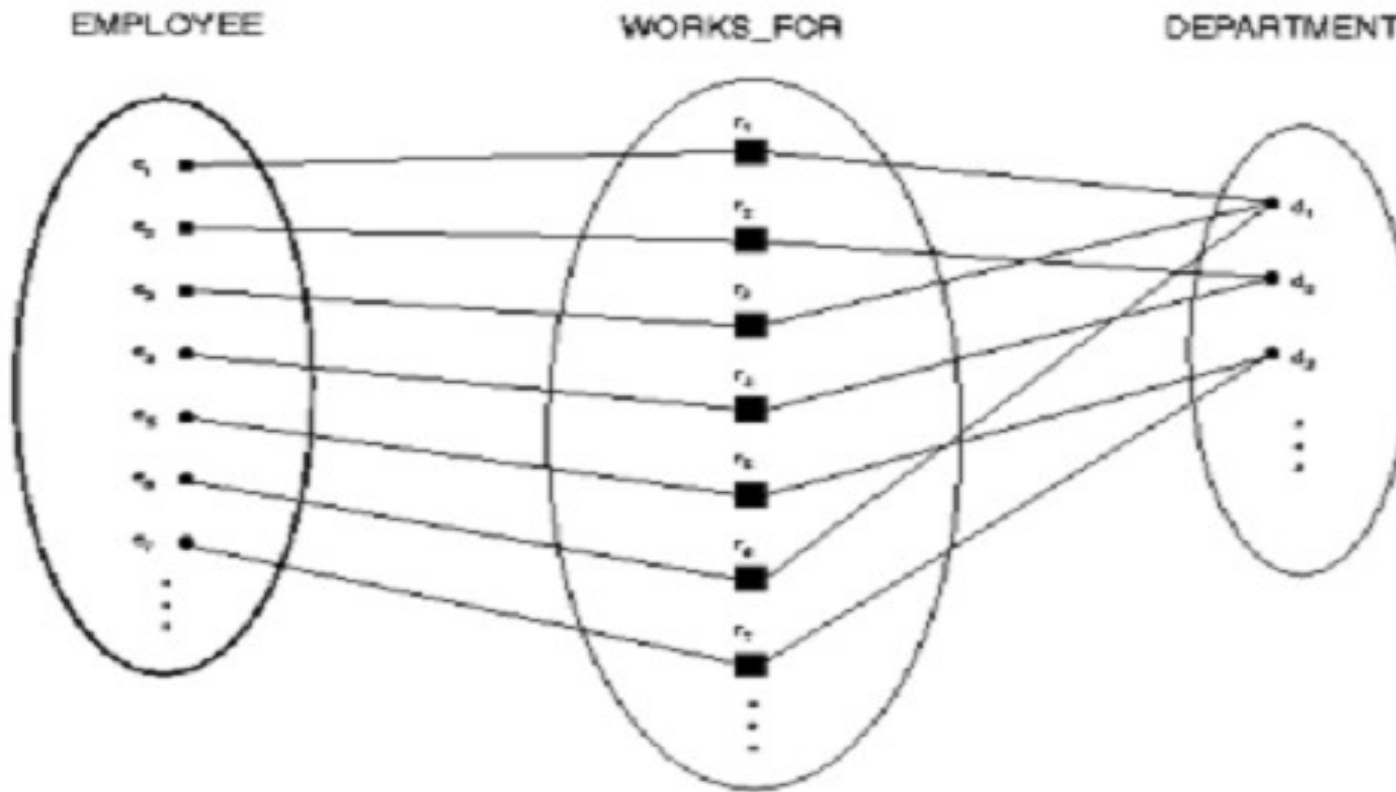
- A weak entity type is one **without** any key attributes of its own. Entities belonging to a weak entity type are identified by being related to another entity type ( called identifying owner) through a relationship type ( called identifying relationship), in combination with values of a set of its own attributes (called partial key). A weak entity type has total participation constraint w.r.t. its identifying relationship.

# Relationship Attributes

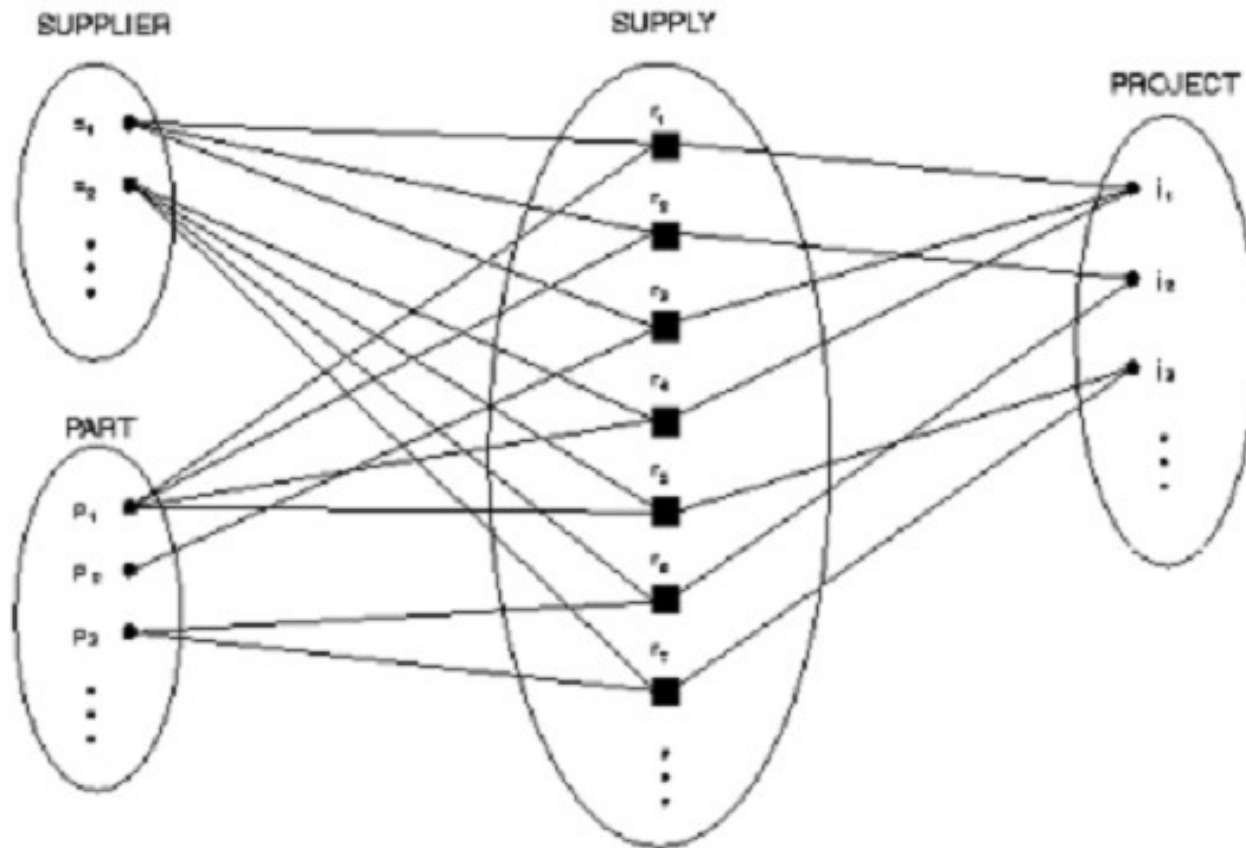
- Relationship types can have attributes as well. in case of 1:1 or 1:N relationships, attributes can be migrated to one of the participating entity types.



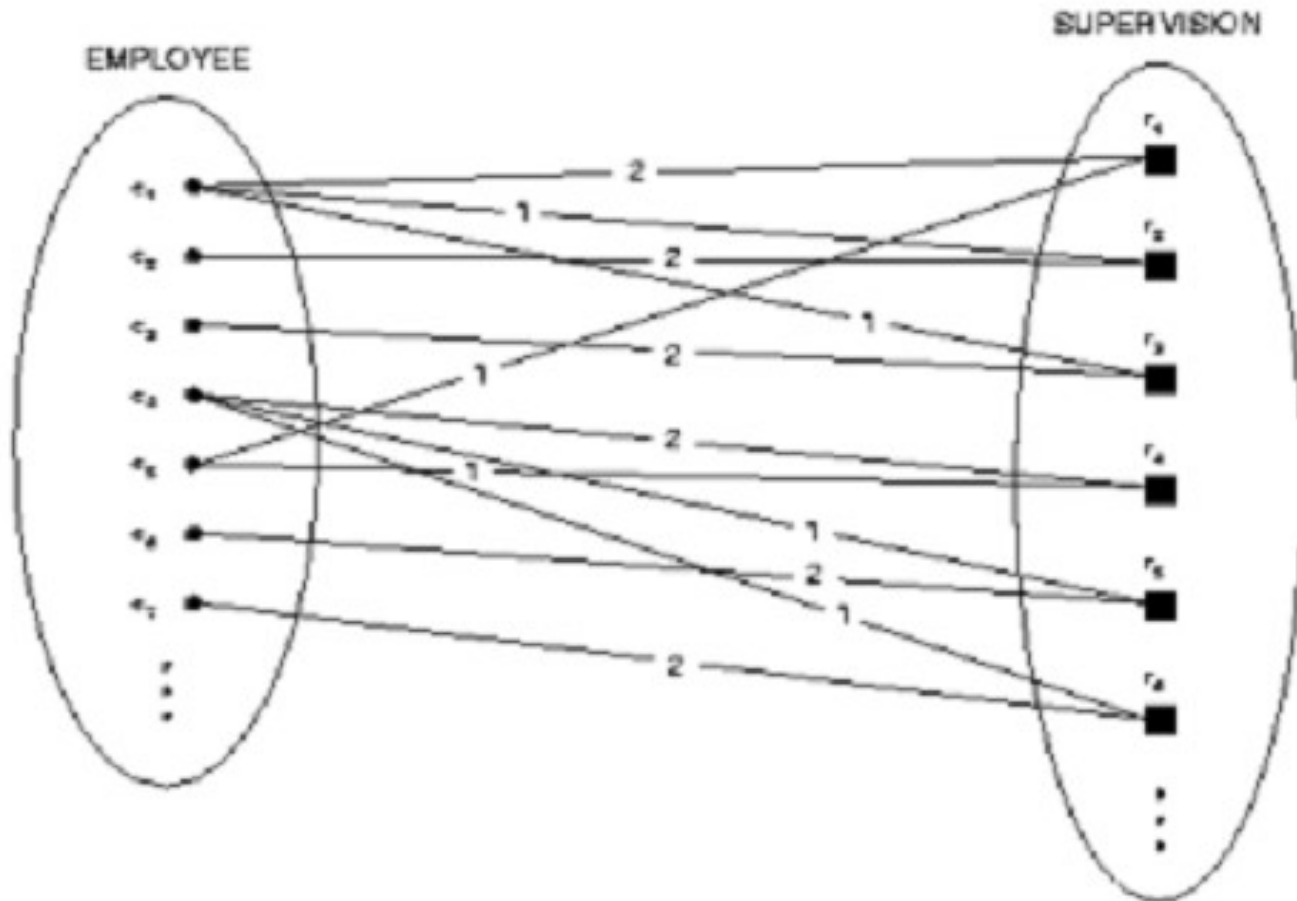
# Relationship[3]



# Relationship[4]



# Relationship[5]



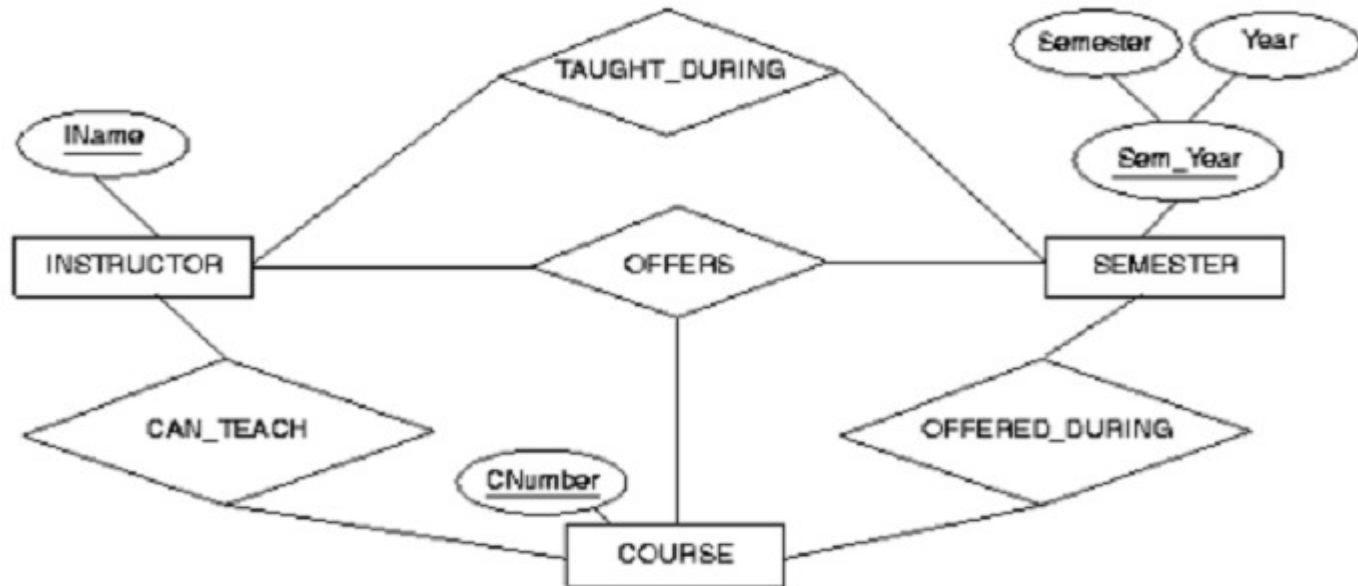
# Structural Constraints

- Structural constraints of a relationship type:
  - **Cardinality ratio:** Limits the number of relationship instances an entity can participate in, eg. **1:1**, **1:N**, **M:N**
  - **Participation constraint:** If each entity of an entity type is **required** to participate in some instance of a relationship type, then that participation is **total**; otherwise, it is **partial**.

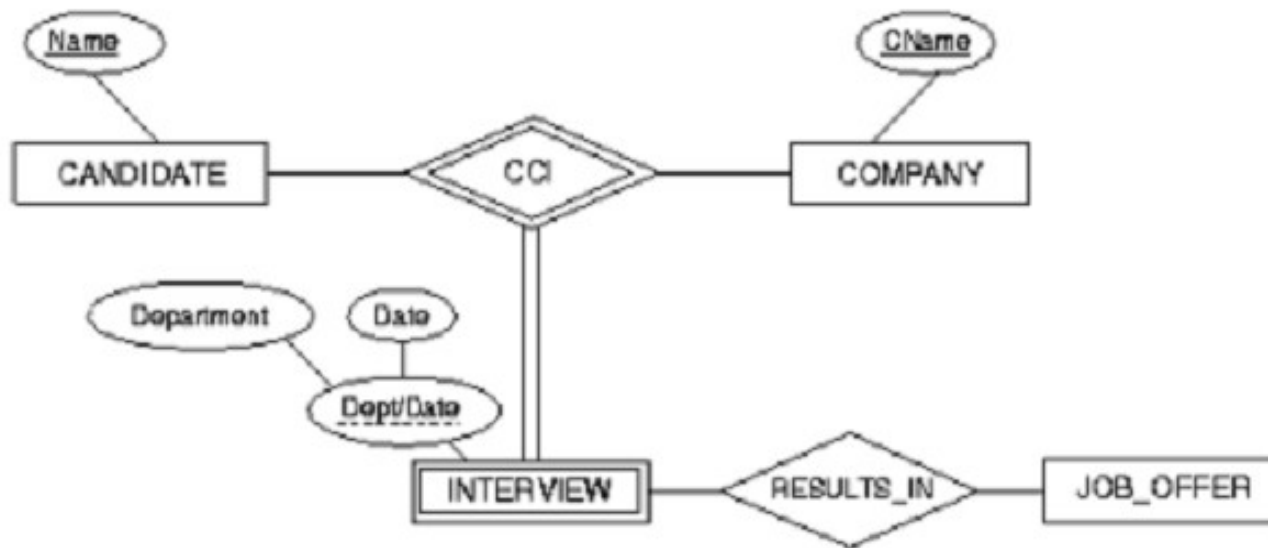
# Structural Constraint Min, Max

- A more complete specification of the structural constraint on a relationship type can be given by the integer pair (min, max), which means an entity must participate in at least min and at most max relationship instances.

A ternary relationship generally represents more information than 3 binary relationships[6]



# A Weak Entity with a Ternary Identifying Relationship[7]



# Introduction

- Database – collection of persistent data
- Database Management System (DBMS) – software system that supports creation, population, and querying of a database



# Relational Database

- Relational Database Management System (RDBMS)
- Consists of a number of *tables* and single *schema* (definition of tables and attributes)
- Students (Sid, name, login, age, gpa)
- **Students** identifies the table
- **Sid, name, login, age, gpa** identify attributes
- **Sid** is primary key

# An Example Table

- Students (Sid: string, name: string, login: string, age: integer, gpa: real)

<u>sid</u>	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

## Another example: Courses

- Courses (cid, instructor, quarter, dept)

<u>cid</u>	instructor	quarter	dept
Carnatic101	Jane	Fall 06	Music
Reggae203	Bob	Summer 06	Music
Topology101	Mary	Spring 06	Math
History105	Alice	Fall 06	History

# Keys

- Primary key – minimal subset of fields that is unique identifier for a tuple
- Sid is primary key for Students
- cid is primary key for Courses
- Foreign key –connections between tables
- Courses (cid, instructor, quarter, dept)
- Students (Sid, name, login, age, gpa)
- How do we express which students take each course?

# Many to many relationships

- In general, need a new table

Enrolled(cid, grade, studid)

Studid is foreign key that references Sid in Student table

Enrolled			Student		
		Foreign key	<u>sid</u>	name	login
<u>cid</u>	grade	studid	50000	Dave	dave@cs
Carnatic101	C	53831	53666	Jones	jones@cs
Reggae203	B	53832	53688	Smith	smith@ee
Topology112	A	53650	53650	Smith	smith@math
History 105	B	53666	53831	Madayan	madayan@mus
			53832	Guldu	guldu@music

# Relational Algebra

- Collection of operators for specifying queries
- Query describes step-by-step procedure for computing answer (i.e., *operational*)
- Each operator accepts one or two relations as input and returns a relation as output
- Relational algebra expression composed of multiple operators

# Basic operators

- Selection – return *rows* that meet some condition
- Projection – return *column* values
- Union
- Cross product
- Difference
- Other operators can be defined in terms of basic operators

# Example Schema (simplified)

- Courses (cid, instructor, quarter, dept)
- Students (Sid, name, gpa)
- Enrolled (cid, grade, studid)



# Selection

Select students with gpa higher than 3.3 from S1:

$\sigma_{gpa > 3.3}(S1)$

**S1**

sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0



sid	name	gpa
53666	Jones	3.4
53650	Smith	3.8

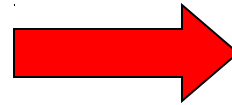
# Projection

Project name and gpa of all students in S1:

$\Pi_{\text{name, gpa}}(\text{S1})$

**S1**

Sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

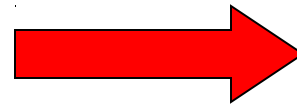


name	gpa
Dave	3.3
Jones	3.4
Smith	3.2
Smith	3.8
Madayan	1.8
Guldu	2.0

# Combine Selection and Projection

- Project name and gpa of students in S1 with gpa higher than 3.3:
- $\Pi_{\text{name,gpa}}(\sigma_{\text{gpa} > 3.3}(\text{S1}))$

Sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0



name	gpa
Jones	3.4
Smith	3.8

# Set Operations

- Union ( $R \cup S$ )
- All tuples in R or S (or both)
- R and S must have same number of fields
- Corresponding fields must have same domains
- Intersection ( $R \cap S$ )
- All tuples in both R and S
- Set difference ( $R - S$ )
- Tuples in R and not S

# Set Operations

- Cross product or Cartesian product ( $R \times S$ )
- All fields in  $R$  followed by all fields in  $S$
- One tuple  $(r,s)$  for each pair of tuples  $r \in R, s \in S$

# Example: Intersection

**S1**

sid	name	gpa
50000	Dave	3.3
53666	Jones	3.4
53688	Smith	3.2
53650	Smith	3.8
53831	Madayan	1.8
53832	Guldu	2.0

**S2**

sid	name	gpa
53666	Jones	3.4
53688	Smith	3.2
53700	Tom	3.5
53777	Jerry	2.8
53832	Guldu	2.0

sid	name	gpa
53666	Jones	3.4
53688	Smith	3.2
53832	Guldu	2.0

**$S1 \cap S2 =$**

# Joins

- Combine information from two or more tables
- Example: students enrolled in courses:



S1			E		
Sid	name	gpa	<u>cid</u>	grade	studid
50000	Dave	3.3			
53666	Jones	3.4	Carnatic101	C	53831
53688	Smith	3.2	Reggae203	B	53832
53650	Smith	3.8	Topology112	A	53650
53831	Madayan	1.8	History 105	B	53666
53832	Guldu	2.0			

**S1**

Sid	name	gpa
-----	------	-----

50000	Dave	3.3
-------	------	-----

53666	Jones	3.4
-------	-------	-----

53688	Smith	3.2
-------	-------	-----

53650	Smith	3.8
-------	-------	-----

53831	Madayan	1.8
-------	---------	-----

53832	Guldu	2.0
-------	-------	-----

sid	name	gpa
-----	------	-----

53666	Jones	3.4
-------	-------	-----

53650	Smith	3.8
-------	-------	-----

53831	Madayan	1.8
-------	---------	-----

53832	Guldu	2.0
-------	-------	-----

**Joins****E**

<u>cid</u>	grade	studid
------------	-------	--------

Carnatic101	C	53831
-------------	---	-------

Reggae203	B	53832
-----------	---	-------

Topology112	A	53650
-------------	---	-------

History 105	B	53666
-------------	---	-------

cid	grade	studid
-----	-------	--------

History105	B	53666
------------	---	-------

Topology112	A	53650
-------------	---	-------

Carnatic101	C	53831
-------------	---	-------

Reggae203	B	53832
-----------	---	-------



# Relational Algebra Summary

- Algebras are useful to manipulate data types (relations in this case)
- Set-oriented
- Brings some clarity to what needs to be done
- Opportunities for optimization
  - May have different expressions that do same thing
- We will see examples of algebras for other types of data in this course

# Intro to SQL

- CREATE TABLE
  - Create a new table, e.g., students, courses, enrolled
- SELECT-FROM-WHERE
  - List all CS courses
- INSERT
  - Add a new student, course, or enroll a student in a course

# Create Table

```
CREATE TABLE Enrolled  
(studid CHAR(20),  
cid CHAR(20),  
grade CHAR(20),  
PRIMARY KEY (studid, cid),  
FOREIGN KEY (studid) references Students)
```

# Select-From-Where query

“Find all students who are under 18”

SELECT \*

FROM Students S

WHERE S.age < 18

# Queries across multiple tables (joins)

“Print the student name and course ID where the student received an ‘A’ in the course”

```
SELECT S.name, E.cid
```

```
FROM   Students S, Enrolled E
```

```
WHERE  S.sid = E.studid AND E.grade = ‘A’
```

# Other SQL features

- MIN, MAX, AVG
  - Find highest grade in fall database course
- COUNT, DISTINCT
  - How many students enrolled in CS courses in the fall?
- ORDER BY, GROUP BY
  - Rank students by their grade in fall database course

# Views

- Virtual table defined on base tables defined by a query
  - Single or multiple tables
- Security – “hide” certain attributes from users
  - Show students in each course but hide their grades
- Ease of use – expression that is more intuitively obvious to user
- Views can be materialized to improve query performance

# Views

- Suppose we often need names of students who got a 'B' in some course:

```
CREATE VIEW B_Students(name, sid, course)
AS SELECT S.sname, S.sid, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.studid and E.grade = 'B'
```

Name	Sid	course
Jones	53666	History105
Guldu	53832	Reggae203



# Indexes

- Idea: speed up access to desired data
- “Find all students with  $\text{gpa} > 3.3$
- May need to scan entire table
- Index consists of a set of *entries* pointing to locations of each *search key*

# Types of Indexes

- **Clustered vs. Unclustered**
  - Clustered- ordering of data records same as ordering of data entries in the index
  - Unclustered- data records in different order from index
- **Primary vs. Secondary**
  - Primary – index on fields that include primary key
  - Secondary – other indexes

# Example: Clustered Index

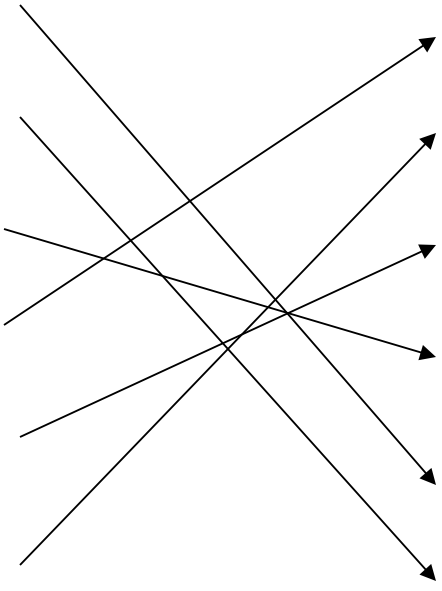
- Sorted by sid

	sid	name	gpa
50000	→ 50000	Dave	3.3
53600	→ 53650	Smith	3.8
	53666	Jones	3.4
53800	→ 53688	Smith	3.2
	→ 53831	Madayan	1.8
	53832	Guldu	2.0

# Example: Unclustered Index

- Sorted by sid
- Index on gpa

	sid	name	gpa
1.8	50000	Dave	3.3
2.0	53650	Smith	3.8
3.2	53666	Jones	3.4
3.3	53688	Smith	3.2
3.4	53831	Madayan	1.8
3.8	53832	Guldu	2.0



# Comments on Indexes

- Indexes can significantly speed up query execution
- But inserts more costly
- May have high storage overhead
- Need to choose attributes to index wisely!
  - What queries are run most frequently?
  - What queries could benefit most from an index?
- Preview of things to come: SDSS

# Summary: Why are RDBMS useful?

- **Data independence** – provides abstract view of the data, without details of storage
- **Efficient data access** – uses techniques to store and retrieve data efficiently
- **Reduced application development time** – many important functions already supported
- Centralized data administration
- Data Integrity and Security
- Concurrency control and recovery

# So, why don't scientists use them?

- “I tried to use databases in my project, but they were just too [slow | hard-to-use | expensive | complex] . So I use files”.
- Gray and Szalay, Where Rubber Meets the Sky: Bridging the Gap Between Databases and Science

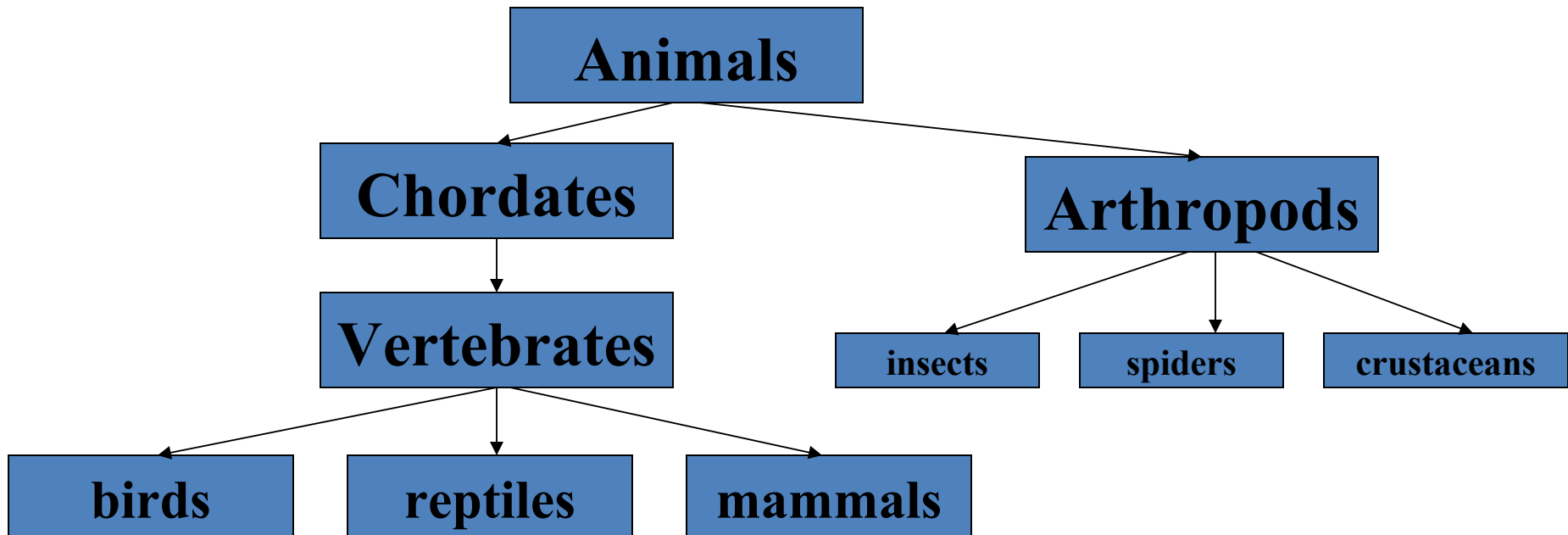
# Some other limitations of RDBMS

- Arrays
- Hierarchical data



# Example: Taxonomy of Organisms[8]

- Hierarchy of categories:
  - Kingdom - phylum – class – order – family – genus – species
  - How would you design a relational schema for this?



# References

1. [http://en.wikipedia.org/wiki/Entity%E2%80%93relationship\\_model](http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model)
2. [http://simple.wikipedia.org/wiki/Entity-relationship\\_model](http://simple.wikipedia.org/wiki/Entity-relationship_model)
3. <https://cs.uwaterloo.ca/~david/cs338/10%20ER%20Model.pdf>
4. <http://people.cs.pitt.edu/~chang/156/03ERmodel.html>
5. [http://www.aquafold.com/aquadatastudio/er\\_modeler.html](http://www.aquafold.com/aquadatastudio/er_modeler.html)
6. <http://searchdatamanagement.techtarget.com/guide/Limitations-of-entity-relationship-models-in-data-modeling>
7. Database System Concepts: Abraham Silberschatz, Henry F. Korth & S., Sudarshan, TATA Mcgraw Hill.
8. Database Systems Concepts, design and Applications 2/e, Singh S. K, Pearson Education
9. SQL- PL/SQL, Ivan bayross, BPB Publications.

# Images References

1. [1.http://www.tutorialspoint.com/dbms/er\\_model\\_basic\\_concepts.htm](http://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm)
2. [2.http://www.tutorialspoint.com/dbms/er\\_diagram\\_representation.htm](http://www.tutorialspoint.com/dbms/er_diagram_representation.htm)
3. <https://www.google.co.in/search?q=ER+Diagram+for+COMPANY+Database+images+in+database+management+system&biw=1024&bih=643&tbm=isch&tbo=u&source=univ&sa=X&ei=2a-qVImcMciJuATmvYLYCQ&ved=0CBsQsAQ>